AUTOMATED ANALYSIS OF MEDICAL RECORDS AND PREDICTIVE MODELING FOR FUTURE PROBLEMS IN PATIENTS USING MACHINE LEARNING ALGORITHMS

Dewantha A.A.A.R.S

(IT20618186)

BSc (Hons) in Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

April 2024

AUTOMATED ANALYSIS OF MEDICAL RECORDS AND PREDICTIVE MODELING FOR FUTURE PROBLEMS IN PATIENTS USING MACHINE LEARNING ALGORITHMS

Dewantha A.A.A.R.S (IT20618186)

***Supervisor**: Dr. Kapila Dissanayaka*

***Co-supervisor:** Mrs. Bhagyanie Chathurika*

Dissertation submitted in partial fulfilment of the requirement for the Bachelor of Information Technology

Specialization in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

April 2024

DECLARATION PAGE OF THE CANDIDATES & SUPERVISOR

I hereby declare that this is my original work and that no previously submitted materials for a degree or certificate from another university or institution of higher learning have been used in this proposal. To the best of my knowledge and belief, it doesn't include any content that has already been published or authored by someone else unless it specifically acknowledges it in the text.

Name	**Student ID**	**Signature**
Dewantha A.A.A.R.S	**IT20618186**	[illegible]

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

…………………………… ……………………………

Signature of the Supervisor Signature of the Co-supervisor

(Dr. Kapila Dissanayake) (Mrs. Bhagyanie Chathurika)

……………………………. …………………………….

Date Date

ABSTRACT

The number of people dying of heart disease in Sri Lanka is increasing day by day. Some heart conditions are manageable by the patient. But due to the inability to be aware of the risk situation in advance, many people are not interested in controlling certain factors. A lot of research has been done to diagnose heart conditions. Among them, much previous research has been conducted based on machine learning techniques to analyze the electronic health records of patients, including the most appropriate and accurate machine learning algorithms. My aim is to give prediction about the life risk that he or she will have to face in the future by identifying the current condition of the heart patient by using medical reports such as Electrocardiogram (ECG), Blood report, Magnetic resonance imaging (MRI) reports and comparing it with the previous condition. Here I hope to predict the level of risk in the form of a graph chart so that patients can understand. The main reason that our team suggested for creating a mobile application was that nowadays most people are tempted to use a smartphone. Mobile apps are easier to use and interact with people more quickly. Also, through this application, patients are anxious to identify their risk condition earlier and get proper treatment for it. Also, from the doctor's point of view, he can be aware of the risk conditions of the patient who comes to him and then check his sensitivity and give him treatment and advice.

Keyword: Electrocardiogram, Machine Learning, Electronic Health Record, Heart Disease, Predictive Modeling

ACKNOWLEDGEMENT

First and foremost, I am deeply thankful to Dr. Kapila Dissanayaka for his invaluable guidance, unwavering support, and insightful supervision throughout every phase of this research endeavor. His expertise and encouragement have been instrumental in shaping this study. I am equally indebted to Dr. Bhagayani Chathurika for her indispensable co-supervision, constructive feedback, and continuous encouragement, which significantly enriched the quality of this work.

Special appreciation is extended to Dr. Susith Athukorala of Apeksha Hospital, Dr. Subashini of Ragama Hospital, and Nursing Sister Chandrani Kumari of Monaragala Sirigala Hospital for their generous support and cooperation from the hospital side. Their expertise, assistance, and willingness to facilitate access to resources have been invaluable to the success of this research. Furthermore, I extend my gratitude to all the members of our research group who tirelessly contributed to various aspects of this project, from data collection to analysis and interpretation. Your dedication and collaborative spirit have been instrumental in overcoming challenges and achieving our research objectives.

Lastly, I would like to express my deepest appreciation to my family and friends for their unwavering encouragement, understanding, and patience throughout this journey. This research would not have been possible without the collective efforts, support, and encouragement of all those mentioned above, and for that, I am truly grateful.

Table of Contents

TABLE OF FIGURES

01. INTRODUCTION

1.1 Background

For the research projects carried out in the fourth academic year, my attention as well as the attention of other members of the group was focused on new technological applications related to the health sector. The health field is a field that is constantly being discovered and developed. Many of these discoveries have made it easier to treat various diseases. We can see clear evidence of how the new technology has been able to affect the health sector in the life of that time. Today, it is possible to reduce the risk of a person's life by using machine learning techniques and artificial intelligence to diagnose and draw comprehensive conclusions. Today, many people are able to predict the life risk of a patient by symptoms or other factors related to the disease.

The focus of all the members of our team was whether there was any method that would help or facilitate the heart attack patients and related specialists. We looked at how to use technology for that. Here we considered the use of machine learning techniques as the first step. We realized that the easiest way to do that is to create a mobile app. Here we mainly focused on four parts.

(1) Making preconceived notions about a person's likelihood of having a heart attack.

(2) Thus, according to the risk level of a person who is at risk of heart attack, finding out how much a person can be protected from heart attack through diet.

(3) To find out if there are methods that can be used to check a self-reported easy report of a heart attack patient.

(4) Also, finding the nearest and most suitable pharmacy for a heart attack patient to get the medication he needs when he is running out of medication.

Here, I assigned a patient mentioned in number **(03)** to test a new technology application using machine learning techniques through this mobile application to automatically analyze the ECG reports he receives.

- What is ECG?

An electrocardiogram (**ECG**), often known as an **EKG**, is a fast test to monitor heartbeat. It captures the electrical impulses produced by the heart. Test findings can be used to determine arrhythmias, or abnormal heartbeats, and heart attacks. Hospitals, operating rooms, medical offices, and ambulances all use ECG machines. Simple ECGs can be performed on some personal electronics, like smartwatches. Find out from your medical practitioner if this is a viable option for you. [1]

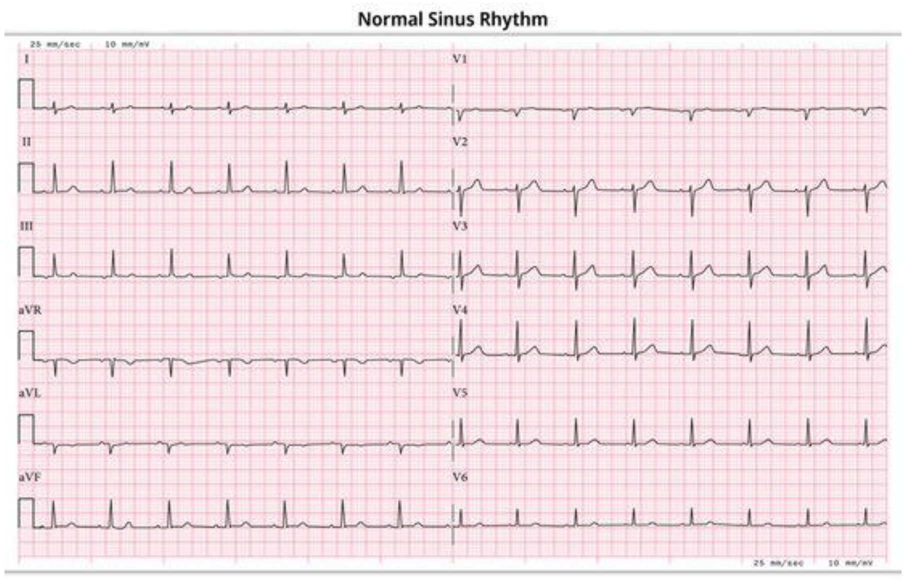


Figure 1 : ECG report of a healthy person.

The above is an ECG report image of a healthy person. To monitor heartbeat, an electrocardiogram, sometimes known as an EKG, is performed. It displays the heart rate in both rapid and slow waves. The findings of an ECG test can assist your care team in diagnosing:

- Arrhythmias are abnormal heartbeats.
- An earlier cardiac arrest.
- the reason for the chest pain. It might, for instance, exhibit indications of constricted or obstructed cardiac arteries.

To assess the effectiveness of heart disease medications and pacemaker functioning, an ECG may also be performed.

An ECG might be necessary if you have:

- ache in the chest.
- disorientation, lightheadedness, or dizziness.
- heartbeat that is pounding, skipping, or flickering.
- rapid heartbeat.
- breathlessness.

- tiredness or weakness.
- decreased capacity for physical activity.

An ECG may be necessary to check for cardiac disease if there is a family history of the condition, even. Any waveform deviations from normal ECG pattern represent numerous cardiac abnormalities, including,

- Cardiac Rhythm Disturbances (such as atrial fibrillation and ventricular tachycardia),
- Improper Blood Flow (such as Myocardial Ischemia and Myocardial Infraction), and
- Electrolyte Disturbances (such as Hypokalemia and Hyperkinemia)

Arrhythmias (Out of Rhythms) occur when the electrical signals that coordinate heartbeats do not work correctly. This may cause me to feel like a racing heart fluttering. With this condition, a person's heart may beat too quickly (Tachycardia), Too slowly (Bradycardia), too early (Premature contribution), or with an irregular rhythm (Flutter/Atrial Fibrillation). Many heart arrhythmias are harmless. However, if they are highly irregular, they can cause severe and potentially fatal symptoms and complications.

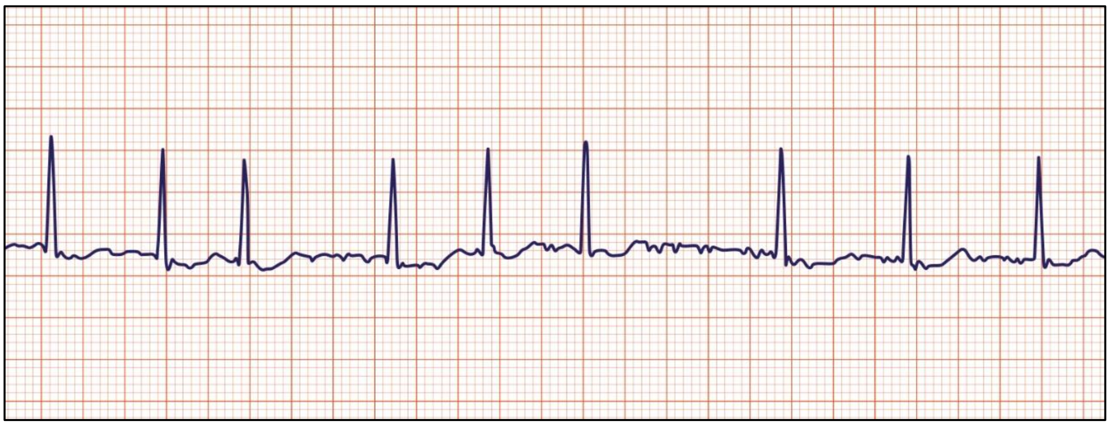


Figure 2: ECG with Arrhythmia (irregular rhythm)

Source: Atrial Fibrillation - ACLS Medical Training

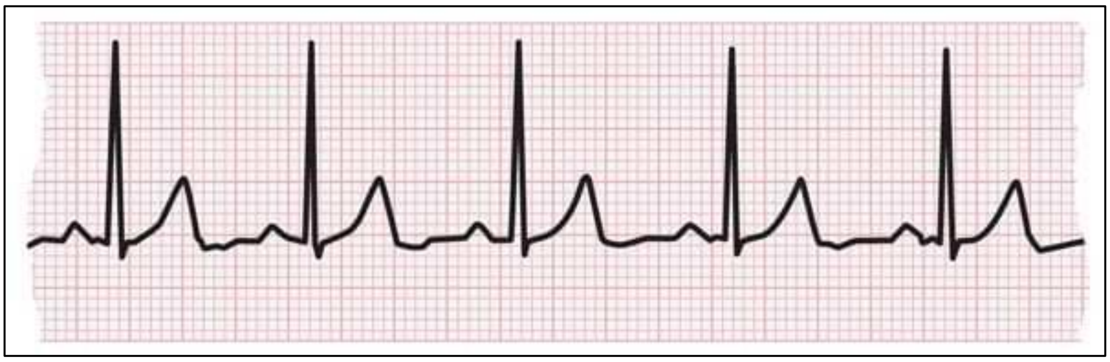


Figure 3: ECG with Normal Sinus Rhythm

In this presentation, only problem of Arrythmia is addressed. The objective is to distinguish the cases of Arrythmia, especially the “Irregular Heart Beats”.

What is QRS Complex?

Ventricular depolarization is represented by the “QRS complex,” which combines the Q, R, and S waves. Although not every ECG lead has all three of these waves, this word can be confusing because a "QRS complex" is still considered to be present. For instance, lead V1's regular QRS complex only consists of a R and a S wave instead of a Q wave, yet the combination of these two waves is still referred to as the lead's QRS complex.

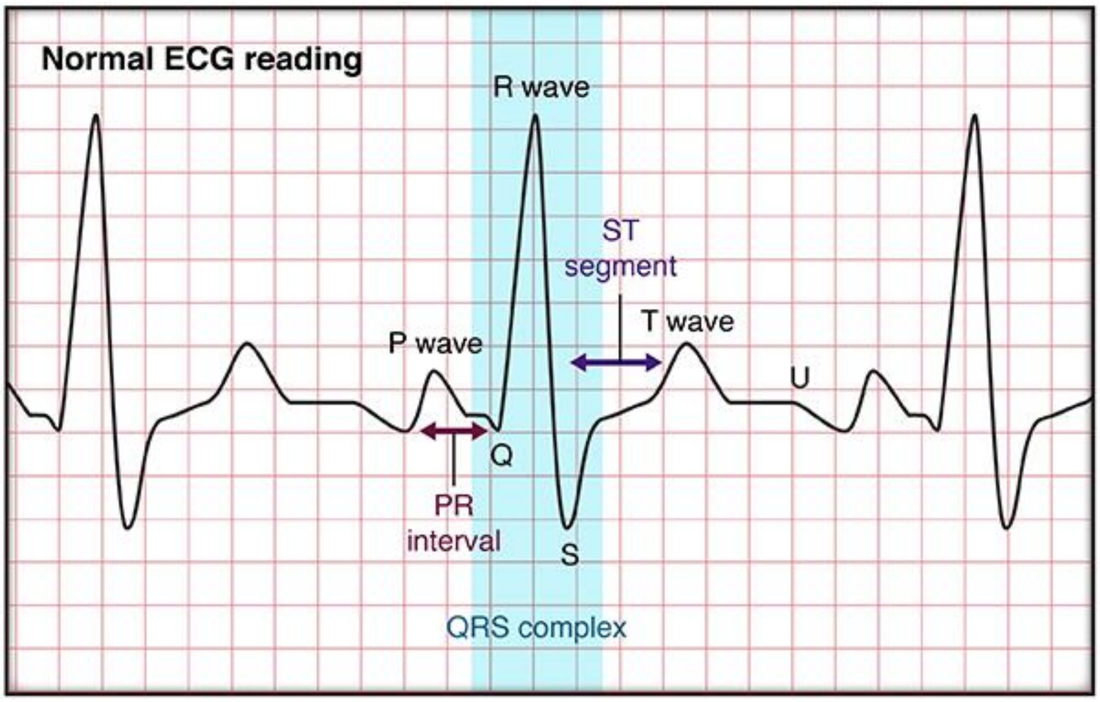


Figure 4: QRS Complex

Source: Arrhythmias - Stanford Medicine Children's Health

The QRS complex typically lasts between 0.08 and 0.10 seconds, or 80 and 100 milliseconds. This is known as its interval. The duration is considered intermediate or slightly prolonged when it falls between 0.10 and 0.12 seconds. If the QRS lasts more than 0.12 seconds, it is deemed abnormal.

When electrical activity takes a while to move throughout the ventricular myocardium, the QRS duration will get longer. The His-Purkinje system, which makes up the ventricles' typical conduction system, is made up of cells with a high rate of electrical conductivity. As a result, a normal QRS length is produced by the quick passage of an electrical impulse through the atrioventricular, or AV, node and on to the ventricles via the His-Purkinje system. The QRS duration is expanded, and a longer time is required when electrical activity goes from myocyte to myocyte rather than via the His-Purkinje pathway. When there is a right bundle branch block, left bundle branch block, non-specific intraventricular conduction delay, or ventricular arrhythmias such ventricular tachycardia, the QRS duration widens. [2]

Nowadays, deep learning (DL), machine learning (ML), and artificial intelligence (AI) are often discussed and used for a range of purposes. These terms are commonly misconstrued, and even if they are similar, it's crucial to know the distinctions between them. John McCarthy used the phrase "artificial intelligence" in the 1950s to refer to robots that could learn on their own1. Artificial Intelligence (AI) broadly refers to intelligent robots and algorithms that can carry out human-like activities like speech or picture recognition. A subclass of artificial intelligence (AI), machine learning first emerged in the 1980s and describes a program's capacity to identify patterns in data without explicit instruction.

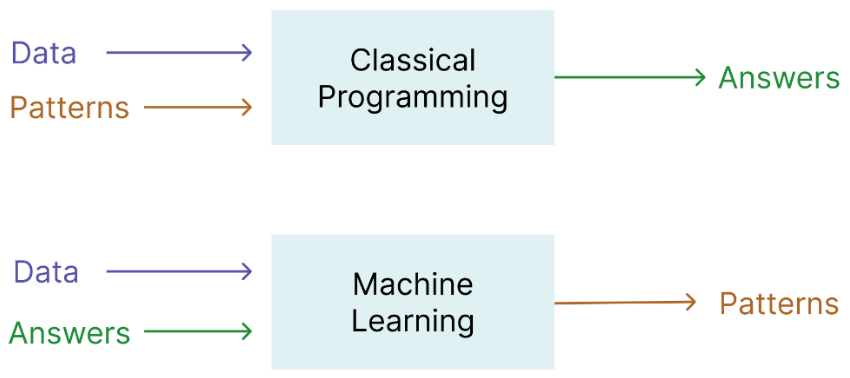


Figure 5: Analyze the data using Machine Learning (ML)

Source: https://cardiologs.com/

Machine learning algorithms are formed by feeding input data and replies (annotated data) into a program, allowing the software to create its own interpretation patterns. This contrasts with classical programming, which involves creating an algorithm with rules and giving data input to get an answer.

Deep learning was created as an additional subset of machine learning techniques throughout the last 20 years. The term "deep learning" refers to the application of algorithms known as deep neural networks (DNN), which are based on the human cerebral cortex and are specifically made to identify patterns in vast volumes of data and resolve complicated reasoning-based problems.

In summary, massive volumes of data are analyzed by deep neural networks, which then develop their own patterns of interpretation on their own. Neural networks function by millions of computations that are difficult to explain with a small set of rules, even if the programmer may control the data that is fed into the DNN to train it and the DNN's performance by comparing its output to reality.

It's this mystery of how the algorithm comes to its findings that's known as the "black box." This might be seen as a major obstacle to the widespread adoption of AI systems since it obscures the reasoning and methods used by algorithms to arrive at results. To encourage patient participation and build trust, clinicians-especially in the healthcare industry-need to understand the rationale behind recommendations.

In this report, I demonstrate that a great deal about DNNs can be explained, providing insight into their behavior and capabilities while also highlighting their limitations, allowing us to better understand why they work so well in some scenarios but also why they can malfunction occasionally. Deep learning relies on training a neural network to mimic human behavior and identify patterns in data or images, which is one of its fundamental features.

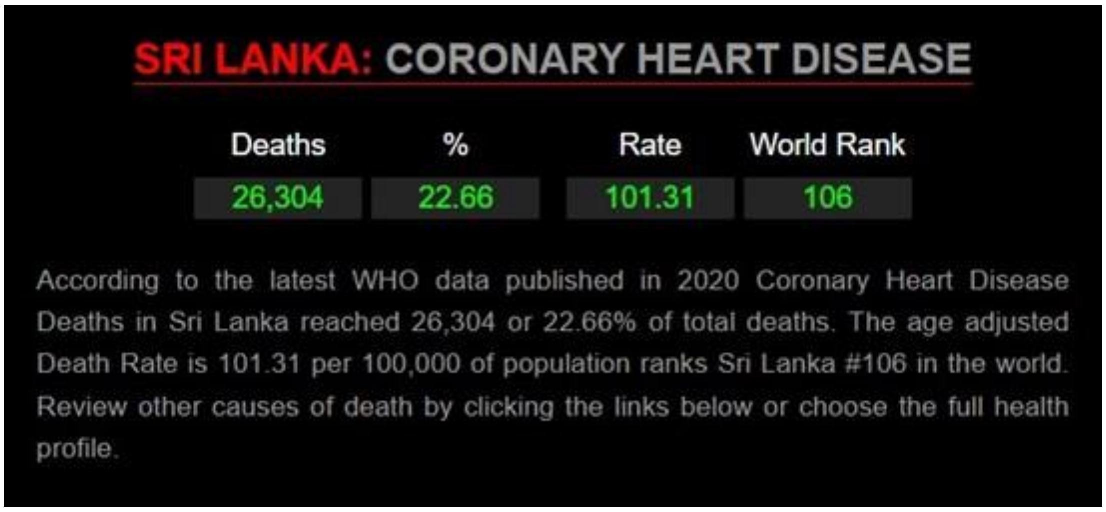
Although there are many other kinds of training techniques, supervised learning is the most widely used. To be more precise, the DNN must first be fed enormous volumes of data that have been annotated in relation to the question that the DNN must answer. Then, the algorithm needs to be validated and tested on separate datasets, to control and validate its performance without bias.

Here, it's critical to comprehend the significance of two terms: enormous volumes of annotated data. Large because, like a human expert, the algorithm needs a great deal of knowledge to be able to generate its own patterns of interpretation and ultimately distinguish one pattern from another. In addition to being enormous, the data must be indicative of data from the actual world. For example, all of those must be present in sufficient numbers in the training dataset if we are to be able to identify different kinds of irregularities.

Secondly, for the algorithm to identify which features of ECGs are linked to a particular kind of arrhythmia, the data must be annotated, which entails labeling each ECG strip in accordance with pre-specified anomalies. [3]

1.2 Literature Review

Heart disease is increasingly becoming prevalent in the world, and Sri Lanka marks no exception. According to data released by the World Health Organization (WHO) in 2020, over 22.6% of deaths in Sri Lanka are caused by cardiovascular diseases. [4] High blood pressure, high blood cholesterol and smoking have been identified as major factors that increase the risk of death from heart disease. Additionally, there are additional risk factors that can determine a person's likelihood of developing cardiovascular disease. Although variables no one can control, a person's age and ancestry play a significant role in assessing their susceptibility to heart disease. [5]



SRI LANKA: CORONARY HEART DISEASE

Deaths	%	Rate	World Rank
26,304	22.66	101.31	106

According to the latest WHO data published in 2020 Coronary Heart Disease Deaths in Sri Lanka reached 26,304 or 22.66% of total deaths. The age adjusted Death Rate is 101.31 per 100,000 of population ranks Sri Lanka #106 in the world. Review other causes of death by clicking the links below or choose the full health profile.

Figure 6: Coronary Heart Disease

Here, we consider helping people with preventable heart disease reduce their lifetime risk by providing advance notice of their risk status. Using the mobile application our team created for this project, people with heart illnesses can readily predict their doctor's recommendations for therapy. Through the AI model we have developed, the doctor can also predict about the patient's medical history who visits him in advance, as well as the likelihood of a heart attack and the risk that may materialize for a patient who is already experiencing a heart attack in the near future. Doctors no longer need to spend much time or effort comparing old and new records. The task I undertook in this section was to evaluate a patient's medical history, including an ECG or an EHR (Electronic Health Record), and determine the patient's future risk status.

Recent developments in automated analysis and predictive modeling for potential patient issues are made possible using machine learning algorithms in healthcare research. The application of machine learning techniques in the medical field has shown promise for transforming patient care, diagnosis, and prognosis. This research effort, which has a particular focus on cardiac patients, intends to build on previous work to develop a more thorough and precise predictive model by utilizing both medical records and electrocardiogram (ECG) data.

These initiatives represent the first steps towards creating predictive models. Previous research in the fields of automated analysis of medical records and predictive modeling for future problems in patients has yielded notable insights and achievements. Early studies primarily centered around applying traditional machine learning algorithms to electronic health records (EHRs) for identifying patterns and predicting medical conditions. Techniques such as decision trees, support vector machines, and logistic regression were used to extract relevant information from patient data.

Compared to other initiatives, medical and AI projects attracted more speculative funding in 2018 [6]. In 2017, “Predictive Modeling of Hospital Readmission Rates Using Electronic Medical Records-Wide Machine Learning: A Case-Study Using Mount Sinai Heart Failure Cohort” According to the research paper, despite current recommendations, predicting readmissions for heart failure (HF) remains difficult. The features used by current models are restricted. This study examined a data-driven methodology by examining every aspect from 1068 HF patients' electronic medical records (EMR). With a Naive Bayes approach and multistep feature selection, the model outperformed previous models in terms of accuracy (AUC=0.78). This shows that EMR-wide data can be used to enhance the prediction of HF readmissions, but more testing and improvement are required. They discuss their attempt to develop a data-driven, electronic medical record-wide (EMR-wide) feature selection approach and subsequent machine learning to predict readmission probabilities. [7]

In 2017, “Predictive Analytics in Health Care Using Machine Learning Tools and Techniques” According to the research paper, potent method for predicting and identifying patterns in big healthcare datasets is machine learning (ML). ML is very useful in the healthcare industry because of the intricacy of health informatics. With time, ML algorithms may learn and become more accurate, leading to more accurate forecasts and better patient care. Applications of machine learning (ML) in healthcare include illness prediction, tailored medication, and decision support systems. [8]

In 2019, “Predicting Future Cardiovascular Events in Patients with Peripheral Artery Disease Using Electronic Health Record Data” According to the research paper, Individuals suffering from peripheral arterial disease (PAD) are susceptible to significant unfavorable cardiac and cerebrovascular incidents. It is challenging to determine which individuals would benefit from more aggressive care since there are no easily accessible risk assessments that can reliably predict which patients are most likely to experience an incident. Therefore, our goal was to create a unique prediction model to determine whether individuals with PAD had a higher risk of experiencing serious adverse cardiac and cerebrovascular events by using machine learning techniques to data from electronic health records. Procedures and Outcomes: Patients at two tertiary care facilities who had been diagnosed with PAD provided the data.

A shared data architecture that supported the use of both structured (coded) and unstructured (text) data was used to construct predictive models. Models were utilized to look at the health system up till the diagnosis of PAD. To create and test the models, nested cross-validation was used. Our prediction models were trained on 7686 patients in total. With an area under the curve of 0.81 (95% CI, 0.80-0.83), our best predictive model, which made use of over 1000 factors, correctly identified the PAD patients who would eventually experience significant adverse cardiac and cerebrovascular events.

In conclusion: The enormous potential of electronic health records to provide automated risk stratification for cardiovascular diseases is highlighted by the ability of machine learning algorithms to learn models that accurately identify PAD patients at risk of major adverse cardiac and cerebrovascular events in the future. These algorithms are applied to data in the electronic health record. Widespread adoption of novel risk-stratification models may depend in part on common data models that facilitate cross-institution research and technological advancement. [9]

In 2019, "Machine learning in the electrocardiogram" According to the research paper, Since the ECG is the most often used diagnostic device for monitoring cardiac electrical activity, it is crucial to use it to find indicators for early diagnosis and detection. The field of machine learning in healthcare innovation has reemerged in recent years due to the massive increase in electronic health records, which contain a systematized collection of various types of digitalized medical data. These records also come with new tools that make it easier to analyze the massive amount of data.

The most current machine learning-based methods used for ECG applications are discussed in this study, along with benefits and drawbacks of their use. Deep learning and other forms of machine learning have been shown to be effective tools for helping doctors with risk assessment and patient screening. They do not, however, offer the physiological foundation for classification results. The interpretation and comprehension of important physiologically significant ECG biomarkers derived from machine learning approaches can be aided by computational modeling and simulation. [10]

In 2020, "Applications of machine learning predictive models in the chronic disease diagnosis" According to the research paper, Patients with pneumonia bear a heavy financial burden from hospital readmissions. The accuracy of current prediction models is frequently poor. In order to create a more accurate model for predicting readmissions for pneumonia patients within 30 days of discharge, this study looked at the application of machine learning. More than 1,500 individual's computerized health records were examined by researchers. Numerous data points were included in the machine learning model, such as vital signs, test findings, demographics, and prescription information. The model's accuracy in predicting readmissions was demonstrated by its AUC (Area Under the Curve) of 0.84. This shows that machine learning may be able to predict hospital readmissions for patients with pneumonia more accurately, which might result in improved patient outcomes and lower medical expenses. [11]

In 2020, “Cardiovascular disease prediction from electrocardiogram by using machine learning” According to the research paper, Globally, cardiovascular disease (CVD) is the primary cause of mortality. In Malaysia, CVD caused 13,503 deaths in 2017. The present methods for predicting CVD are typically expensive and intrusive. Machine learning (ML) approaches make use of the intricate relationships between pertinent risk factors to provide an accurate forecast. This work reports on a case-control study with sixty individuals from a prospective population-based initiative called The Malaysian Cohort. Systolic and diastolic blood pressure, total cholesterol, the R-R interval, and the root mean square of consecutive differences retrieved from the electrocardiogram (ECG) were the five measures that were statistically significant in predicting CVD.

Six ML algorithms, namely, linear discriminant analysis, linear and quadratic support vector machines, decision tree, k nearest neighbor, and artificial neural network (ANN), were evaluated to determine the most accurate classifier in predicting CVD risk. ANN, which achieved 90% specificity, 90% sensitivity, and 90% accuracy, demonstrated the highest prediction performance among the six algorithms. In summary, by utilizing ML techniques, ECG data can serve as a good parameter for CVD prediction among the Malaysian multiethnic population. [12]

In 2023, “Heart disease risk prediction using deep learning techniques with feature augmentation” According to the research paper, one of the biggest causes of death for the general public is cardiovascular disease. The likelihood of survival for individuals with cardiac problems is significantly impacted by late identification. Life-threatening cardiac disorders are known to be influenced by a number of factors, including age, sex, blood sugar, cholesterol, and heart rate. However, because there are so many variables, it can be challenging for an expert to assess each patient while taking this information into consideration. The authors of this publication suggest assessing a patient's risk of cardiovascular disease by combining deep learning algorithms with feature augmentation approaches. The results show a considerable improvement, even more so when it comes to an issue that affects a wide population. The suggested approaches beat other state of the art methods by 4.4%, resulting in a 90% accuracy. [13]

In 2023, “Predicts: An IoT and machine learning-based system to predict risk level of cardio-vascular diseases” According to the research paper, Globally, cardiovascular disease (CVD) continues to be a major health burden. This study suggests a wearable and machine learning-based mobile device to forecast the risk of CVD. The technology uses user data analysis to accurately classify users into risk categories (F1 score up to 91%). This demonstrates how these methods can lead to better healthcare outcomes and early CVD risk assessment. [14]

Another one in 2023, “End-to-end risk prediction of atrial fibrillation from the 12-Lead ECG by deep neural networks” According to the research paper, Millions of individuals worldwide suffer with atrial fibrillation (AF), one of the most prevalent cardiac arrhythmias. AF is strongly associated with a higher risk of cardiovascular disorders such heart failure and stroke. When assessing the likelihood of atrial fibrillation based on the ECG, machine learning techniques have demonstrated encouraging outcomes. Our goal is to create and assess one of these algorithms using a sizable CODE dataset that was gathered in Brazil. Methods: Without the use of additional digital indicators, we developed and tested a model for AF risk prediction for individual patients based on the raw ECG recordings using the CODE cohort.

The cohort consists of a set of ECG recordings annotated by the Brazilian Telehealth Network of Minas Gerais. For the purpose of classifying AF, a convolutional neural network built on a residual network architecture was used to generate class probabilities. The probabilities were utilized to create a Kaplan-Meier model for survival analysis and a Cox proportional hazards model. As a result, in individuals without AF, our model can estimate their probability of developing the illness.

Results: With an AUC value of 0.845, the deep neural network model identified individuals who did not exhibit AF in the given ECG but who will eventually develop AF. Our survival model indicates that patients in the minimal-risk group (i.e., with the probability of a future AF case being less than or equal to 0.1) have >85% chance of remaining AF free until after seven years, whereas patients in the high-risk group (i.e., with the probability of a future AF case being >0.7) are 50% more likely to develop AF within 40 weeks. In conclusion, we created and verified an AF risk prediction model. The model has the ability to offer significant and practical information for decision-making and patient care procedures when implemented in clinical practice. [15]

My aim is to give a new face to this system and technologies by comparing them with the previous research papers that I have considered above. Let us consider the new technical inputs and machine learning techniques that I have used for this system that I have built under the following topic.

1.3 Research Gap

My research topic is “Automated Analysis of Medical Records and Predictive Modeling for Future Problems in Patients Using Machine Learning Algorithms.” As I hope in this research, what I will do is analyze the data and factors obtained from a heart attack patient's ECG reports and blood reports using a machine learning algorithm and estimate the patient's life risk. Through the mobile application, both the patient and the doctor are provided with a graphical chart to monitor their progress according to the treatment he or she has received, the patient's current condition, and the risks condition that may occur in the future.

Research A – “Prediction of Diabetes Using Machine Learning Algorithms in Healthcare” According to this research paper [16], For experiment purpose, a dataset of patient’s medical record is obtained, and six different machine learning algorithms are applied on the dataset. Performance and accuracy of the applied algorithms is discussed and compared. Comparison of the different machine learning techniques used in this study reveals which algorithm is best suited for prediction of diabetes. The performance and accuracy of the machine learning algorithms used here are discussed based on the data sets.

Research B – “IoT based heart disease prediction and diagnosis model for healthcare using machine learning models” According to this research paper [17], To avail good service to the user using the online healthcare services, a fresh Cloud as well as IoT based Healthcare application to monitor in addition to diagnose serious diseases is developed. In this study, an efficient framework is utilized for heart disease is created utilizing the UCI Repository dataset as well as the healthcare sensors to predict the public who suffer from heart disease. Here, using both techniques called IoT sensors and amazing cloud computing, a person's blood pressure and ECG are taken into consideration to highlight whether he is a heart patient or not. This is a somewhat successful experiment [8].

Research C – “Predicting Future Cardiovascular Events in Patients with Peripheral Artery Disease Using Electronic Health Record Data” According to this research paper [9], Patients with peripheral artery disease (PAD) are at risk of major adverse cardiac and cerebrovascular events. There are no readily available risk scores that can accurately identify which patients are most likely to sustain an event, making it difficult to identify those who might benefit from more aggressive intervention. Thus, they aimed to develop a novel predictive model—using machine learning methods on electronic health record data—to identify which PAD patients are most likely to develop major adverse cardiac and cerebrovascular events.

According to the research A, B and C considered above, I was able to find new profiles for the component to be developed. I hope to include features such as using a mobile application, generating information and graphical graphs depicting the risk level of heart patients, automatically analyzing disease and medical records, and especially predicting the current condition of a heart patient as well as possible future risk conditions in this research.

Features	**A**	**B**	**C**	**Proposed System**
Mobile App	✗	✓	✗	✓
Generates informative diagrams or charts	✗	✗	✗	✓
Automated Analysis of Medical Records	✓	✓	✓	✓
predict future problems and condition in heart patients	✗	✗	**Predict future cardiovascular events**	✓
Machine Learning Algorithms	K-Nearest Neighbors (KNN), Naive Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression (LR) and Random Forest (RF)	logistic regression (LR), multilayer perception (MLP), support vector machine (SVM)	Decision Tree (DT), Logistic Regression (LR) and Random Forest (RF)	K-Nearest Neighbors (KNN), Convolutional Neural Networks (CNN), Support Vector Machines (SVM), Random Forest

1.4 Research Problem

Today, the number of deaths due to heart diseases has increased in Sri Lanka. The main reason for this is that patients suffering from heart diseases do not focus on controlling the disease. It is very important for people suffering from heart diseases to know the level of their disease tendency at present by following the treatment they receive, and the instructions prescribed by the doctors. It is important to be aware of it in advance in order not to allow your condition to become dangerous in the coming period. For this, what usually happens in Sri Lanka is that most people meet with heart disease specialists and show their medical records to gain awareness. Although this is a very successful method, if one wants to know the level of tendency in one's diseased nature now, it cannot be done. For that, those patients must bear their time, effort, and financial costs.

And nowadays, with the increase in population, there is also an increase in the number of patients. Therefore, the number of sick people who visit doctors every day is very high. Patients with minor heart failure as well as those suffering from severe heart disease visit the doctors. But doctors should devote equal time to both of them. Therefore, it is a difficult matter for doctors to choose the person who has an urgent need. When the doctor advises a patient to get ECG reports, the patient gets the same from the concerned medical center or hospital. After receiving that ECG report, he or she visits a doctor again and is anxious to know what his or her risk status is. Indeed, instead of meeting the doctor for that purpose, if he can do it by himself using technology, it is a great relief for the doctors as well as the patient who has to take the initiative.

According to my research topic, the medical records of a person with a heart disease will be automatically analyzed through an image processing machine learning algorithm to thereby identify the current condition of that patient as well as the risk situations that may occur in the future. Although many tests have been conducted related to early detection of heart failure, very little research has been done to detect this kind of life risky condition early for heart patient. And since this is used as a mobile application, the patient and the doctor can share information with each other, and the doctor can be aware of the current condition of his patient and possible future risks. If the doctor examines the dangerous condition and gives treatment for it first, the life risk of the patient can be reduced. I am very anxious to avoid some of the errors in them according to the research that has been done in the past.

1.5 Research Objective

1.5.1 Main Objective

Develop an automated system that utilizes machine learning algorithms to analyze ECG medical records, predict future heart disease-related issues, and generate a graph chart for visual representation of the patient's current and predicted future status.

1.5.2 Specific Objectives

- To study and understand what the conditions and factors heart patients are.
- To study and understand analyzing ECG and Other electronic medical records.
- Collecting a diverse and representative dataset of ECG records.
- Preprocessing the ECG data to remove noise and artifacts using machine learning algorithms.
- Develop a user-friendly interface for generating graph charts to the doctor and the patient, to take care of patient’s risk.
- Create a No-SQL database for store patient’s medical reports and ECG Record Images

02. METHODOLOGY

2.1 Methodology

Data Collection and Dataset

Here, the mobile application developed by our team is used by the doctors who examine heart attack patients and their respective heart attack patients. What I am doing here is to create a new item using machine learning technique to self-analyze patient's ACG records. I collected the relevant ECG reports from the Maharagama General Hospital in Sri Lanka. Reports were obtained under four categories. If they are,

1. Normal Person ECG Images (284x12=3408)
2. ECG Images of Patient that have abnormal heartbeat (233x12=2796)
3. ECG Images of Patient that have History of MI (172x12=2064)
4. ECG Images of Myocardial Infarction Patients (240x12=2880)

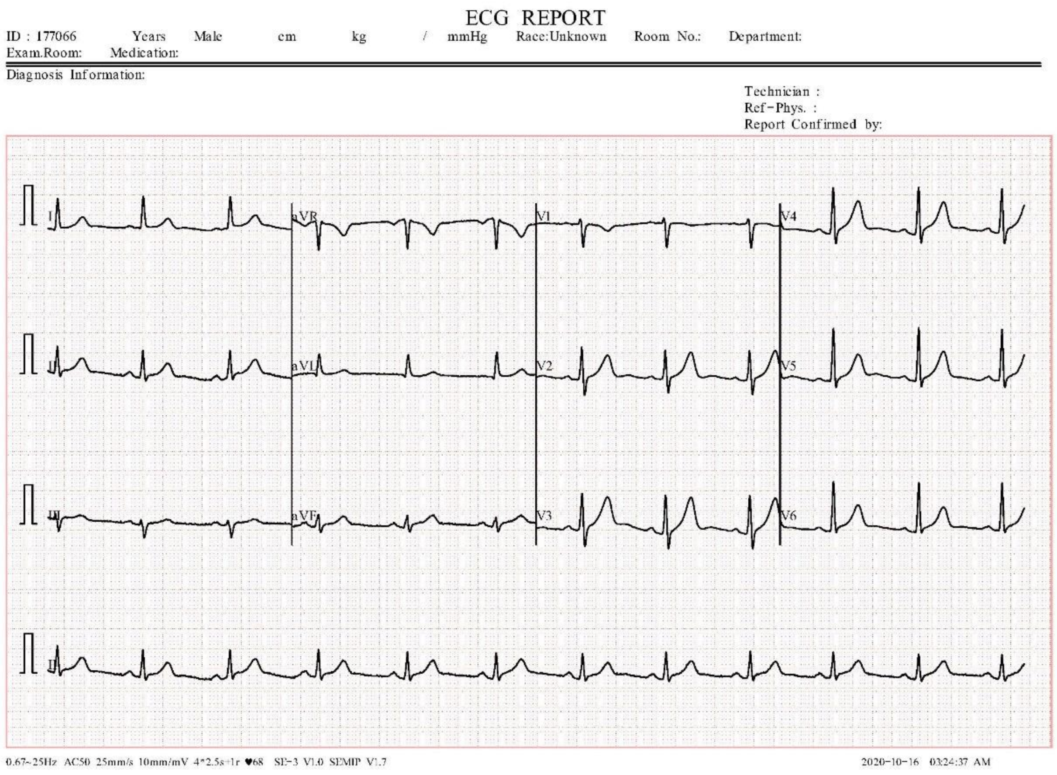


Figure 7: Normal Person ECG Images (284x12=3408)

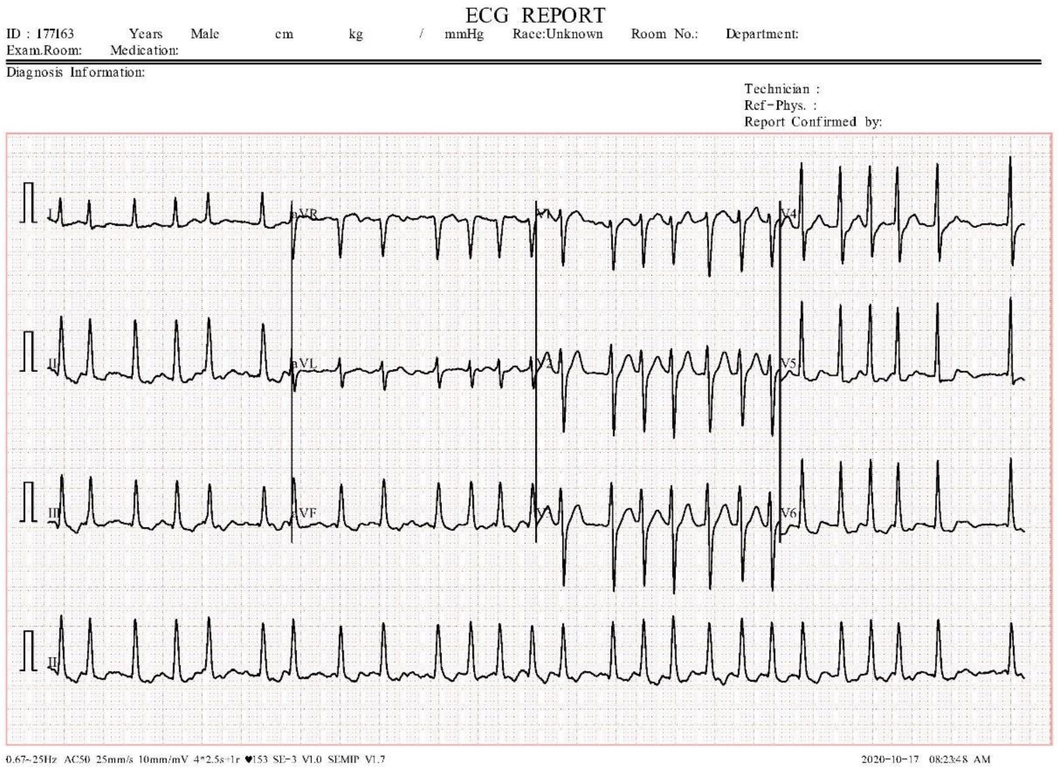


Figure 8: ECG Images of Patient that have abnormal heartbeat (233x12=2796)

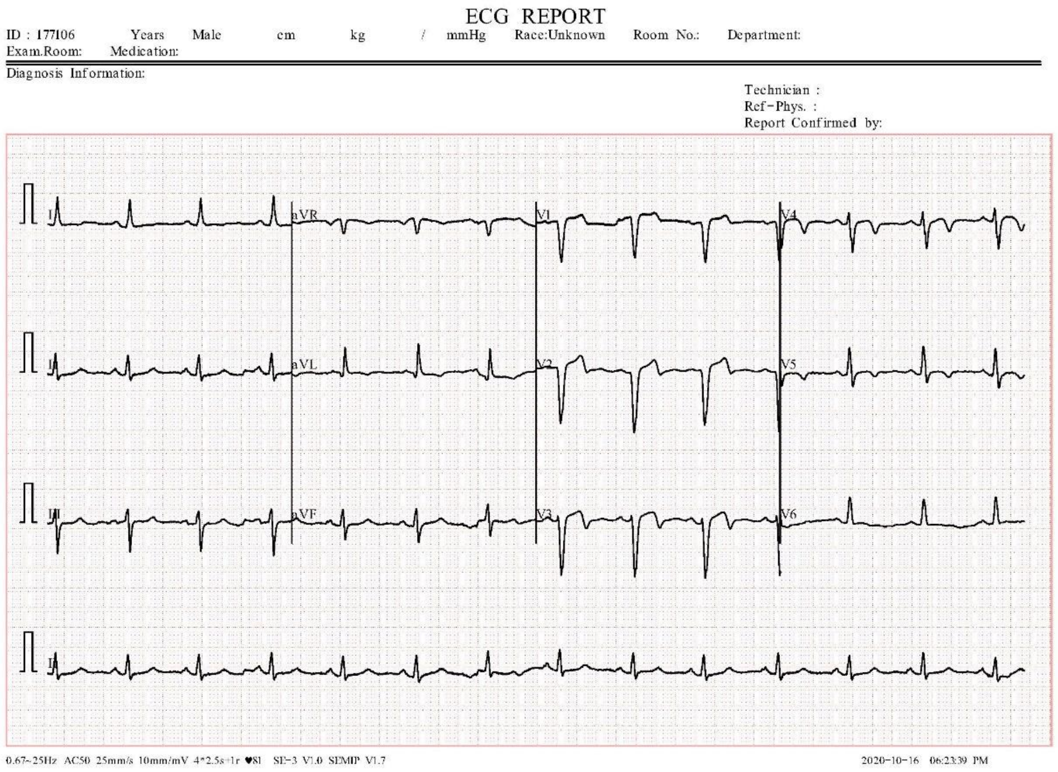


Figure 9: ECG Images of Patient that have History of MI (172x12=2064)

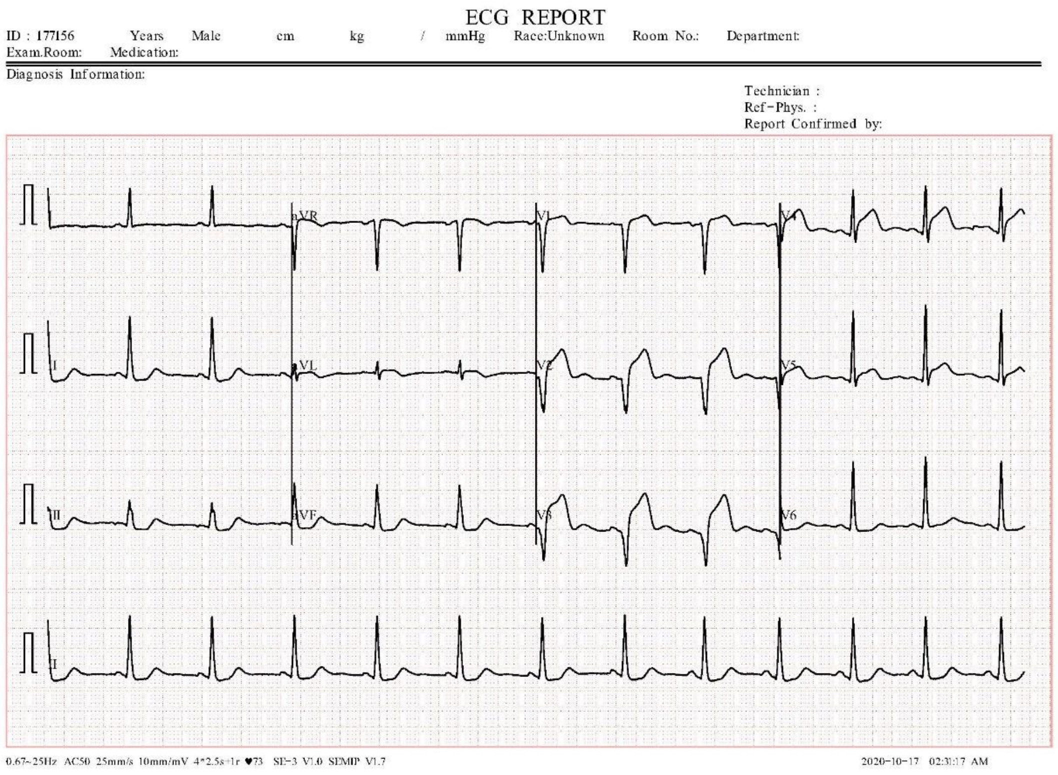


Figure 10: ECG Images of Myocardial Infarction Patients (240x12=2880)

Also, necessary knowledge was obtained from a heart disease specialist doctor to analyze the reports. A practical application was substituted for this using the knowledge gained there and the machine learning techniques I discovered.

Methods

The first thing I decided to use for this was the **CNN** machine learning technique. According to the photographs of the ECG records I took there; the photograph can be used for analysis according to the way the waves in it vary. But it is not possible to observe those waves precisely. The reason for this is that the clarity of the ECG reports that I had obtained on enlarging the photocopies was reduced. Then I used **KNN** machine learning technique for this. Here I also used machine learning techniques called **SVM**, **RNN**. But the machine learning technique known as KNN was successful among them.

2.2 Commercialization Aspects of the Product

Today all fields are getting connected with technology. It is a fact that we all know that through the new discovery of the world, through new ideas, technological development as well as social development. Technological innovations are emerging using artificial intelligence and machine learning techniques. Today, the world is focused on talking about artificial intelligence, investigating, and researching it. It is considered by many as a new breakthrough in technology.

Today, there is a great advancement in medicine with technology. This mobile application created by our teams can be of great help to heart patients and heart specialists. The mobile app is very helpful in saving time, managing risk situations, facilitating daily tasks, and providing guidance. Here the new item that I have made is important for people, in fact it is important for patients suffering from heart disease. They often need to contact a doctor to understand the ECG reports they receive from time to time. There is no difficulty here. But by creating a self-analysis application for that, it is easy for those patients to look at their ECG reports from home and understand the risk situations they are facing and get an understanding of the measures to be taken to control the environment and the risk situation.

There is a huge demand for technological inputs related to the health sector in the world. Not only that, but it is also normal that many things become more efficient with the advancement of technology. Along with this, there is a huge demand in the world market for such mobile applications designed to control risky situations.

The mobile application itself, created using machine learning techniques, can be called a new creation related to artificial intelligence. It is a known fact that everything is designed so that everything can be taken care of on the mobile phone. This is very important for many people who lead a busy life in their daily life. Everything is designed to facilitate time management. At the same time, mobile phones are used today to perform tasks efficiently and accurately.

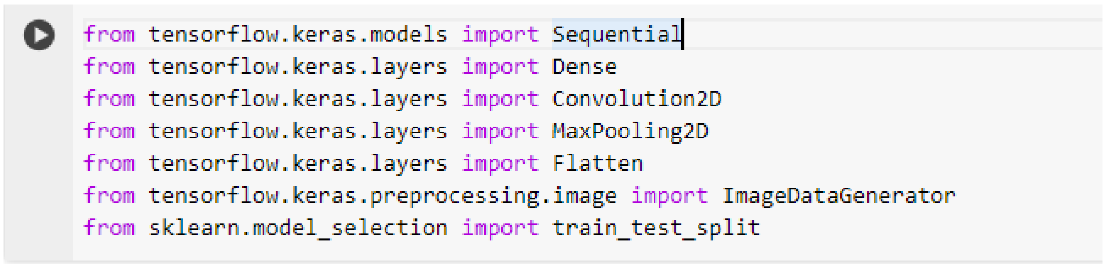
At a time when every process came to a standstill due to the Corona virus, it was seen that all the tasks of daily life were being done online. Online teaching, online medical consultation, online bill payment etc. were all done through mobile phone. This means that in the future, everything that happens in daily life will be done online. Therefore, we must always create for the future. Communication between doctor and patient can be achieved through mobile phones. Also, this mobile application is very important to prevent heart disease and reduce the number of deaths caused by it.

2.3 Testing and Implementation

Implementation

The first thing I decided to use for this was the **CNN** machine learning technique. According to the photographs of the ECG records I took there; the photograph can be used for analysis according to the way the waves in it vary. But it is not possible to observe those waves precisely. The reason for this is that the clarity of the ECG reports that I had obtained on enlarging the photocopies was reduced. Then I used **KNN** machine learning technique for this. Here I also used machine learning techniques called **SVM**, **RNN**. But the machine learning technique known as **KNN** was successful among them.

Import Libraries



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
```

Figure 11: Import TensorFlow's Kera's libraries.

Using TensorFlow's Kera's library, this code creates a Convolutional Neural Network (CNN) model. Below is a summary of the functions of each line:

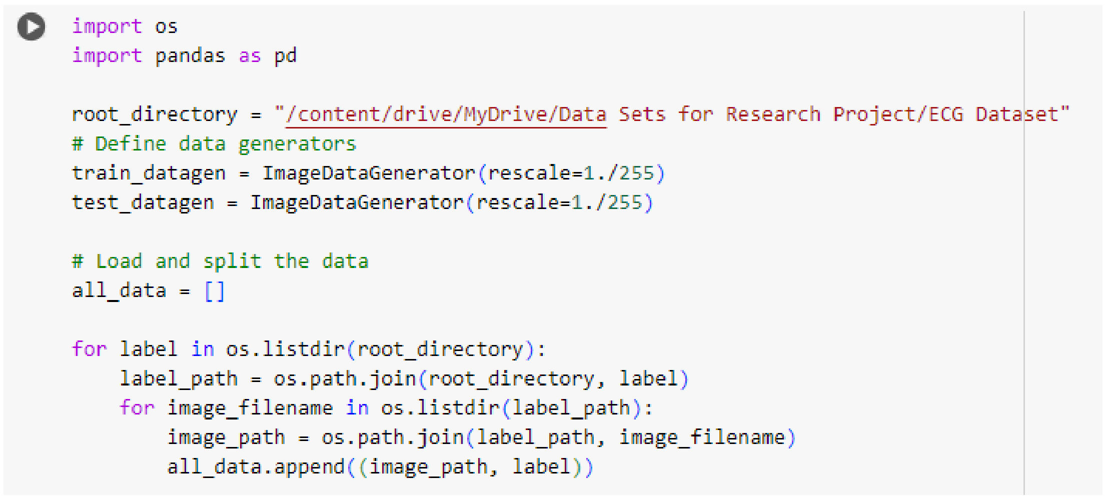
1. '**from keras.models.tensorflow import Sequential**': To construct a sequential model, or a linear stack of layers, this line imports the **Sequential** class from Keras.

2. '**Import Dense, Convolution2D, MaxPooling2D, Flatten from tensorflow.keras.layers**': These lines import several kinds of layers that are going to be utilized in the model. Fully connected layers utilize **dense**, convolutional layers use **convolution2D**, max pooling layers use **max pooling2D**, and convolutional layer output is flattened using flatten.

3. 'import image from tensorflow.keras.preprocessing. ImageDataGenerator': The '**ImageDataGenerator**' class, which is utilized for preprocessing and real-time data augmentation of pictures, is imported in this line.

4. ‘**from train_test_split import sklearn.model_selection**’: The ‘train_test_split’ function from scikit-learn, which divides the dataset into training and testing sets, is imported in this line.

In image classification tasks, this code is usually used to train the CNN model to identify patterns in pictures. The performance of the model may be enhanced by preprocessing the photos and augmenting the dataset using the ‘**ImageDataGenerator**’. To guarantee that the model is tested on untested data and provides a more accurate assessment of its performance, utilize the ‘**train_test_split**’ function.


```
import os
import pandas as pd

root_directory = "/content/drive/MyDrive/Data Sets for Research Project/ECG Dataset"
# Define data generators
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

# Load and split the data
all_data = []

for label in os.listdir(root_directory):
    label_path = os.path.join(root_directory, label)
    for image_filename in os.listdir(label_path):
        image_path = os.path.join(label_path, image_filename)
        all_data.append((image_path, label))
```
Figure 12: Load and Split the data.

This code is used to load and preprocess an ECG dataset for use in a machine learning model. Here's a breakdown of what each line does:

1. ‘**Import os**’: This command imports the operating system-dependent functionality (the os module).

2. '**import pandas as pd**': This line, following standard practice, imports the '**pandas**' library and aliases it as '**pd**'.

3. "/content/drive/MyDrive/Data Sets for Research Project/ECG Dataset" is the **root directory**. The ECG dataset's root directory is specified by this line.

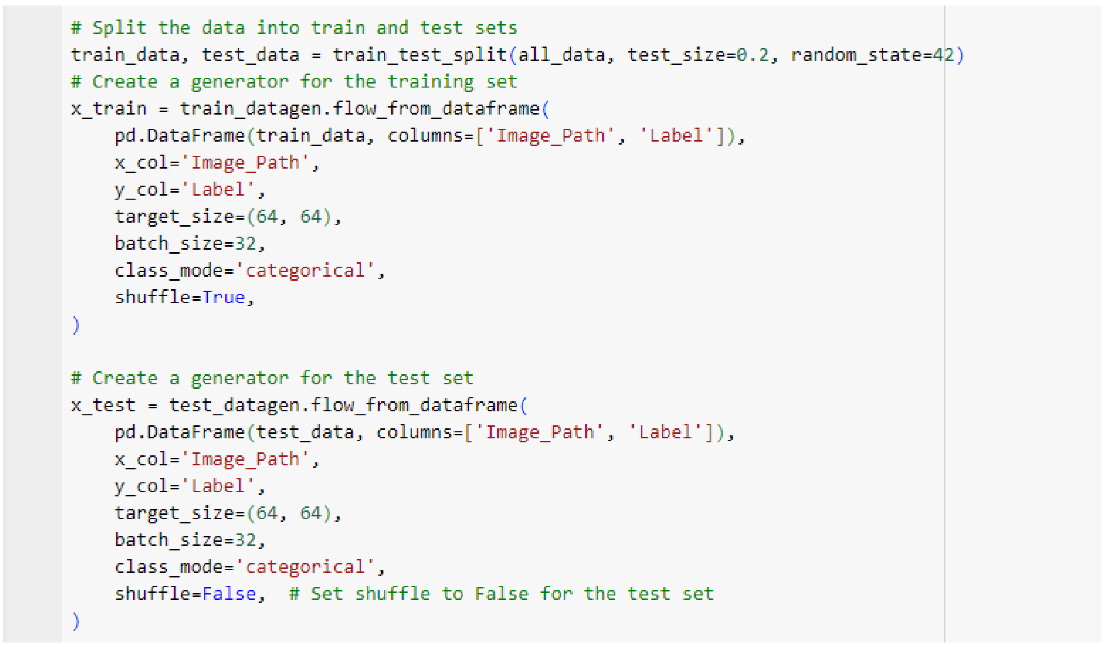
4. "**train_datagen**" and "**test_datagen**" are instances of ImageDataGenerator functions with rescale values of 1. /255 and 1, respectively. For the training and testing sets, these lines produce data generators, respectively. The photos' pixel values are scaled to fall between 0 and 1 using the '**rescale**' argument.

5. "**all_data = []**": This line initializes a blank list that will hold the paths to the picture files and the labels that go with them.

6. Every label directory in the root directory and every image file in the label directory are iterated over by the nested '**for**' loops.
7. '**os.path.join(label_path, image_filename)**' for **image_path**: The complete path to every picture file is created by this line.

8. '**all_data.append((image_path, label))**': The image path and matching label are appended to the all_data list with this line.

This code loads the ECG dataset and does pixel-by-pixel scaling to the pictures as a preprocess. A machine learning model may be trained and tested using the generated **'all_data'** list.



```
# Split the data into train and test sets
train_data, test_data = train_test_split(all_data, test_size=0.2, random_state=42)
# Create a generator for the training set
x_train = train_datagen.flow_from_dataframe(
    pd.DataFrame(train_data, columns=['Image_Path', 'Label']),
    x_col='Image_Path',
    y_col='Label',
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical',
    shuffle=True,
)

# Create a generator for the test set
x_test = test_datagen.flow_from_dataframe(
    pd.DataFrame(test_data, columns=['Image_Path', 'Label']),
    x_col='Image_Path',
    y_col='Label',
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical',
    shuffle=False,  # Set shuffle to False for the test set
)
```

Figure 13: create a generator for the train set and test set.

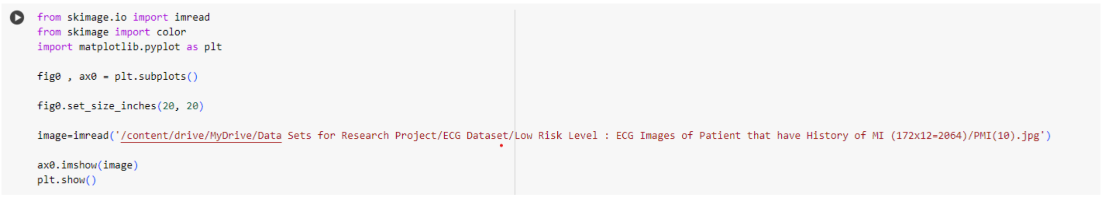
The ECG dataset is divided into training and testing sets using this code, and data generators are made for each set. Below is a summary of the functions of each line:

1. '**train data, test data = train test split (all data, test size = 0.2, random state = 42)**': Using the scikit-learn '**train_test_split**' function, this line divides the '**all_data**' list into training and testing sets. 20% of the data will be utilized for testing because the '**test_size**' option is set to 0.2. For repeatability, the random state parameter is set to 42.

2. "**train_datagen.flow_from_dataframe(...)**" for **x_train**: Using the '**flow_from_dataframe**' function of the 'train_datagen' object, this line generates a data generator for the training set. The **'pd.DataFrame'** function is used to generate a **dataframe** from the **'train_data'** list, which is sent into the procedure. The columns in the **dataframe** containing the picture paths and labels are indicated by the **'x_col'** and **'y_col'** arguments, respectively. The photos will be downsized to 64x64 pixels as the **'target_size'** argument is set to '**(64, 64)**'. The photos will be fed into the model in batches of 32 since the **'batch_size'** parameter is set to 32. Since 'categorical' is the value of the **'class_mode'** option, one-hot encoding of the labels is expected. This indicates that the photos will be shuffled before being input into the model because the '**shuffle**' option is set to '**True**'.

3. The expression {**x_test = test_datagen.flow_from_dataframe(...)**} Using the `**flow_from_dataframe**} function of the `**test_datagen**` object, this line generates a data generator for the testing set. The `**pd.DataFrame**` function is used to generate a dataframe from the `**test_data**` list, which is supplied as input to the method. With the exception of setting the `**shuffle**` parameter to `**False**}, which prevents the pictures from being shuffled before being given to the model, the parameters are the same as those for the training set.

Using this code, data generators for the training and testing sets are made. These generators feed the model photos in batches and carry out preprocessing and real-time data augmentation. Larger datasets can be used more effectively and efficiently by using the data generators rather than putting the complete dataset into memory.


```
from skimage.io import imread
from skimage import color
import matplotlib.pyplot as plt

fig0 , ax0 = plt.subplots()

fig0.set_size_inches(20, 20)

image=imread('/content/drive/MyDrive/Data Sets for Research Project/ECG Dataset/Low Risk Level : ECG Images of Patient that have History of MI (172x12=2064)/PMI(10).jpg')

ax0.imshow(image)
plt.show()
```
Figure 14: display an ECG image for visual inspection.

Using the matplotlib and scikit-image packages, this code shows an ECG picture. Below is a summary of the functions of each line:

1. **Imreading** a picture file is done by importing the `**imread**` function from the `**skimage.io**` module with this line: {**from skimage.io import imread**}.

2. {**from skimage import color**}: This command imports the color module, which has methods for handling color images, from `**scikit-image**`.

3. {**import matplotlib.pyplot as plt**}: This command imports the `**matplotlib.pyplot**` module and, in keeping with standard practice, aliases it as **plt**.

4. `**fig0, ax0 = plt.subplots()**`: This line uses the `**matplotlib.pyplot**` subplots method to construct a new figure and axis object.

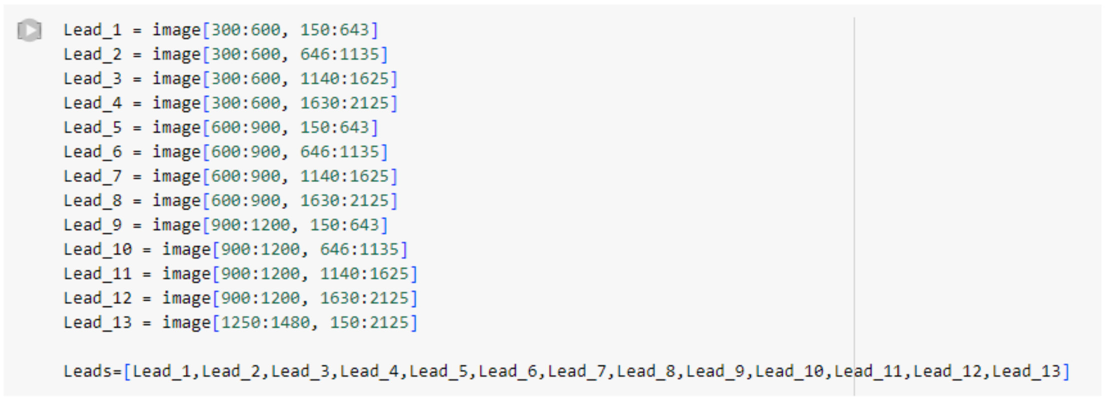
5. The line {**fig0.set_size_inches(20, 20)**} establishes the dimensions of the figure as 20 by 20 inches.

6. **"/content/drive/MyDrive/Data Sets for Research Project/ECG Dataset/Low Risk Level: ECG Images of Patients with MI History (172x12=2064)/PMI(10).jpg**").jpg Using the `**imread**` function, this line reads the ECG image file and puts it in the `**image**` variable.

7. `**ax0.imshow(image)**`: This line uses the `**imshow**` function from `**matplotlib.pyplot**` to display the ECG picture on the axis object.

8. `**plt.show()**`: The figure with the ECG picture is shown in this line.

An ECG picture is shown with this code so that it may be examined visually. The image file is read using the **imread** function, and the picture is shown on the figure using the **imshow** function. The figure's size may be adjusted using the set_size_inches function, and it can be shown using the show function.



```
Lead_1 = image[300:600, 150:643]
Lead_2 = image[300:600, 646:1135]
Lead_3 = image[300:600, 1140:1625]
Lead_4 = image[300:600, 1630:2125]
Lead_5 = image[600:900, 150:643]
Lead_6 = image[600:900, 646:1135]
Lead_7 = image[600:900, 1140:1625]
Lead_8 = image[600:900, 1630:2125]
Lead_9 = image[900:1200, 150:643]
Lead_10 = image[900:1200, 646:1135]
Lead_11 = image[900:1200, 1140:1625]
Lead_12 = image[900:1200, 1630:2125]
Lead_13 = image[1250:1480, 150:2125]

Leads=[Lead_1,Lead_2,Lead_3,Lead_4,Lead_5,Lead_6,Lead_7,Lead_8,Lead_9,Lead_10,Lead_11,Lead_12,Lead_13]
```

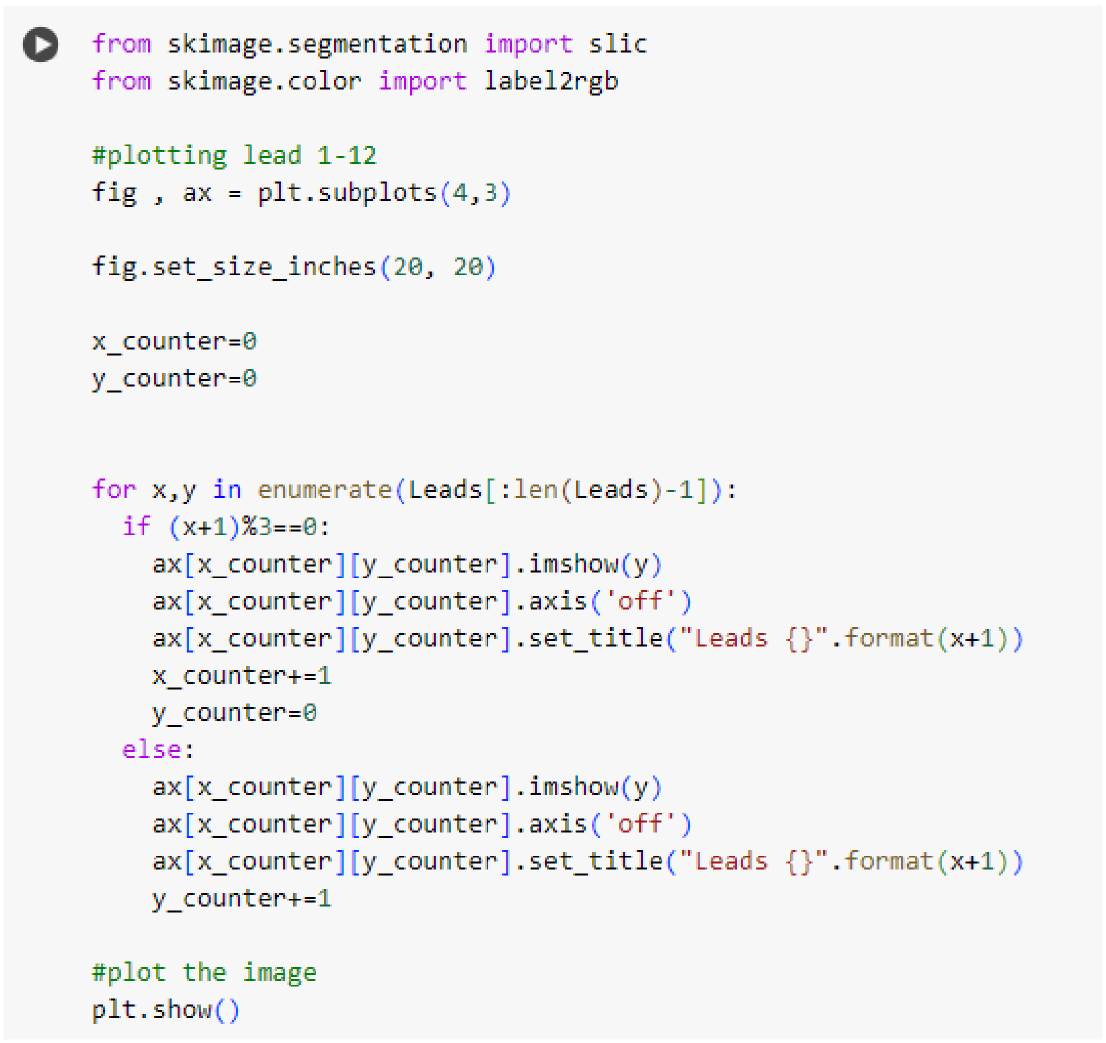
Figure 15: 13 Leads in Image of ECG Record.

The 12-lead ECG signals may be extracted from an ECG picture using this code. Below is a summary of the functions of each line:

1. The 12 leads of the ECG signal are extracted from the original ECG picture in the first 12 lines of code. By splitting the picture array with the correct indices, each lead is retrieved. The lead's position in the original ECG picture is used to determine the index selection.

2. 'Leads' is a list created by the final line of code that holds the 12 retrieved leads.

The 12-lead ECG signals are extracted from an ECG picture using this code, which is then utilized for additional processing and analysis. The 'leads' that are retrieved are kept in a list named Leads, which is utilized to process the signals through several operations including classification, feature extraction, and filtering. Depending on which exact ECG picture is being utilized, different indices may be used to extract the leads, which are dependent on where the leads are located in the original image.



```python
from skimage.segmentation import slic
from skimage.color import label2rgb

#plotting lead 1-12
fig , ax = plt.subplots(4,3)

fig.set_size_inches(20, 20)

x_counter=0
y_counter=0


for x,y in enumerate(Leads[:len(Leads)-1]):
  if (x+1)%3==0:
    ax[x_counter][y_counter].imshow(y)
    ax[x_counter][y_counter].axis('off')
    ax[x_counter][y_counter].set_title("Leads {}".format(x+1))
    x_counter+=1
    y_counter=0
  else:
    ax[x_counter][y_counter].imshow(y)
    ax[x_counter][y_counter].axis('off')
    ax[x_counter][y_counter].set_title("Leads {}".format(x+1))
    y_counter+=1

#plot the image
plt.show()
```

Figure 16: plotting lead 1-12.

The 12-lead ECG signals that were extracted from the ECG picture using the previously given code are visualized using this code. It generates a subplot grid with one of the collected leads displayed in each subplot. The matplotlib library's `imshow` function is used to illustrate the leads.

1. Bring in the required libraries: import label2rgb from skimage.color, import slic from skimage.segmentation, and import pyplot as plt from matplotlib.

2. Using plt.subplots, create a figure and a collection of subplots. The 20x20-inch figure size is selected.

3. Set x_counter and y_counter to zero upon initialization. The location of the subplots in the grid is tracked using these variables.

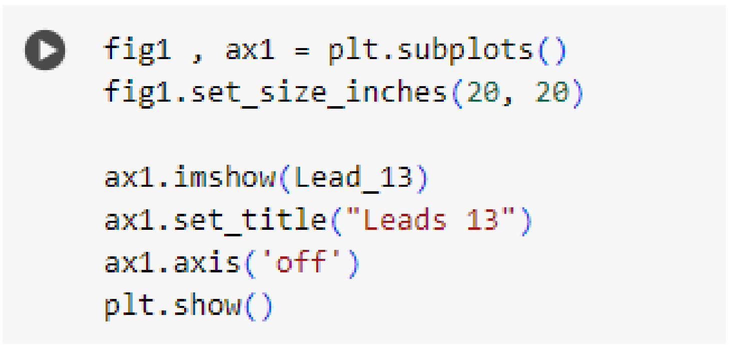
4. Using a for loop and an enumerate function, iterate over the leads. The index and value of every lead in the list may be obtained using the enumerate function.

5. Use the modulo operator to see if the current index is a multiple of three. Should such be the case, change the y_counter variable to 0 and increase the x_counter variable by 1. When the index hits a multiple of three, this is what's done to add a new row of subplots to the grid.

6. Just increase the y_counter variable by 1 if the current index is not a multiple of 3. When the index falls short of a multiple of three, this is the action used to generate a new subplot in the grid's current row.

7. To display the matching lead in each subplot, use the imshow function from the matplotlib package. The axis function from the matplotlib package is used to disable the axis of each subplot.

Lastly, use the matplotlib library's show function to display the figure. This code visualizes the 12-lead ECG data to facilitate analysis and comprehension. The signals from various leads may be more easily compared and analyzed when they are presented in a grid format.



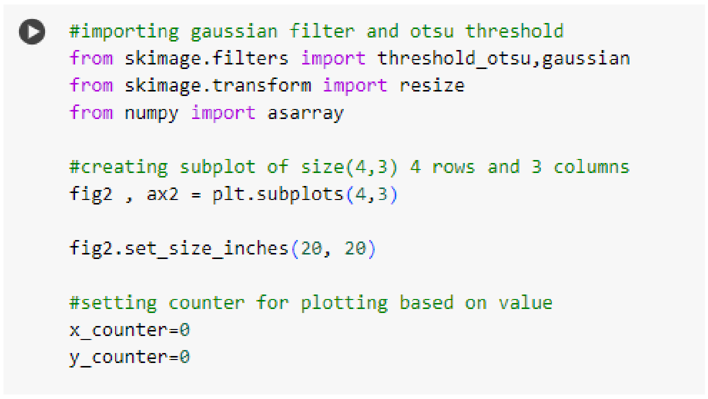
```
fig1 , ax1 = plt.subplots()
fig1.set_size_inches(20, 20)

ax1.imshow(Lead_13)
ax1.set_title("Leads 13")
ax1.axis('off')
plt.show()
```

Figure 17: Show lead 13.

The code that was previously supplied is used to extract the 13th lead of the ECG signal from the original ECG picture and view it.

1. Import the required library by importing Pyplot as plt from Matplotlib.
2. Using plt.subplots, create a figure and a collection of subplots. The 20x20-inch figure size is selected.
3. To see the 13th lead in the subplot, use the matplotlib library's imshow function.
4. Using the set_title function from the matplotlib package, change the subplot's title to "Leads 13".
5. Using the axis function from the matplotlib package, disable the subplot's axis.
6. Lastly, use the matplotlib library's show function to display the figure.

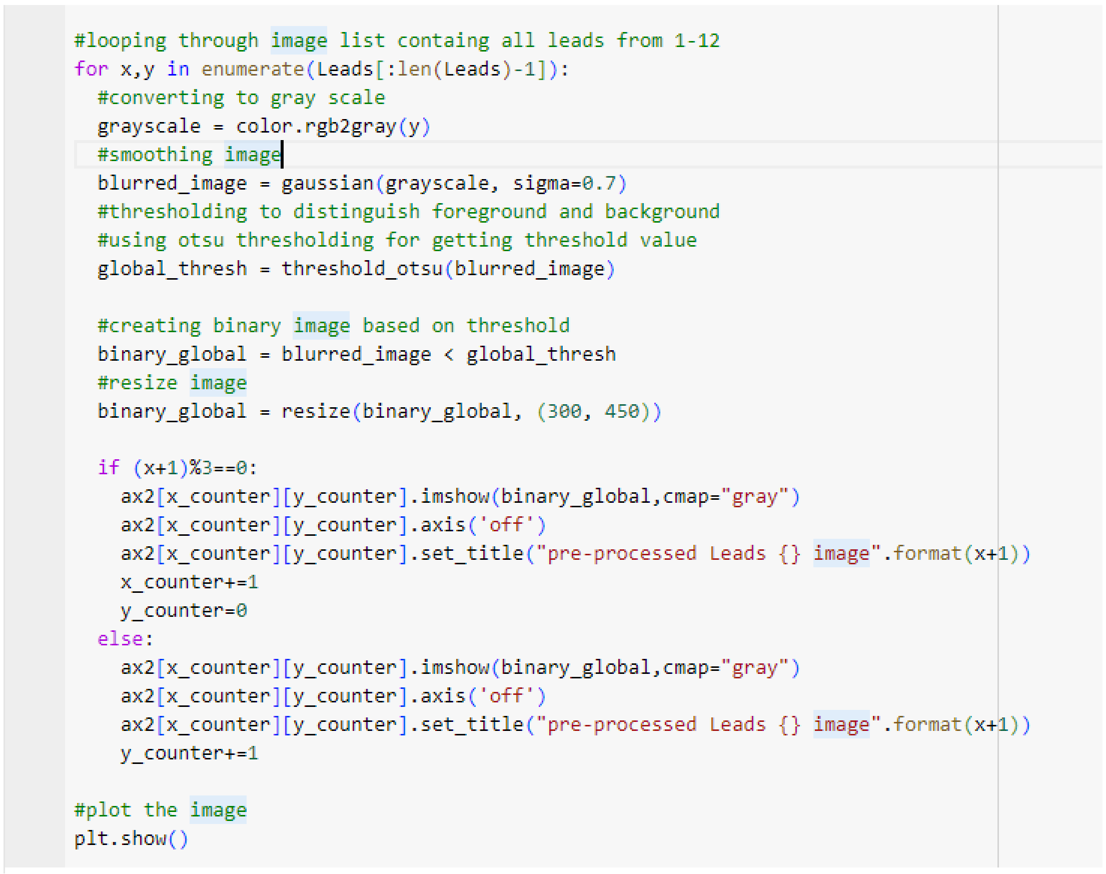
This code's goal is to show the ECG signal's thirteenth lead for easier comprehension and study. Using the code previously supplied, the 13th lead is extracted from the original ECG picture and visualized using the matplotlib library's imshow function. To emphasize the signal itself, the subplot's axis is switched off and its label is changed to "Leads 13" for clarity.



```
#importing gaussian filter and otsu threshold
from skimage.filters import threshold_otsu,gaussian
from skimage.transform import resize
from numpy import asarray

#creating subplot of size(4,3) 4 rows and 3 columns
fig2 , ax2 = plt.subplots(4,3)

fig2.set_size_inches(20, 20)

#setting counter for plotting based on value
x_counter=0
y_counter=0
```

Figure 18: Setting counter for plotting based on value.

In order to apply a Gaussian filter and Otsu thresholding to the ECG data, this code installs the required libraries. It then constructs a subplot with a size of (4,3) to visualize the findings. To monitor where the subplots are located in the grid, the x_counter and y_counter variables are initialized to 0.

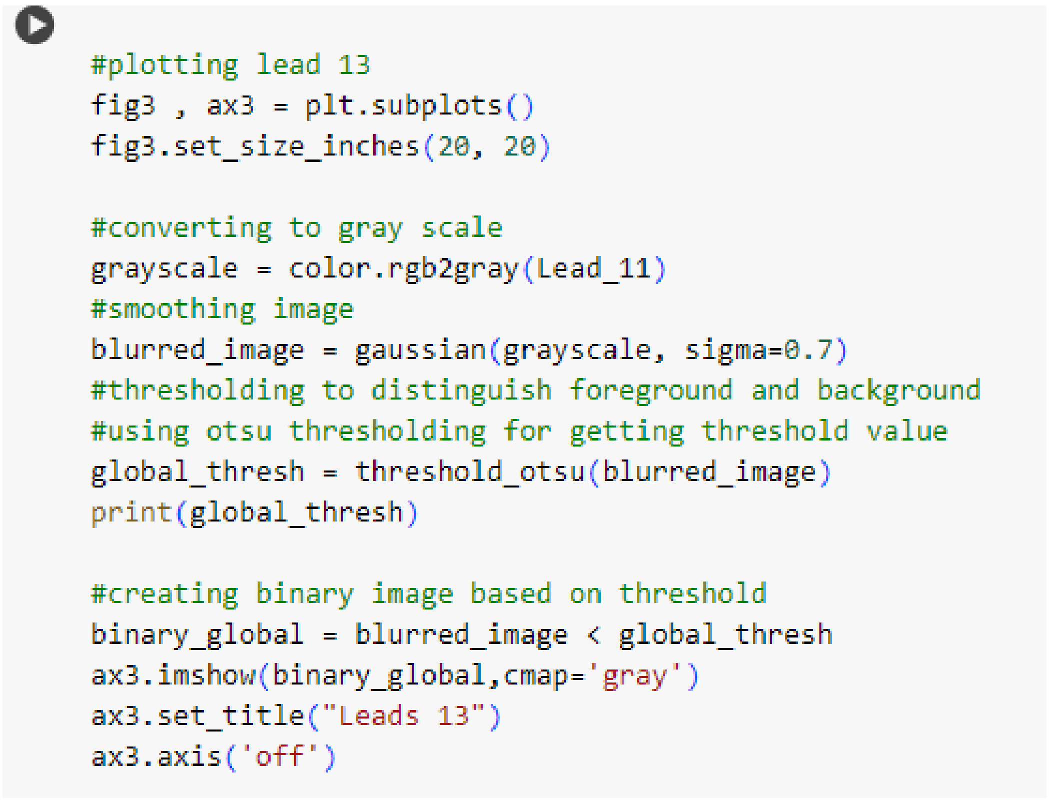


```
#looping through image list containg all leads from 1-12
for x,y in enumerate(Leads[:len(Leads)-1]):
  #converting to gray scale
  grayscale = color.rgb2gray(y)
  #smoothing image
  blurred_image = gaussian(grayscale, sigma=0.7)
  #thresholding to distinguish foreground and background
  #using otsu thresholding for getting threshold value
  global_thresh = threshold_otsu(blurred_image)

  #creating binary image based on threshold
  binary_global = blurred_image < global_thresh
  #resize image
  binary_global = resize(binary_global, (300, 450))

  if (x+1)%3==0:
    ax2[x_counter][y_counter].imshow(binary_global,cmap="gray")
    ax2[x_counter][y_counter].axis('off')
    ax2[x_counter][y_counter].set_title("pre-processed Leads {} image".format(x+1))
    x_counter+=1
    y_counter=0
  else:
    ax2[x_counter][y_counter].imshow(binary_global,cmap="gray")
    ax2[x_counter][y_counter].axis('off')
    ax2[x_counter][y_counter].set_title("pre-processed Leads {} image".format(x+1))
    y_counter+=1

#plot the image
plt.show()
```

Figure 19: Looping through image list contain all leads from 1-12

The ECG signals in every lead of the picture are pre-processed by this code. Pre-processing involves converting the RGB picture to grayscale, using a Gaussian filter to smooth the image, and utilizing Otsu thresholding to separate the ECG signal in the foreground from the background. After that, the binary picture is scaled to the industry standard of 300 by 450 pixels.

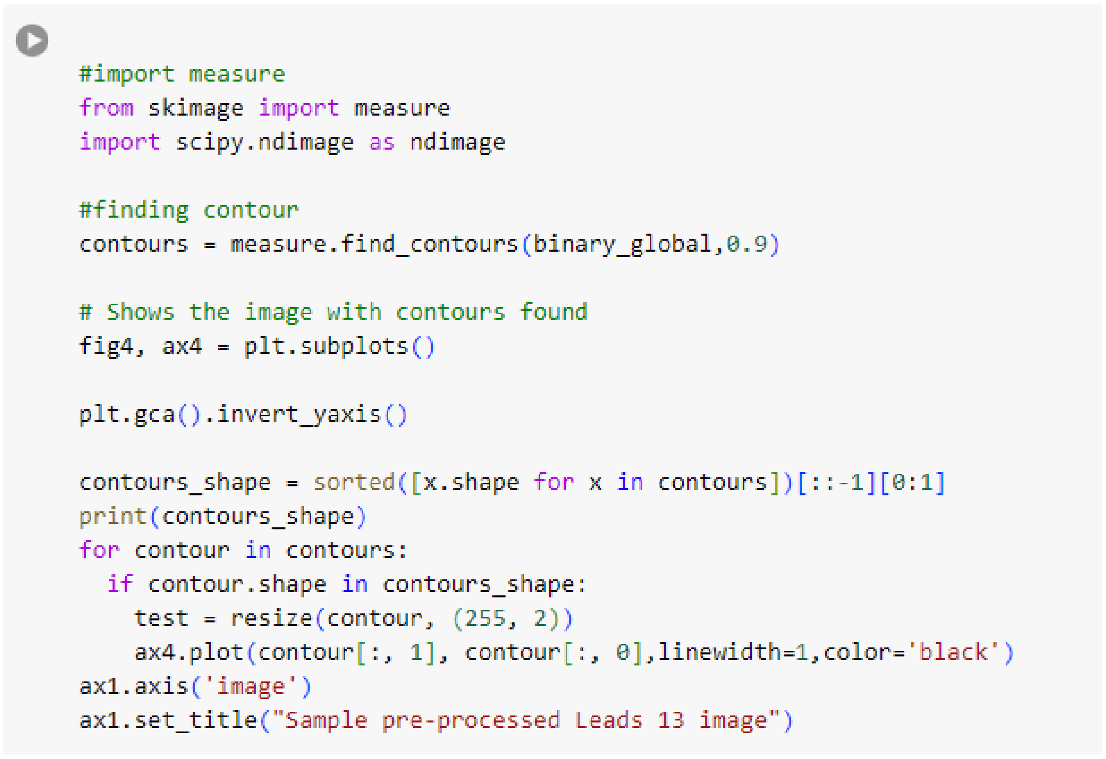
The pre-processing processes are applied to each lead in the Leads list by the code, which then loops over each lead and shows the binary picture in a (4,3) subplot. The subplots' locations inside the grid are tracked by the x_counter and y_counter variables.



```
#plotting lead 13
fig3 , ax3 = plt.subplots()
fig3.set_size_inches(20, 20)

#converting to gray scale
grayscale = color.rgb2gray(Lead_11)
#smoothing image
blurred_image = gaussian(grayscale, sigma=0.7)
#thresholding to distinguish foreground and background
#using otsu thresholding for getting threshold value
global_thresh = threshold_otsu(blurred_image)
print(global_thresh)

#creating binary image based on threshold
binary_global = blurred_image < global_thresh
ax3.imshow(binary_global,cmap='gray')
ax3.set_title("Leads 13")
ax3.axis('off')
```

Figure 20: Converting to gray scale.

The 13th lead of the ECG signal, which was taken from the original ECG picture using the previously given code, is subjected to pre-processing processes by this code. Pre-processing involves converting the RGB picture to grayscale, using a Gaussian filter to smooth the image, and utilizing Otsu thresholding to separate the ECG signal in the foreground from the background. Next, a subplot displaying the binary picture is shown.



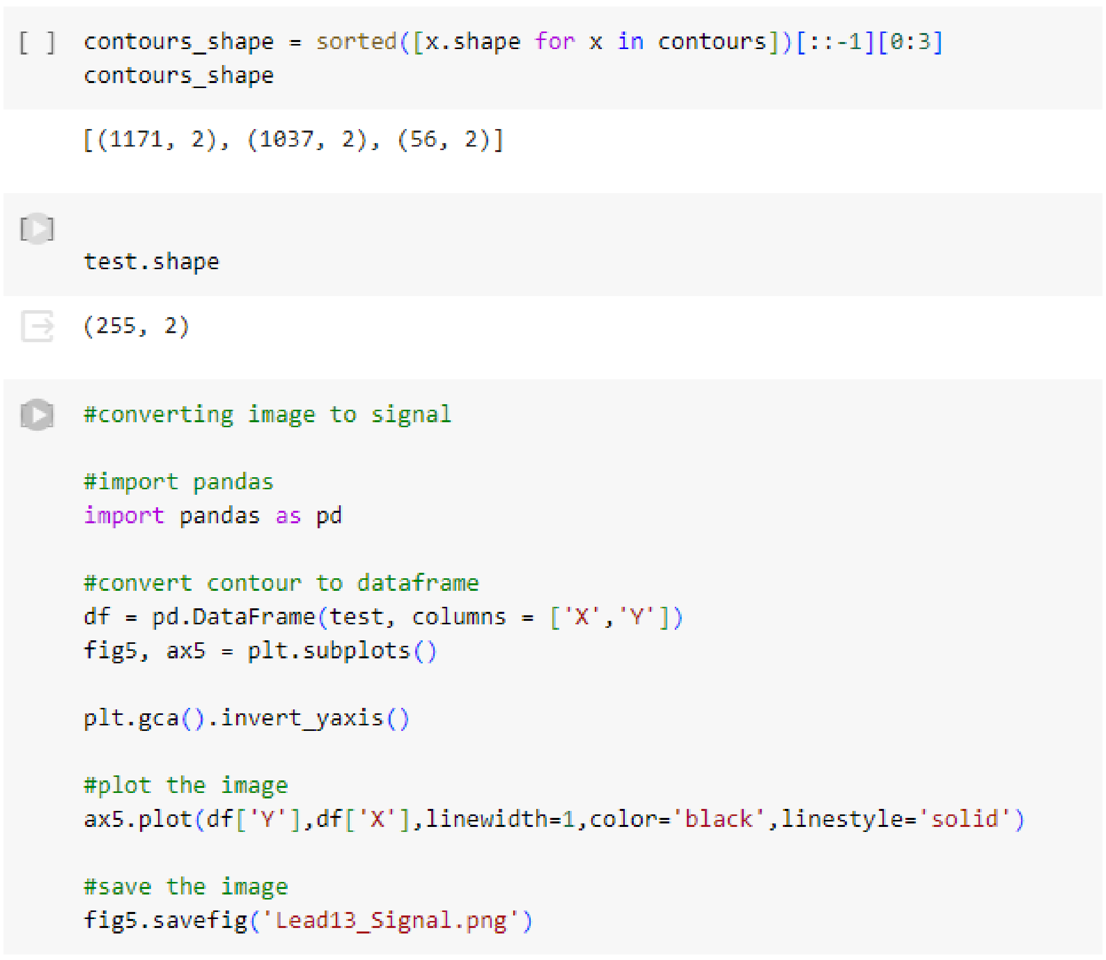
```
#import measure
from skimage import measure
import scipy.ndimage as ndimage

#finding contour
contours = measure.find_contours(binary_global,0.9)

# Shows the image with contours found
fig4, ax4 = plt.subplots()

plt.gca().invert_yaxis()

contours_shape = sorted([x.shape for x in contours])[::-1][0:1]
print(contours_shape)
for contour in contours:
  if contour.shape in contours_shape:
    test = resize(contour, (255, 2))
    ax4.plot(contour[:, 1], contour[:, 0],linewidth=1,color='black')
ax1.axis('image')
ax1.set_title("Sample pre-processed Leads 13 image")
```

Figure 21: shows the image with contours found.

Finding and displaying the contours of the pre-processed 13th lead of the ECG data is the aim of this code. The contours of the binary picture are located using the find_contours function, and the contours are then shown on the image using the plot function. The plot's y-axis is inverted using the invert_yaxis function to show the contours in an upright position. The contours are resized to a standard size of 255x2 pixels using the resize function. The contours' shape is saved in the contours_shape variable, and it is printed to the console using the print function. Plot's axis may be modified to 'image' using the axis function, and its title can be changed to "Sample pre-processed Leads 13 image" using the set_title function.

This code locates and shows a binary image's contours. The following summarizes the functions of each section of the code:

1. Bring in the required libraries:
   - The image's contours may be found using skimage.measure.
   - To process images, scipy.ndimage is utilized.
   - matplotlib. The picture and contours are plotted and shown using pyplot.
2. Using a level of 0.9, locate contours in the binary picture binary_global.
3. Matplotlib.pyplot is used to display the image with the contours that were obtained. Choose the biggest contour by sorting the others according to shape. Plot the contours you've chosen on the picture.



```
contours_shape = sorted([x.shape for x in contours])[::-1][0:3]
contours_shape
```

```
[(1171, 2), (1037, 2), (56, 2)]
```

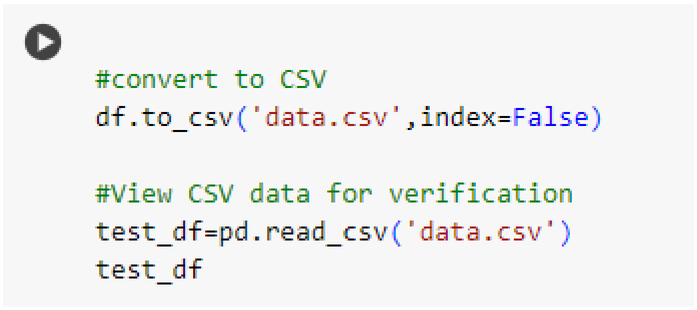
```
test.shape
```

```
(255, 2)
```

```
#converting image to signal

#import pandas
import pandas as pd

#convert contour to dataframe
df = pd.DataFrame(test, columns = ['X','Y'])
fig5, ax5 = plt.subplots()

plt.gca().invert_yaxis()

#plot the image
ax5.plot(df['Y'],df['X'],linewidth=1,color='black',linestyle='solid')

#save the image
fig5.savefig('Lead13_Signal.png')
```

Figure 22: Converting image to signal.

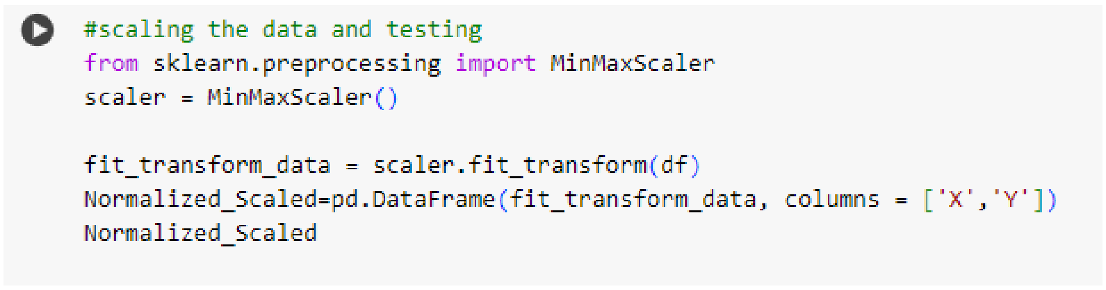
Before storing the contour data—which represents an ECG signal—as a picture, this function transforms it into a graphical representation. The contour data is organized into a Data Frame with 'X' and 'Y' columns using pandas. Next, using Matplotlib, a plot is made, with the Y-axis inverted to correctly show the signal. The data is shown with the Y values on the X-axis and X values on the Y-axis, using a solid black line of width 1. At last, the plot is effectively shown using the coordinates that were supplied, and it is saved as an image file called 'Lead13_Signal.png'.



```
#convert to CSV
df.to_csv('data.csv',index=False)

#View CSV data for verification
test_df=pd.read_csv('data.csv')
test_df
```

Figure 23: convert to CSV.

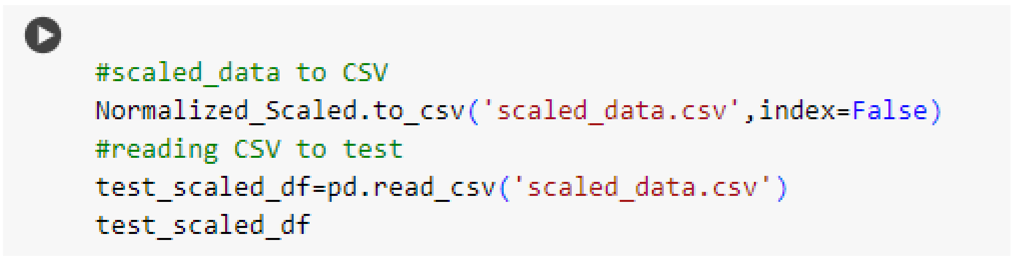
With the 'X' and 'Y' columns of contour data, this code writes the DataFrame df to a CSV file called data.csv, omitting the index. To confirm that the data stored is accurate, it then reads the CSV file again into a fresh DataFrame called test_df. It is possible to visually verify that the data has been correctly stored and retrieved by displaying test_df.



```
#scaling the data and testing
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

fit_transform_data = scaler.fit_transform(df)
Normalized_Scaled=pd.DataFrame(fit_transform_data, columns = ['X','Y'])
Normalized_Scaled
```

Figure 24: scaling the data and testing.

Using the MinMaxScaler from scikit-learn, this code scales the 'X' and 'Y' values in the DataFrame df to the range [0, 1]. An array of scaled values is produced once the scaler is initially fitted to the data and transformed. These values are then kept in the original column names, "X" and "Y," in a new DataFrame called Normalized_Scaled. After that, the scaled DataFrame is shown for confirmation. To improve model performance, data must be standardized before machine learning algorithms are used. This is where the normalization process comes in.

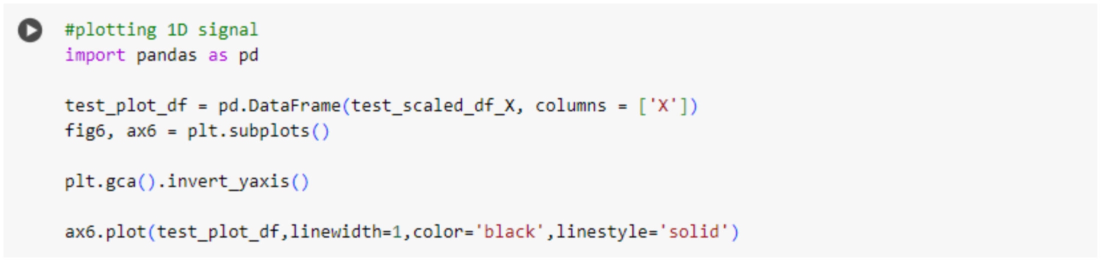


```
#scaled_data to CSV
Normalized_Scaled.to_csv('scaled_data.csv',index=False)
#reading CSV to test
test_scaled_df=pd.read_csv('scaled_data.csv')
test_scaled_df
```

Figure 25: Scaled test data to csv.

Without omitting the index, this code saves the normalized DataFrame Normalized_Scaled to a CSV file called scaled_data.csv. To confirm that the data has been successfully stored and can be reliably retrieved, it then reads the CSV file back into a new DataFrame called test_scaled_df. The scaled data may be visually verified by displaying test_scaled_df. This procedure makes sure that the data has been normalized and is ready to be loaded and saved.

With the index omitted, this code saves the 'X' column of the normalized DataFrame Normalized_Scaled to a different CSV file called scaled_data_X.csv. After that, it reads the CSV file again into a fresh DataFrame called test_scaled_df_X to confirm that the data was stored accurately and to examine the DataFrame's form. By doing this, it is made sure that the X-axis data—which represents the signal's high and low points—is precisely saved and retrievable for additional examination.



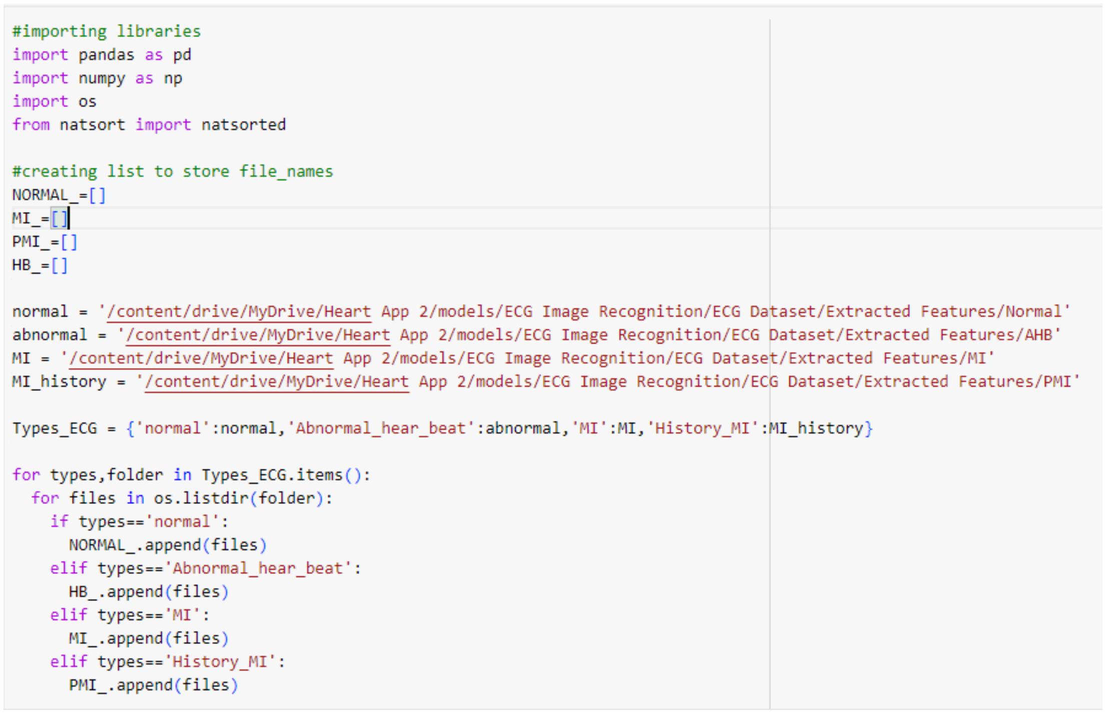
```
#plotting 1D signal
import pandas as pd

test_plot_df = pd.DataFrame(test_scaled_df_X, columns = ['X'])
fig6, ax6 = plt.subplots()

plt.gca().invert_yaxis()

ax6.plot(test_plot_df,linewidth=1,color='black',linestyle='solid')
```

Figure 26: plotting 1D signal.

Using the 'X' column from the test_scaled_df_X DataFrame, this code shows a 1D signal. To begin, import pandas and create a DataFrame test_plot_df from test_scaled_df_X, making sure that the column bearing the name 'X' is called thus. After that, a fresh plot is made with the Y-axis inverted to show the signal correctly. The values of 'X' are represented by a black, solid line with a width of 1. By emphasizing the high and low points of the data, which correlate to the signal's salient characteristics, this technique visualizes the 1D signal.

The Convert_Image_Lead function, defined in this code, extracts, and saves individual lead pictures from an ECG image. The program separates the image into 13 segments, each of which corresponds to an ECG lead, after reading the input image file using the matplotlib plugin's imread function. Based on predetermined pixel coordinates, these segments are removed to provide unique pictures for every lead. The '.jpg' extension is eliminated from the picture file name to create a folder name, and the extracted leads are kept in a list called Leads. The function's next section creates and saves unique pictures for each lead by iterating over the Leads list. A fresh directory called after the input picture file (without the '.jpg' extension) is created, and the image is saved there.

For every lead, a new plot is made and shown using Matplotlib without axes. The directory is created if it doesn't already exist. Where {x} is the lead number, the photos are stored in the format 'Lead_{x}_Signal.png'. The method extract_signal_leads is called with the list of leads, the parent folder, and the folder name as parameters after all lead photos have been saved, most likely to process the extracted lead images further.



```
#importing libraries
import pandas as pd
import numpy as np
import os
from natsort import natsorted

#creating list to store file_names
NORMAL_=[]
MI_=[]
PMI_=[]
HB_=[]

normal = '/content/drive/MyDrive/Heart App 2/models/ECG Image Recognition/ECG Dataset/Extracted Features/Normal'
abnormal = '/content/drive/MyDrive/Heart App 2/models/ECG Image Recognition/ECG Dataset/Extracted Features/AHB'
MI = '/content/drive/MyDrive/Heart App 2/models/ECG Image Recognition/ECG Dataset/Extracted Features/MI'
MI_history = '/content/drive/MyDrive/Heart App 2/models/ECG Image Recognition/ECG Dataset/Extracted Features/PMI'

Types_ECG = {'normal':normal,'Abnormal_hear_beat':abnormal,'MI':MI,'History_MI':MI_history}

for types,folder in Types_ECG.items():
  for files in os.listdir(folder):
    if types=='normal':
      NORMAL_.append(files)
    elif types=='Abnormal_hear_beat':
      HB_.append(files)
    elif types=='MI':
      MI_.append(files)
    elif types=='History_MI':
      PMI_.append(files)
```

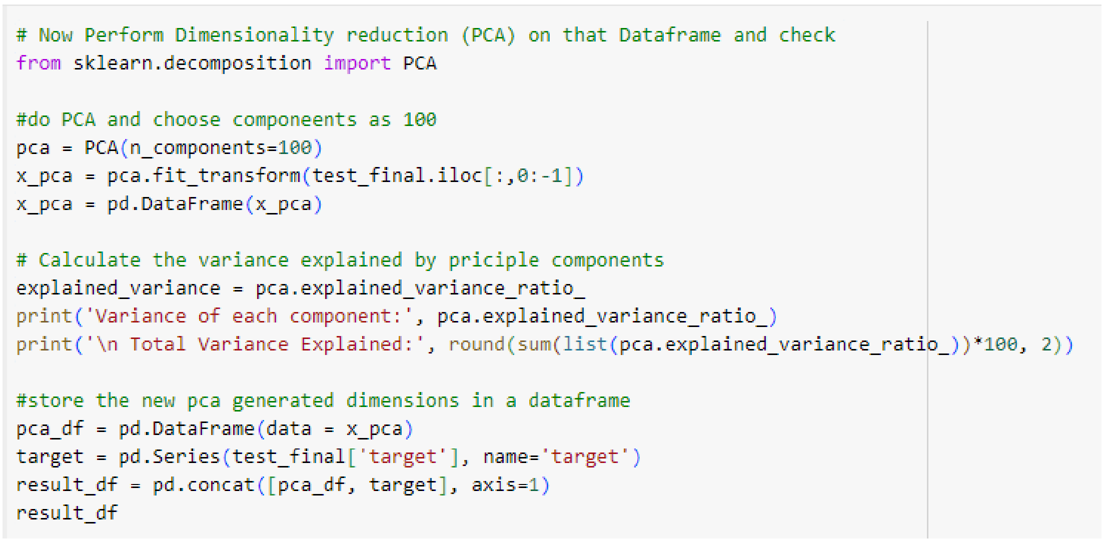
Figure 27: organizes and categorizes the filenames of ECG data into different lists based on their types.

The function's next section creates and saves unique pictures for each lead by iterating over the Leads list. A fresh directory called after the input picture file (without the '.jpg' extension) is created, and the image is saved there. For every lead, a new plot is made and shown using Matplotlib without axes. The directory is created if it doesn't already exist. Where {x} is the lead number, the photos are stored in the format 'Lead_{x}_Signal.png'. The method extract_signal_leads is called with the list of leads, the parent folder, and the folder name as parameters after all lead photos have been saved, most likely to process the extracted lead images further.

The Types_ECG dictionary's key-value pairs are iterated over by nested loops in the code. It loops over the files in the appropriate location for every kind of ECG data. It appends the filename to the relevant list: NORMAL, HB, MI, or PMI, depending on the kind of ECG data being analyzed ('normal', 'Abnormal_heart_beat', 'MI', or 'History_MI').

The lists NORMAL_ and HB_, which hold the filenames for normal and abnormal ECG data, respectively, are sorted first in this code segment. Next, it reads the associated CSV files for each type of ECG data—normal, abnormal heartbeat, MI, and prior MI—by iterating over the length of the MI_ list (if all lists have the same length). Using the pd.concat() method, the data from these CSV files is concatenated along the rows to create a composite DataFrame final_df. The resultant combined DataFrame is then stored as 'Combined_IDLead_{x}.csv', where {x} is the iteration index plus one.

The combined CSV file "Combined_IDLead_1.csv," which is likely to include data for lead 1, is read by the second section of the code. Next, it verifies that the values in the 'Target' column are unique and most likely correspond to the class labels. If the 'Unnamed: 0' column includes redundant index information, it is removed from the DataFrame. The ngroup() function is utilized to numerically encode the 'Target' column, allocating a distinct number label to every individual target value. The 'target' column is a new column that holds these encoded labels. Ultimately, the DataFrame test_final is produced once the 'Target' column is removed.

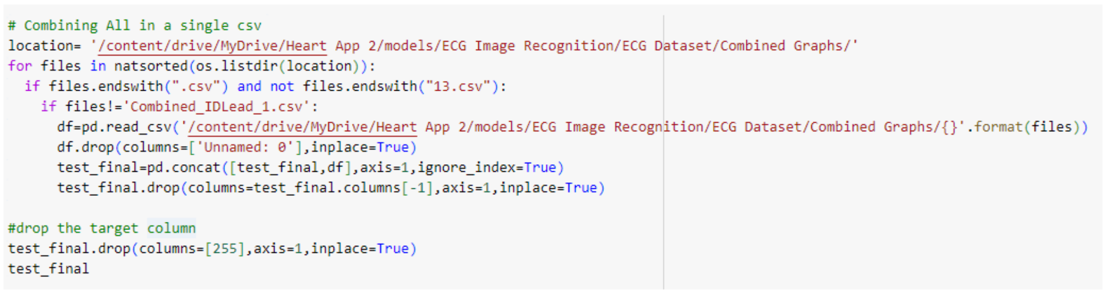


```
# Now Perform Dimensionality reduction (PCA) on that Dataframe and check
from sklearn.decomposition import PCA

#do PCA and choose componeents as 100
pca = PCA(n_components=100)
x_pca = pca.fit_transform(test_final.iloc[:,0:-1])
x_pca = pd.DataFrame(x_pca)

# Calculate the variance explained by priciple components
explained_variance = pca.explained_variance_ratio_
print('Variance of each component:', pca.explained_variance_ratio_)
print('\n Total Variance Explained:', round(sum(list(pca.explained_variance_ratio_))*100, 2))

#store the new pca generated dimensions in a dataframe
pca_df = pd.DataFrame(data = x_pca)
target = pd.Series(test_final['target'], name='target')
result_df = pd.concat([pca_df, target], axis=1)
result_df
```

Figure 28: this code performs PCA to reduce the dimensionality of the input data while preserving as much variance as possible.

This code section uses Principal Component Analysis (PCA) to reduce the dimensionality of the input DataFrame test_final. First, PCA is imported using the scikit-learn package. Subsequently, PCA is instantiated with 100 components. The input DataFrame is transformed into the lower-dimensional space specified by the first 100 main components using the fit_transform method. Next, a new DataFrame called x_pca contains the altered data. Next, using the explained_variance_ratio_ property of the PCA object, the variance explained by each principal component is determined. This sheds light on the proportion of each primary component's explanation of the overall variance in the original data. Each component's variance is presented together with the overall variation that each of the 100 components can explain.

The modified PCA dimensions are then saved in a DataFrame called pca_df, and the target column—which is likely to reflect class labels—is taken out of test_final and put into a Series called target. The adjusted PCA dimensions and the target labels are combined in a new DataFrame result_df by concatenating these two data structures along the columns axis.

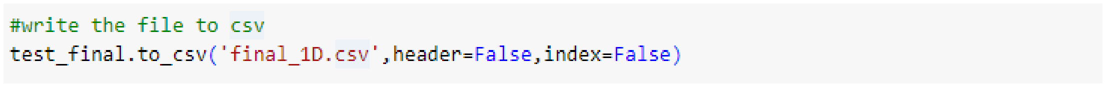


```
# Combining All in a single csv
location= '/content/drive/MyDrive/Heart App 2/models/ECG Image Recognition/ECG Dataset/Combined Graphs/'
for files in natsorted(os.listdir(location)):
  if files.endswith(".csv") and not files.endswith("13.csv"):
    if files!='Combined_IDLead_1.csv':
      df=pd.read_csv('/content/drive/MyDrive/Heart App 2/models/ECG Image Recognition/ECG Dataset/Combined Graphs/{}'.format(files))
      df.drop(columns=['Unnamed: 0'],inplace=True)
      test_final=pd.concat([test_final,df],axis=1,ignore_index=True)
      test_final.drop(columns=test_final.columns[-1],axis=1,inplace=True)

#drop the target column
test_final.drop(columns=[255],axis=1,inplace=True)
test_final
```

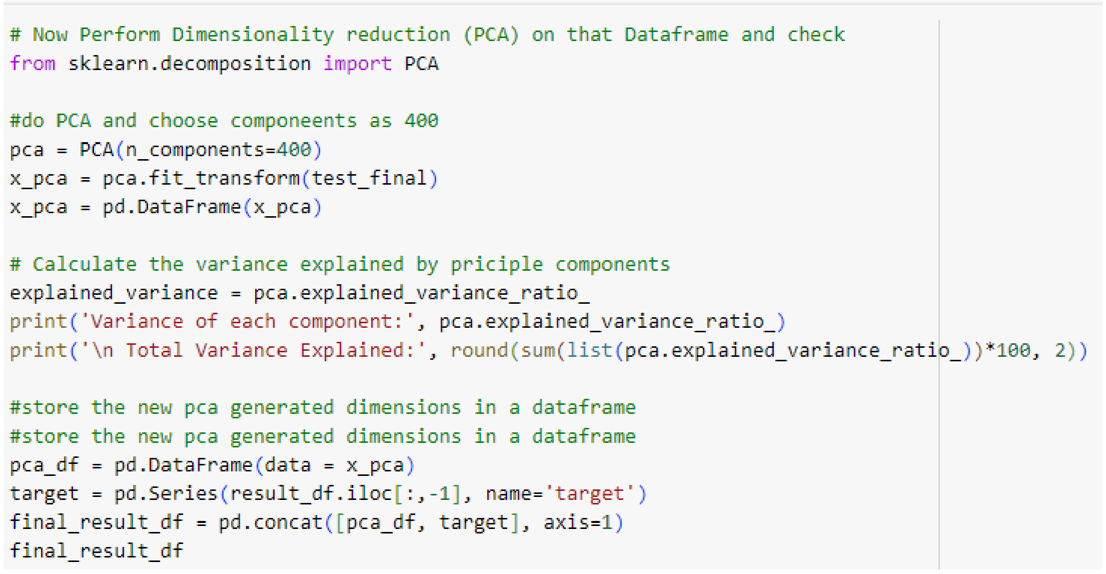
Figure 29: combining all in a single csv.

The goal of this code snippet is to create a single CSV file from many CSV files that include ECG data. With os.listdir(), it iterates over the files in the designated directory (location), verifying that each file ends in ".csv" and not "13.csv". It also makes sure that the file isn't the original "Combined_IDLead_1.csv" CSV file. The function reads the CSV file into a DataFrame (df) if these requirements are satisfied. The 'Unnamed: 0' column is then removed, presumably since it does not contain any redundant index data.

The data is then effectively combined by concatenating the columns of the current DataFrame (df) with the old DataFrame test_final along the column's axis. By using the ignore_index=True argument, concatenation prevents duplicate column indices. The last column of test_final is then removed to prevent duplication because it is inserted once again after each loop iteration.The target column is finally removed from test_final, presumably because it was inserted earlier in the code for additional processing, like PCA. All the CSV files' combined data may be found in the resultant DataFrame test_final.



```
#write the file to csv
test_final.to_csv('final_1D.csv',header=False,index=False)
```



```
# Now Perform Dimensionality reduction (PCA) on that Dataframe and check
from sklearn.decomposition import PCA

#do PCA and choose componeents as 400
pca = PCA(n_components=400)
x_pca = pca.fit_transform(test_final)
x_pca = pd.DataFrame(x_pca)

# Calculate the variance explained by priciple components
explained_variance = pca.explained_variance_ratio_
print('Variance of each component:', pca.explained_variance_ratio_)
print('\n Total Variance Explained:', round(sum(list(pca.explained_variance_ratio_))*100, 2))

#store the new pca generated dimensions in a dataframe
#store the new pca generated dimensions in a dataframe
pca_df = pd.DataFrame(data = x_pca)
target = pd.Series(result_df.iloc[:,-1], name='target')
final_result_df = pd.concat([pca_df, target], axis=1)
final_result_df
```

Figure 30: store the new pca generated dimensions in a data frame.

The DataFrame test_final is initially written in this code segment to a CSV file called "final_1D.csv" with the parameters header=False and index=False, indicating that the output file will not contain column names or row indices.Next, it applies Principal Component Analysis (PCA) to the test_final DataFrame to reduce dimensionality.

It instantiates PCA with 400 components after importing it from scikit-learn. The data is then transformed into a lower-dimensional space specified by the first 400 main components by using the fit_transform method on test_final. A new DataFrame called x_pca contains the altered data.

Next, using the explained_variance_ratio_ property of the PCA object, the variance explained by each principal component is determined. This sheds light on the proportion of each primary component's explanation of the overall variance in the original data. Each component's variance is presented together with the overall variation that each of the 400 components may explain. The target column is then taken from result_df (assuming result_df is the DataFrame from earlier code segments) and put in a Series called target.

The altered PCA dimensions are then saved in a DataFrame called pca_df. The altered PCA dimensions and the target labels are combined in a new DataFrame called final_result_df by concatenating these two data structures along the columns axis.

Testing Approaches

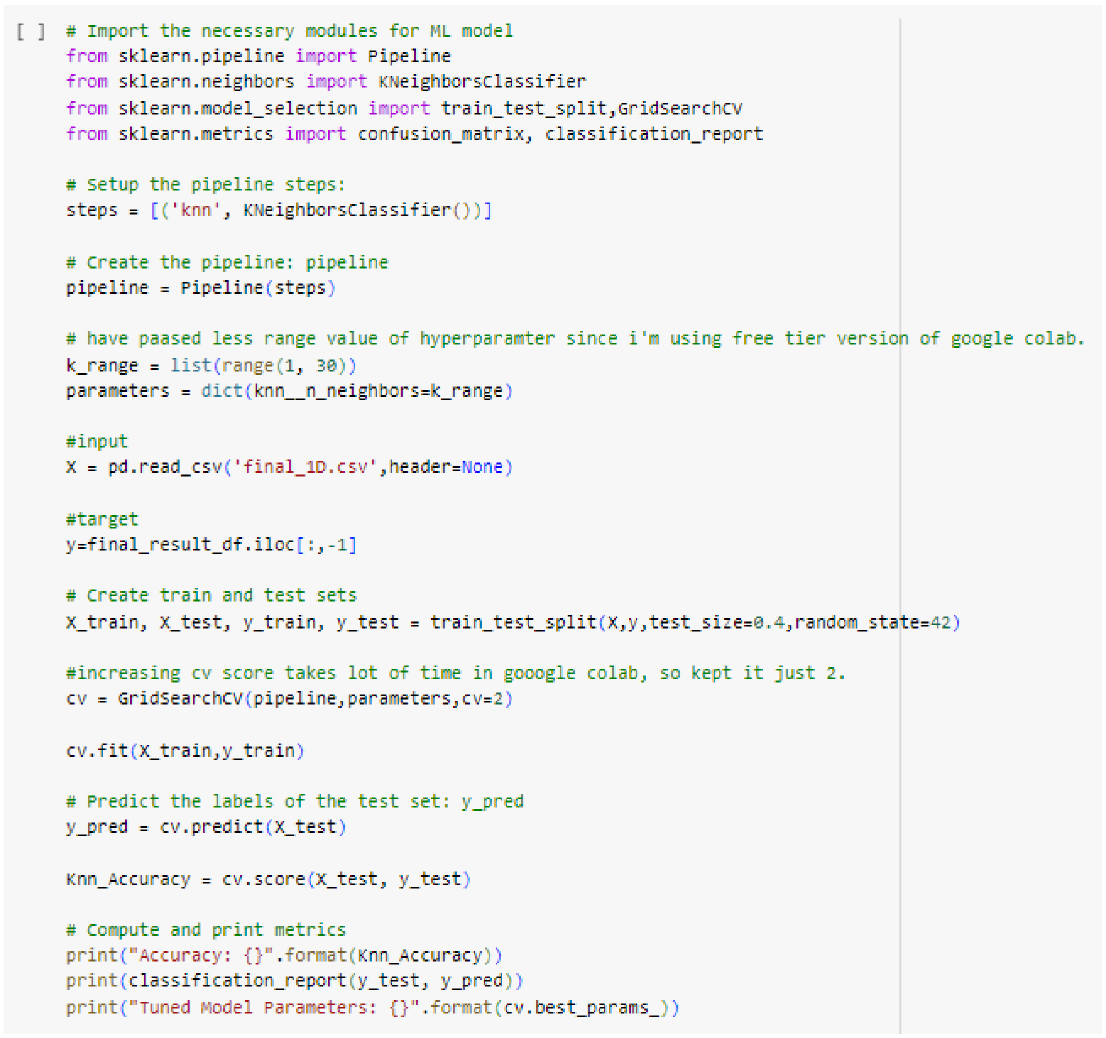
KNN Approaches

This method uses scikit-learn to set up a pipeline for a K-Nearest Neighbors (KNN) classifier. Pipeline, KNeighborsClassifier, train_test_split, GridSearchCV, confusion_matrix, and classification_report is among the modules that are imported.For the KNN classifier, the pipeline stages are defined with a single step.

After that, follow these steps to construct a pipeline object pipeline.A grid search across the KNN classifier's number of neighbors (k) is used to tune the hyperparameters. K has a specified range of values from 1 to 30.The target data, y, is taken from the final_result_df DataFrame, presumably containing the target labels, whereas the input data, X, is read from the CSV file "final_1D.csv," presuming it includes the features.

Using the train_test_split function, the dataset is divided into training and testing sets, with 40% of the data going toward testing and 60% going toward training. To determine the KNN classifier's ideal hyperparameters, GridSearchCV is used. The cross-validation parameter (cv) is set to 2, which lessens the computational burden at the risk of perhaps less accurate findings, because computation time is a problem (probably because of resource limits).

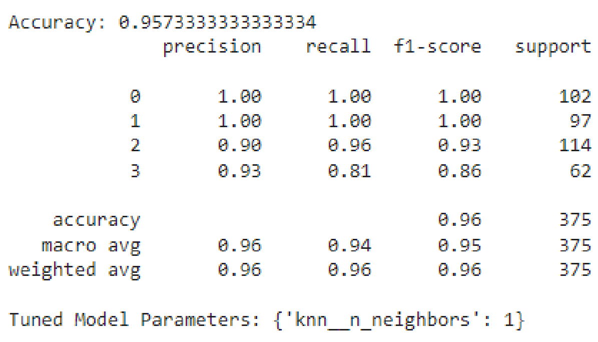
KNN APPROCH



```
[ ] # Import the necessary modules for ML model
    from sklearn.pipeline import Pipeline
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import train_test_split,GridSearchCV
    from sklearn.metrics import confusion_matrix, classification_report

    # Setup the pipeline steps:
    steps = [('knn', KNeighborsClassifier())]

    # Create the pipeline: pipeline
    pipeline = Pipeline(steps)

    # have paased less range value of hyperparamter since i'm using free tier version of google colab.
    k_range = list(range(1, 30))
    parameters = dict(knn__n_neighbors=k_range)

    #input
    X = pd.read_csv('final_1D.csv',header=None)

    #target
    y=final_result_df.iloc[:,-1]

    # Create train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4,random_state=42)

    #increasing cv score takes lot of time in gooogle colab, so kept it just 2.
    cv = GridSearchCV(pipeline,parameters,cv=2)

    cv.fit(X_train,y_train)

    # Predict the labels of the test set: y_pred
    y_pred = cv.predict(X_test)

    Knn_Accuracy = cv.score(X_test, y_test)

    # Compute and print metrics
    print("Accuracy: {}".format(Knn_Accuracy))
    print(classification_report(y_test, y_pred))
    print("Tuned Model Parameters: {}".format(cv.best_params_))
```

Figure 31: KNN Approach.

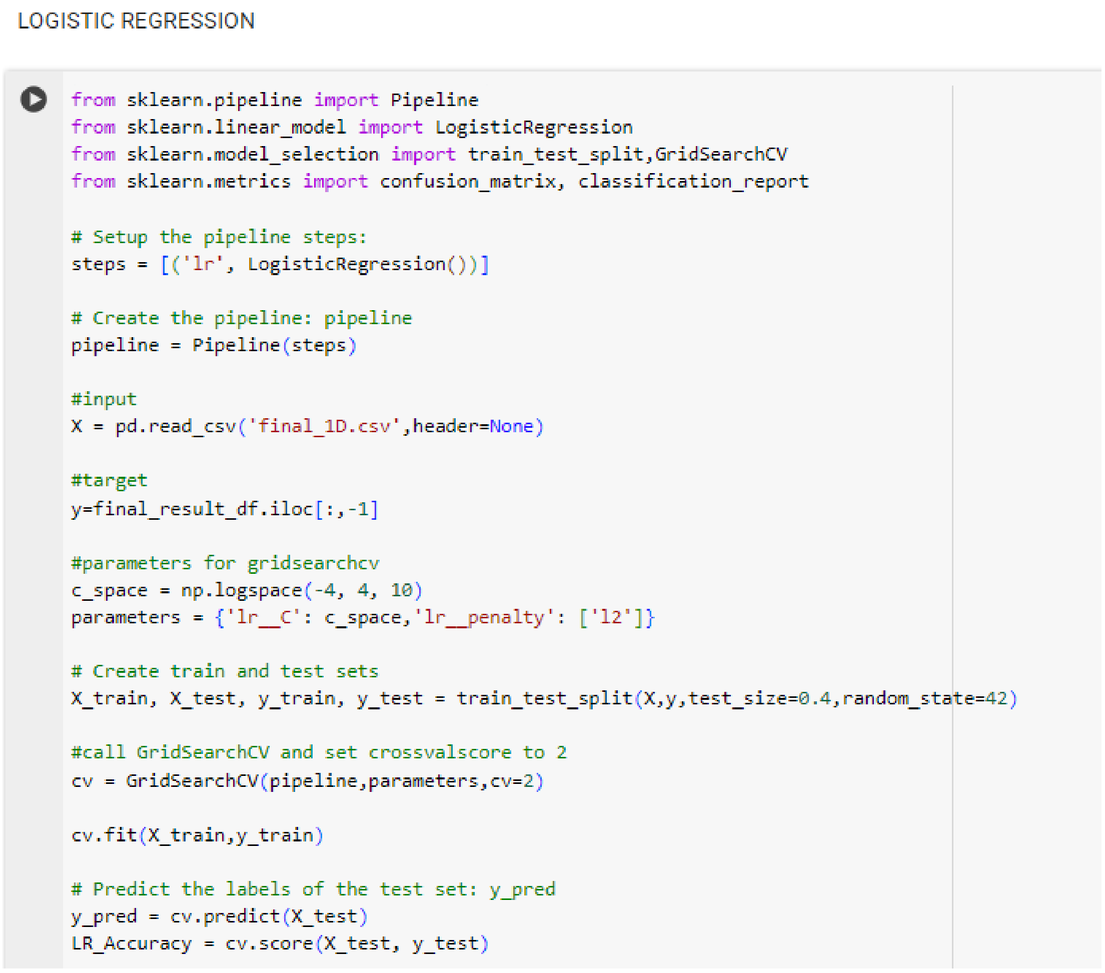
Predictions are made using the test data after the model has been fitted to the training set. Metrics like precision, recall, and F1-score for every class are included in the classification report, which is also presented together with the model's accuracy on the test set. The model's adjusted hyperparameters are also printed.



```
Accuracy: 0.9573333333333334
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       102
           1       1.00      1.00      1.00        97
           2       0.90      0.96      0.93       114
           3       0.93      0.81      0.86        62

    accuracy                           0.96       375
   macro avg       0.96      0.94      0.95       375
weighted avg       0.96      0.96      0.96       375

Tuned Model Parameters: {'knn__n_neighbors': 1}
```

Figure 32: Accuracy of KNN Approach.

Logistic Regression Approach



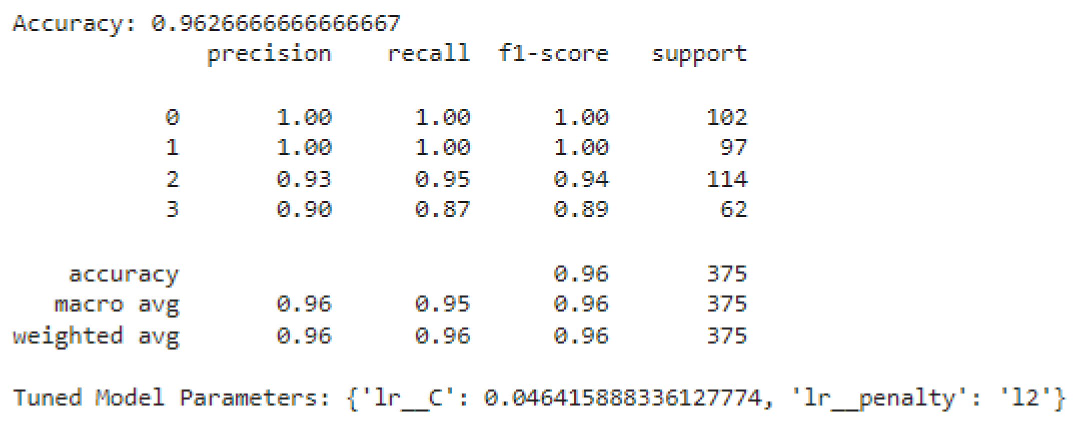
LOGISTIC REGRESSION

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report

# Setup the pipeline steps:
steps = [('lr', LogisticRegression())]

# Create the pipeline: pipeline
pipeline = Pipeline(steps)

#input
X = pd.read_csv('final_1D.csv',header=None)

#target
y=final_result_df.iloc[:,-1]

#parameters for gridsearchcv
c_space = np.logspace(-4, 4, 10)
parameters = {'lr__C': c_space,'lr__penalty': ['l2']}

# Create train and test sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4,random_state=42)

#call GridSearchCV and set crossvalscore to 2
cv = GridSearchCV(pipeline,parameters,cv=2)

cv.fit(X_train,y_train)

# Predict the labels of the test set: y_pred
y_pred = cv.predict(X_test)
LR_Accuracy = cv.score(X_test, y_test)
```

Figure 33: Logistic Regression Approach.

This section of code configures a scikit-learn pipeline for a logistic regression classifier. Pipeline, Logistic Regression, train_test_split, GridSearchCV, confusion_matrix, and classification_report is among the modules that are imported.For the logistic regression classifier, a single step defines the pipeline stages. After that, follow these steps to construct a pipeline object pipeline. Grid search over the regularization strength (C) for logistic regression and the penalty type (penalty) is used for hyperparameter tweaking. Logarithmic spacing is used to specify the range of values for C, which is 10^{-4} to 10^{4}, with 'l2' as the penalty.

The target data, y, is taken from the final_result_df DataFrame, presumably containing the target labels, whereas the input data, X, is read from the CSV file "final_1D.csv," presuming it includes the features.Using the train_test_split function, the dataset is divided into training and testing sets, with 40% of the data going toward testing and 60% going toward training. The logistic regression classifier's ideal hyperparameters are found using GridSearchCV. The cross-validation parameter (cv) is set to 2, which lessens the computational burden at the risk of perhaps less accurate findings, because computation time is a problem (probably because of resource limits).Predictions are made using the test data after the model has been fitted to the training set. The variable LR_Accuracy is used to calculate and record the model's accuracy on the test set. The accuracy is 0.962.



```
Accuracy: 0.9626666666666667
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       102
           1       1.00      1.00      1.00        97
           2       0.93      0.95      0.94       114
           3       0.90      0.87      0.89        62

    accuracy                           0.96       375
   macro avg       0.96      0.95      0.96       375
weighted avg       0.96      0.96      0.96       375

Tuned Model Parameters: {'lr__C': 0.046415888336127774, 'lr__penalty': 'l2'}
```

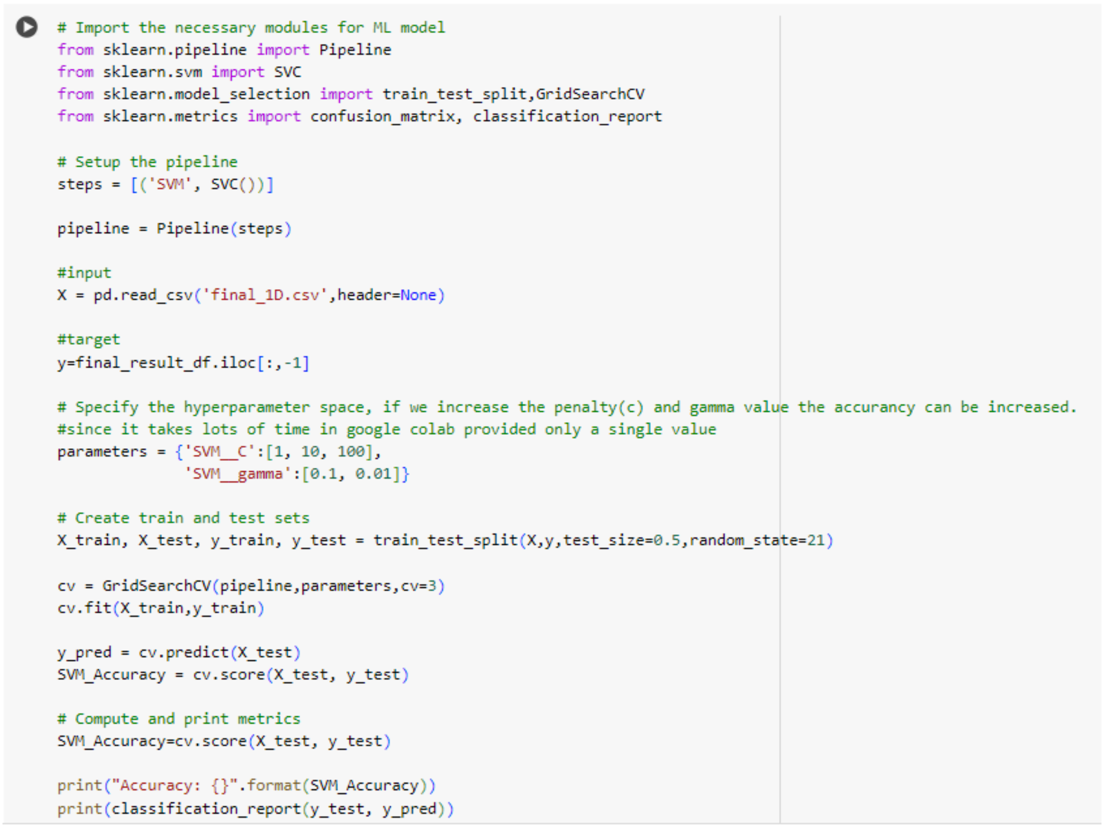
Figure 34: Accuracy of LR Approach.

Support Vector Machine Approach

This section of code configures a scikit-learn pipeline for a Support Vector Machine (SVM) classifier. Pipeline, SVC, train_test_split, GridSearchCV, confusion_matrix, and classification_report is among the modules that are imported.For the SVM classifier, a single step defines the pipeline stages. After that, follow these steps to construct a pipeline object pipeline. The target data, y, is taken from the final_result_df DataFrame, presumably containing the target labels, whereas the input data, X, is read from the CSV file "final_1D.csv," presuming it includes the features. Using the train_test_split function, the dataset is divided into training and testing sets, with 50% of the data designated for training and 50% for testing.

To get the best hyperparameters for the SVM classifier, GridSearchCV is used. The regularization parameter (C) and the kernel coefficient (gamma) are given distinct values in the hyperparameter space. Only a restricted set of hyperparameters is offered owing to computational restrictions (presumably resulting from limited resources in the Google Colab environment), which lessens the computational burden.

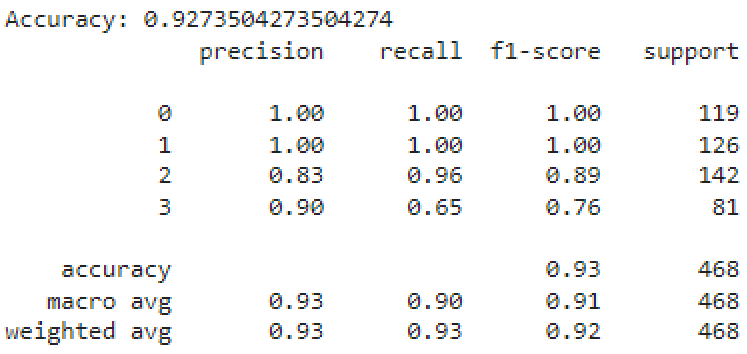
SVM



```
# Import the necessary modules for ML model
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report

# Setup the pipeline
steps = [('SVM', SVC())]

pipeline = Pipeline(steps)

#input
X = pd.read_csv('final_1D.csv',header=None)

#target
y=final_result_df.iloc[:,-1]

# Specify the hyperparameter space, if we increase the penalty(c) and gamma value the accurancy can be increased.
#since it takes lots of time in google colab provided only a single value
parameters = {'SVM__C':[1, 10, 100],
              'SVM__gamma':[0.1, 0.01]}

# Create train and test sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.5,random_state=21)

cv = GridSearchCV(pipeline,parameters,cv=3)
cv.fit(X_train,y_train)

y_pred = cv.predict(X_test)
SVM_Accuracy = cv.score(X_test, y_test)

# Compute and print metrics
SVM_Accuracy=cv.score(X_test, y_test)

print("Accuracy: {}".format(SVM_Accuracy))
print(classification_report(y_test, y_pred))
```

Figure 35: SVM Approach.

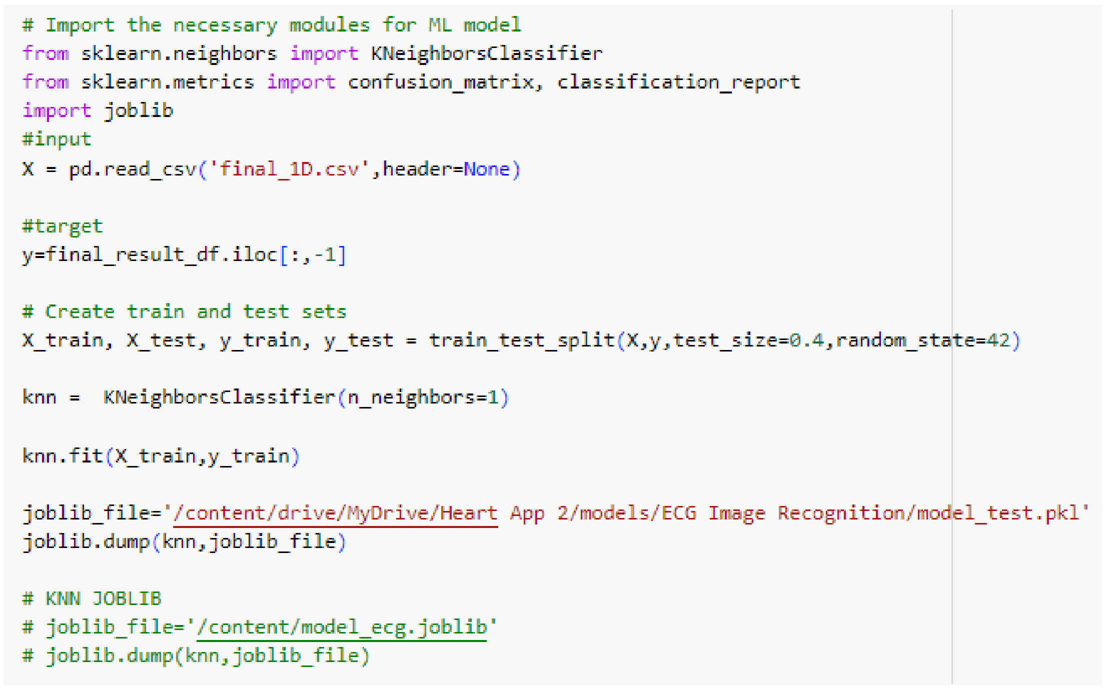
Predictions are made using the test data after the model has been fitted to the training set. The variable SVM_Accuracy is used to calculate and record the model's accuracy on the test set. The classification report, which contains metrics for each class such as F1-score, recall, and accuracy, is also printed.



```
Accuracy: 0.9273504273504274
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       119
           1       1.00      1.00      1.00       126
           2       0.83      0.96      0.89       142
           3       0.90      0.65      0.76        81

    accuracy                           0.93       468
   macro avg       0.93      0.90      0.91       468
weighted avg       0.93      0.93      0.92       468
```

Figure 36: Accuracy of SVM Approach.

This code section uses the input attributes X and the target labels y to train a K-Nearest Neighbors (KNN) classifier. It begins by reading the input features from the CSV file "final_1D.csv" into DataFrame X. If the final_result_df DataFrame has the target labels, it then extracts those labels from it. The train_test_split function is then used to divide the dataset into training and testing sets, allocating 40% of the data for testing and 60% for training.

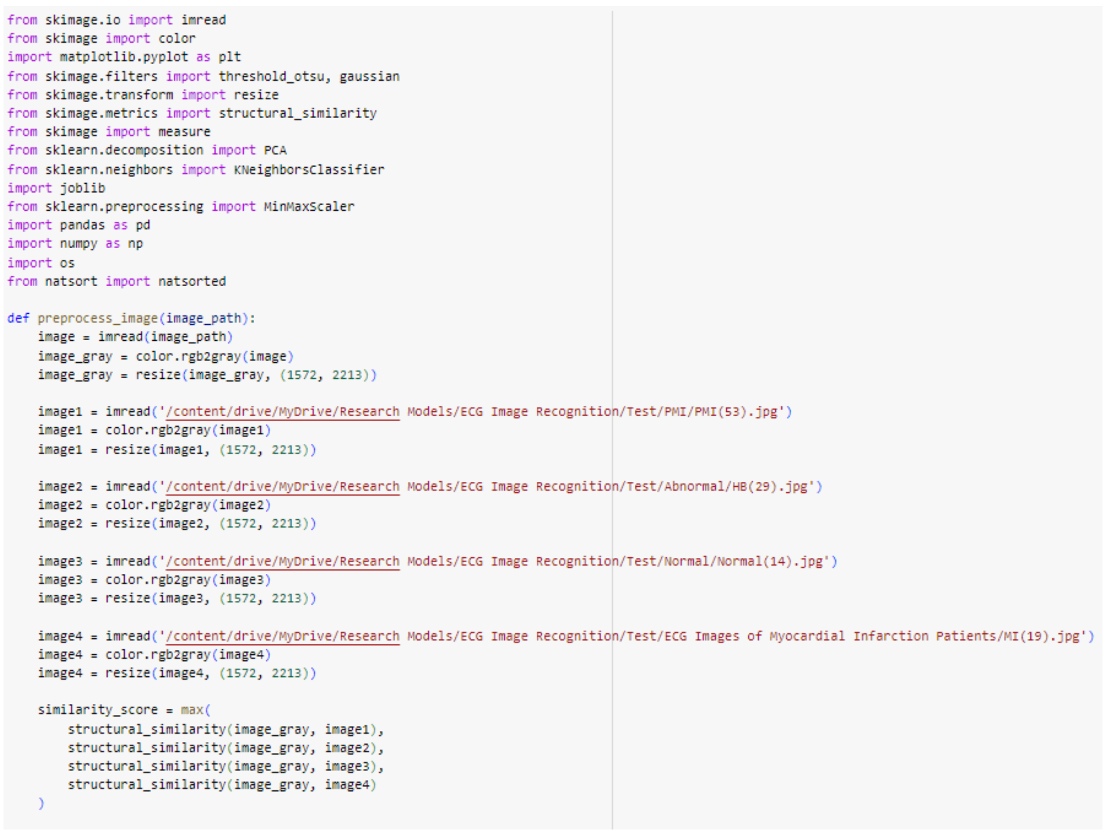


```python
# Import the necessary modules for ML model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report
import joblib
#input
X = pd.read_csv('final_1D.csv',header=None)

#target
y=final_result_df.iloc[:,-1]

# Create train and test sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4,random_state=42)

knn =  KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train,y_train)

joblib_file='/content/drive/MyDrive/Heart App 2/models/ECG Image Recognition/model_test.pkl'
joblib.dump(knn,joblib_file)

# KNN JOBLIB
# joblib_file='/content/model_ecg.joblib'
# joblib.dump(knn,joblib_file)
```

Figure 37: Import the KNN model for testing.

The first neighbor of a KNN classifier is one (n_neighbors=1). The fit approach is then used to train this classifier on the training set of data. After training, the model is stored to a file called "model_test.pkl" in the directory "/content/drive/MyDrive/Heart App 2/models/ECG Image Recognition/" after being serialized using the joblib.dump function. The model is transformed through the process of serialization into a byte stream that may be stored on disk and then deserialized to restore the original model. This eliminates the need to completely retrain the taught model to utilize it again later.

Testing

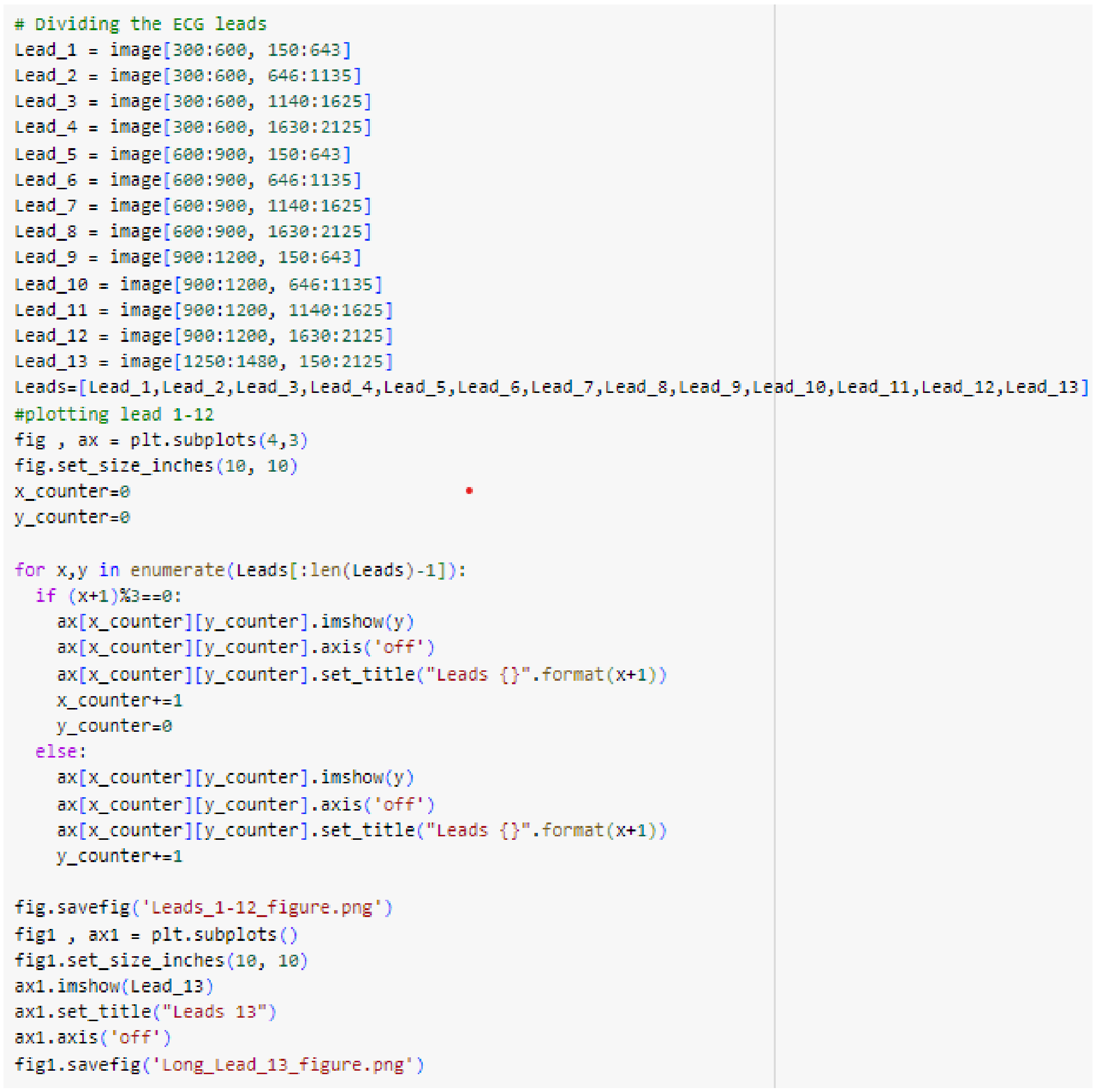


```
from skimage.io import imread
from skimage import color
import matplotlib.pyplot as plt
from skimage.filters import threshold_otsu, gaussian
from skimage.transform import resize
from skimage.metrics import structural_similarity
from skimage import measure
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
import joblib
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
import numpy as np
import os
from natsort import natsorted

def preprocess_image(image_path):
    image = imread(image_path)
    image_gray = color.rgb2gray(image)
    image_gray = resize(image_gray, (1572, 2213))

    image1 = imread('/content/drive/MyDrive/Research Models/ECG Image Recognition/Test/PMI/PMI(53).jpg')
    image1 = color.rgb2gray(image1)
    image1 = resize(image1, (1572, 2213))

    image2 = imread('/content/drive/MyDrive/Research Models/ECG Image Recognition/Test/Abnormal/HB(29).jpg')
    image2 = color.rgb2gray(image2)
    image2 = resize(image2, (1572, 2213))

    image3 = imread('/content/drive/MyDrive/Research Models/ECG Image Recognition/Test/Normal/Normal(14).jpg')
    image3 = color.rgb2gray(image3)
    image3 = resize(image3, (1572, 2213))

    image4 = imread('/content/drive/MyDrive/Research Models/ECG Image Recognition/Test/ECG Images of Myocardial Infarction Patients/MI(19).jpg')
    image4 = color.rgb2gray(image4)
    image4 = resize(image4, (1572, 2213))

    similarity_score = max(
        structural_similarity(image_gray, image1),
        structural_similarity(image_gray, image2),
        structural_similarity(image_gray, image3),
        structural_similarity(image_gray, image4)
    )
```

Figure 38 : Testing code implementation 01.

ECG pictures may be preprocessed using the Python function preprocess image before being fed into a machine learning model. It imports the required modules, including scikit-learn for machine learning tasks, matplotlib for visualization, and skimage for image processing. An image file path (image_path) is entered into the method. Using color.rgb2gray, it transforms the picture to grayscale, resizes it to a standard size of (1572, 2213), then reads the image using imread from skimage. It also loads four reference pictures, which are categorized as follows: ECG images of patients with myocardial infarction, abnormal, normal, and PMI (myocardial infarction). Every reference picture is scaled to match the input image's dimensions and converted to grayscale.

The function then calculates the structural similarity score between the input image and each reference image using structural_similarity from skimage.metrics. The maximum similarity score among these four scores is stored in the variable similarity_score. This score serves as an indicator of how closely the input image resembles any of the reference images, which can be useful for classification tasks or anomaly detection.

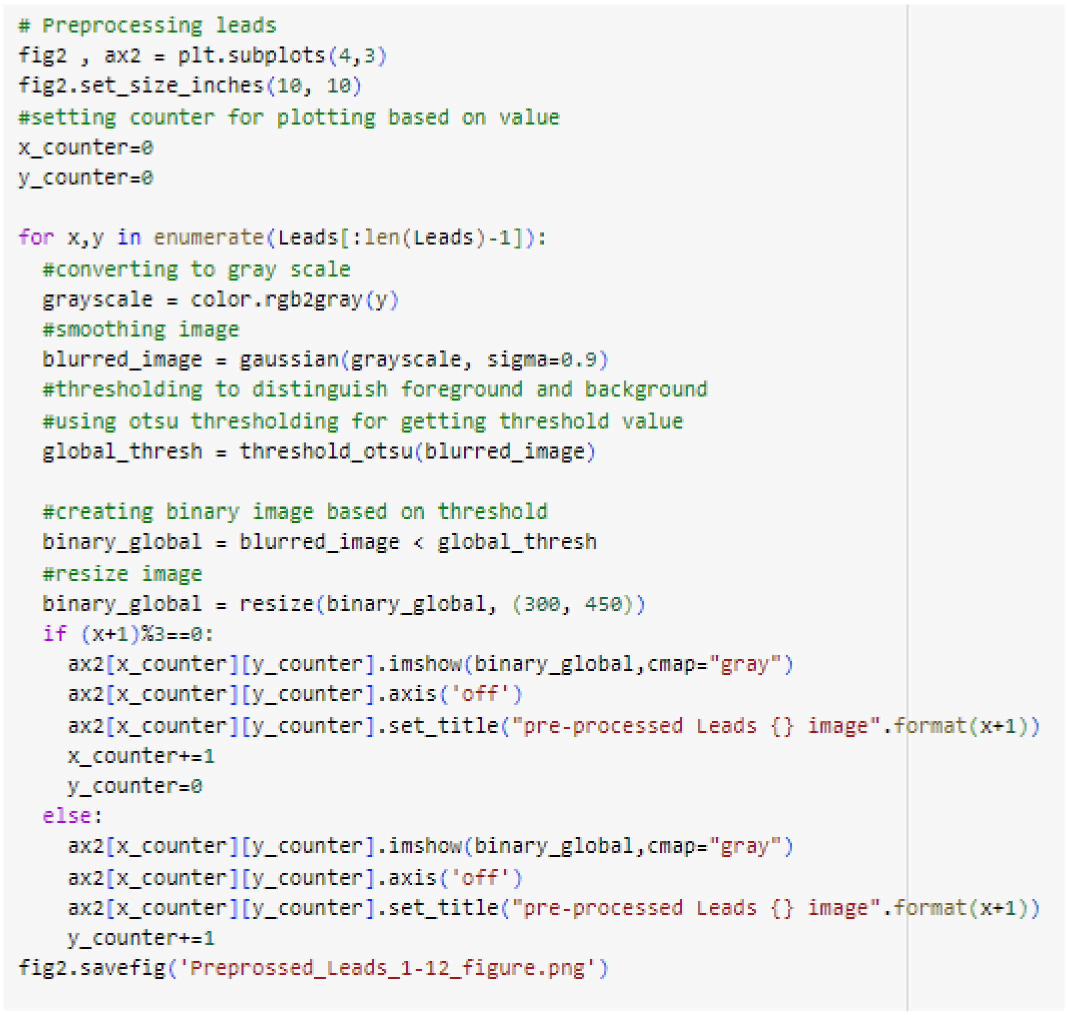


```python
# Dividing the ECG leads
Lead_1 = image[300:600, 150:643]
Lead_2 = image[300:600, 646:1135]
Lead_3 = image[300:600, 1140:1625]
Lead_4 = image[300:600, 1630:2125]
Lead_5 = image[600:900, 150:643]
Lead_6 = image[600:900, 646:1135]
Lead_7 = image[600:900, 1140:1625]
Lead_8 = image[600:900, 1630:2125]
Lead_9 = image[900:1200, 150:643]
Lead_10 = image[900:1200, 646:1135]
Lead_11 = image[900:1200, 1140:1625]
Lead_12 = image[900:1200, 1630:2125]
Lead_13 = image[1250:1480, 150:2125]
Leads=[Lead_1,Lead_2,Lead_3,Lead_4,Lead_5,Lead_6,Lead_7,Lead_8,Lead_9,Lead_10,Lead_11,Lead_12,Lead_13]
#plotting lead 1-12
fig , ax = plt.subplots(4,3)
fig.set_size_inches(10, 10)
x_counter=0
y_counter=0

for x,y in enumerate(Leads[:len(Leads)-1]):
  if (x+1)%3==0:
    ax[x_counter][y_counter].imshow(y)
    ax[x_counter][y_counter].axis('off')
    ax[x_counter][y_counter].set_title("Leads {}".format(x+1))
    x_counter+=1
    y_counter=0
  else:
    ax[x_counter][y_counter].imshow(y)
    ax[x_counter][y_counter].axis('off')
    ax[x_counter][y_counter].set_title("Leads {}".format(x+1))
    y_counter+=1

fig.savefig('Leads_1-12_figure.png')
fig1 , ax1 = plt.subplots()
fig1.set_size_inches(10, 10)
ax1.imshow(Lead_13)
ax1.set_title("Leads 13")
ax1.axis('off')
fig1.savefig('Long_Lead_13_figure.png')
```

Figure 39: Testing code implementation 02.

A conditional statement that determines whether the previously computed similarity_score is larger than 0.70 is present in this code block. If the requirement is satisfied, there may be a match since the input ECG image appears to be sufficiently similar to one of the reference images. In these situations, the function continues with additional processing and visual aids.First, it uses matplotlib's plt.imshow and plt.show() to display the input picture. Next, it separates the image's ECG leads into 13 segments, each of which represents a distinct lead. For every lead region, these segments are retrieved using predetermined pixel coordinates.

Next, the extracted leads 1 through 12 are plotted in a 4x3 grid using subplots. Every lead has its own subplot that displays the lead image and lead number in the accompanying title. The file name of the generated subplot grid is 'Leads_1-12_figure.png'.Finally, lead 13 is plotted and saved as a separate image named "Long_Lead_13_figure.png." Lead 13 is shown separately from the others since its size and aspect ratio set it apart from the others.

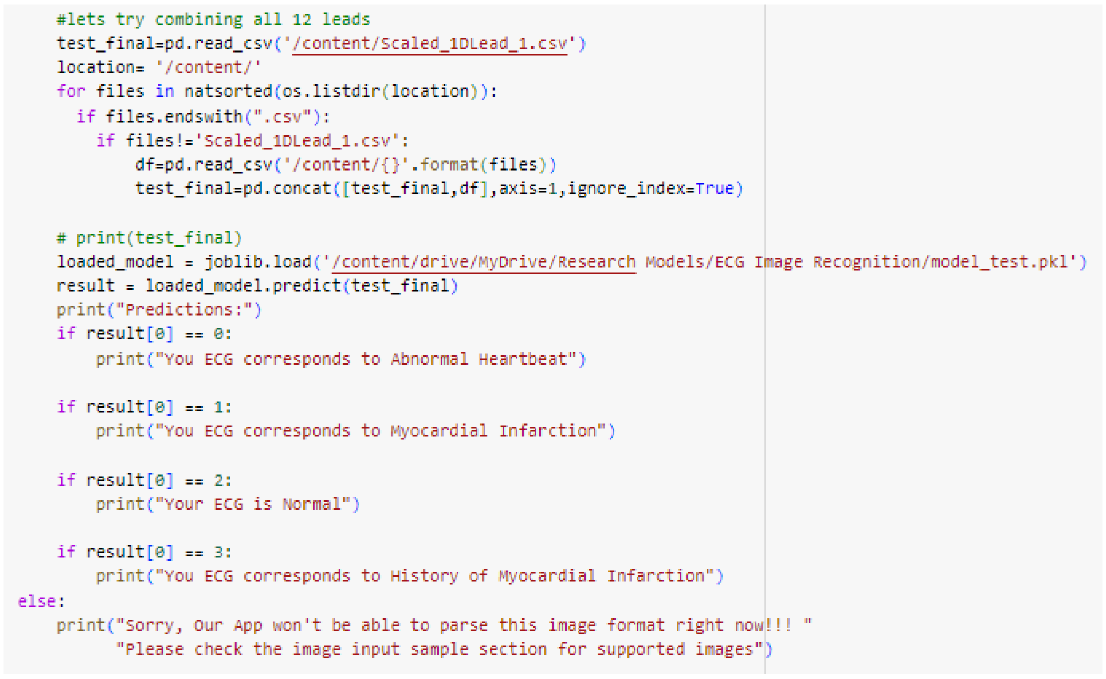


```
# Preprocessing leads
fig2 , ax2 = plt.subplots(4,3)
fig2.set_size_inches(10, 10)
#setting counter for plotting based on value
x_counter=0
y_counter=0

for x,y in enumerate(Leads[:len(Leads)-1]):
  #converting to gray scale
  grayscale = color.rgb2gray(y)
  #smoothing image
  blurred_image = gaussian(grayscale, sigma=0.9)
  #thresholding to distinguish foreground and background
  #using otsu thresholding for getting threshold value
  global_thresh = threshold_otsu(blurred_image)

  #creating binary image based on threshold
  binary_global = blurred_image < global_thresh
  #resize image
  binary_global = resize(binary_global, (300, 450))
  if (x+1)%3==0:
    ax2[x_counter][y_counter].imshow(binary_global,cmap="gray")
    ax2[x_counter][y_counter].axis('off')
    ax2[x_counter][y_counter].set_title("pre-processed Leads {} image".format(x+1))
    x_counter+=1
    y_counter=0
  else:
    ax2[x_counter][y_counter].imshow(binary_global,cmap="gray")
    ax2[x_counter][y_counter].axis('off')
    ax2[x_counter][y_counter].set_title("pre-processed Leads {} image".format(x+1))
    y_counter+=1
fig2.savefig('Preprossed_Leads_1-12_figure.png')
```

Figure 40: Testing code implementation 03.

Preprocessing the collected ECG leads from the previous phase is the main goal of this code block. It seeks to improve lead quality to facilitate more effective feature extraction and analysis. The code iterates through each lead picture, uses color.rgb2gray to convert it to grayscale, and then uses Gaussian smoothing (gaussian(grayscale, sigma=0.9)) to minimize noise and improve the signal's clarity. It uses Otsu's thresholding method (threshold_otsu) to separate the pixels in the foreground and background after smoothing. Using the picture histogram as a guide, this approach automatically determines the ideal threshold value. This threshold is then applied to the picture to binarize it, producing a binary image with white foreground pixels and black background pixels. To guarantee uniformity across all leads, the binary picture is also scaled to a common size of (300, 450).Subplots are used to plot the preprocessed lead pictures in a 4x3 grid. Each subplot shows the preprocessed lead image and its accompanying caption, which indicates the lead number. 'Preprocessed_Leads_1-12_figure.png' is the file name of the subplot grid that is produced.

The main tasks of this code block are to visualize and extract signals from Lead 13 and to derive contours from Leads 1 through 12.Lead 13 is first processed in a manner akin to that of the preceding leads, which involves converting it to grayscale, smoothing it with Gaussian filtering, and thresholding to produce a binary picture. Otsu's approach is used to calculate the threshold value. Plotting follows, and the binary picture is saved as "Preprocessed_Leads_13_figure.png." Subsequently, it takes shape from Leads 1 through 12. It smoothes, thresholds, and transforms the binary picture to grayscale for each lead. The biggest contour is then chosen as the main signal once contours are located using the measure.find_contours function. This contour is scaled to a standard size and plotted on a 4x3 grid of subplots, where the contour of each lead is represented by a separate subplot. Saved as "Contour_Leads_1-12_figure.png" is the final grid.Furthermore, the recovered contour data is stored as distinct CSV files called "Scaled_1DLead_{lead_no}.csv" for each lead after being scaled using Min-Max scaling.



```
    #lets try combining all 12 leads
    test_final=pd.read_csv('/content/Scaled_1DLead_1.csv')
    location= '/content/'
    for files in natsorted(os.listdir(location)):
      if files.endswith(".csv"):
        if files!='Scaled_1DLead_1.csv':
            df=pd.read_csv('/content/{}'.format(files))
            test_final=pd.concat([test_final,df],axis=1,ignore_index=True)

    # print(test_final)
    loaded_model = joblib.load('/content/drive/MyDrive/Research Models/ECG Image Recognition/model_test.pkl')
    result = loaded_model.predict(test_final)
    print("Predictions:")
    if result[0] == 0:
        print("You ECG corresponds to Abnormal Heartbeat")

    if result[0] == 1:
        print("You ECG corresponds to Myocardial Infarction")

    if result[0] == 2:
        print("Your ECG is Normal")

    if result[0] == 3:
        print("You ECG corresponds to History of Myocardial Infarction")
else:
    print("Sorry, Our App won't be able to parse this image format right now!!! "
          "Please check the image input sample section for supported images")
```

Figure 41: Testing code implementation 04.

This code block aims to create a single dataframe for prediction using a pre-trained machine learning model by combining the retrieved signals from each of the ECG's 12 leads. First, it loads the first lead's data from 'Scaled_1DLead_1.csv' into an empty dataframe called test_final. Next, it concatenates the data from each CSV file in the designated directory (apart from "Scaled_1DLead_1.csv") to test_final along the column’s axis. As a result, a single dataframe including the signals from all 12 leads is created.

Then, using joblib, it imports a machine learning model that has already been trained ('model_test.pkl'), and it makes use of this model to predict the ECG classification based on the combined signals in test_final. It prints a matching message giving the ECG's categorization based on the expected outcome.

Here I decide to continue my research using KNN model. The main reason behind it was the increase in accuracy of KNN model. The results obtained here are classified under four categories. They can be introduced as 'Normal heartbeat', 'Abnormal heartbeat', 'Myocardial infarction', 'History of Myocardial infarction'. And these four categories are classified under three levels of risk. If the result is normal heartbeat, he or she is at low risk of having a heart attack, if abnormal heartbeat he or she is at some level of risk (medium risk), and if myocardial infarction or history of myocardial infarction is indicated here, he or she is at high risk. By retrieving the results of such reports obtained from time to time in the data base, a variable graph can be obtained between the risk levels of that person and depending on the nature of the function, it can be determined whether the risk level is gradually increasing over time or whether the risk level is decreasing.

03. RESULT & DISCUSSION

3.1 Results

When used to predict the likelihood of future heart attacks, the k-Nearest Neighbors (KNN) model has a remarkable training accuracy of 96%. However, when evaluated on fresh, unpublished data, this performance indicator falls to 88%. The reason for the disparity in accuracy between training and testing is because each patient's ECG report is distinct by nature. The graphical depictions of the electrical activity of the heart found in each person's ECG readings have distinct patterns and variances that make it difficult for the model to extrapolate from the training set to new situations.

The significant individual variation in ECG patterns is one of the main causes of this decline in accuracy. The electrical activity of the heart, as recorded by an ECG, can be affected by a variety of factors, such as age, physical state, genetic makeup, and pre-existing medical disorders. When presented with a variety of fresh inputs, these variances make it challenging for the KNN model, which is based on similarity metrics, to reliably classify ECGs with high accuracy.

The KNN model finds the k most comparable cases by comparing a fresh ECG report to the training set. The majority danger level of these k neighbors is then used to calculate the risk level of the new report. Although this method works well when fresh data points are like those in the training set, it has trouble processing the high-dimensional and complicated nature of ECG data, where even minute variations might reveal important shifts in heart health.

Furthermore, there is a degree of abstraction introduced by the division of risk levels into three groups: low risk for normal ECGs (represented by 0), medium risk for irregular heartbeats (represented by 1), and high risk for myocardial infarction and its history (represented by 2). This division may not adequately reflect the subtle variations amongst individual cases. Even though it is required for practical implementation, this simplification could be part of the reason why the model performs less well on a range of testing data.

The way that ECG data is represented is another crucial factor. Feature extraction is the step involved in transforming ECG pictures into a format that the KNN model can analyze. This procedure might result in mistakes or the loss of important information. The model's predictions will degrade if the extracted features are unable to precisely capture the key ECG properties associated with heart attack risk. Because KNN is being used here, it also indicates that the model's sensitivity to the selection of k—the number of neighbors considered-is considerable. An incorrect selection of k can result in either underfitting (where the model performs poorly on both training and testing data) or overfitting (where the model performs well on training data but poorly on testing data). Finding the ideal k is essential to achieving a balance in the trade-off between variance and bias.

Furthermore, as ECG readings are time-series data, precise forecasting depends on understanding the temporal correlations between readings. Because standard KNN does not naturally take these temporal dependencies into account, it may miss significant long-term trends and patterns. Time-series analysis combined with hybrid models or more sophisticated methods may be able to better capture these dynamics and increase prediction accuracy.

It is also important to remember that both the quantity and quality of the training data may have an impact on the model's performance. An inadequately sized or skewed dataset may result in a poorly generalizing model. To increase the resilience and accuracy of the model on untested data, it is important to make sure the training set is extensive, diversified, and includes a wide range of ECG patterns.

A convenient method of tracking heart health over time is provided by the mobile application that creates a graph with Dates on the x-axis and Risk Level on the y-axis (Figure 42). Users and healthcare professionals may monitor changes in risk levels with the use of this visual depiction, which may assist spot patterns or initiate early treatments. However, the accuracy and dependability of the underlying model have a major impact on how useful such a tool is.

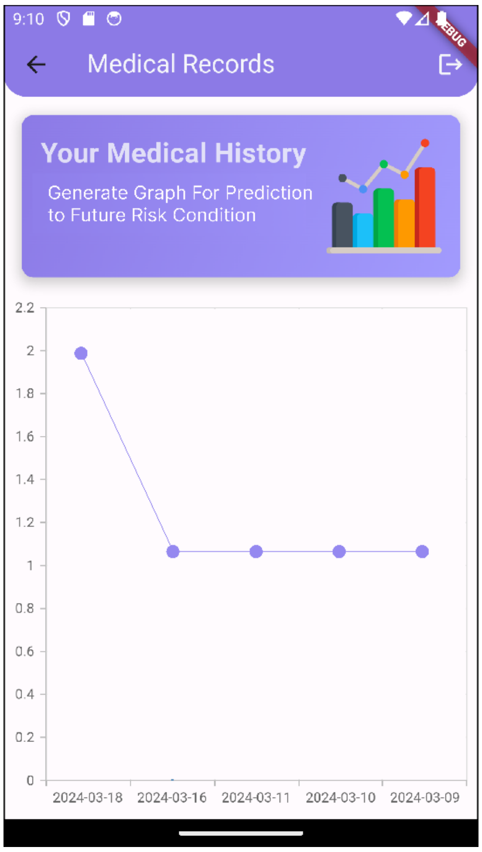


Figure 42: A graph generated by the mobile application.

3.2 Research Findings

The study's conclusions show how well an algorithm known as the k-nearest neighbors (KNN) algorithm works when evaluating ECG data to forecast the likelihood of a heart attack. The model focused on categorizing ECG recordings into four groups: Normal Person Heartbeat, Abnormal Heartbeat, History of Myocardial Infarction, and Myocardial Infarction. It was constructed utilizing expert knowledge and data from Maharagama Apeksha Hospital. An essential part of this approach was extracting 12 leads from ECG scans and creating a 13th lead for thorough signal analysis, all in the format appropriate for machine learning analysis.

Preprocessing the ECG pictures, which included smoothing them to eliminate high-frequency noise and converting them to grayscale, was a major factor in the model's 96% training accuracy. This high accuracy shows that the KNN model successfully used the training data to identify the patterns linked to each ECG category. These outcomes were mostly due to the methodological measures implemented to guarantee data quality and relevance, such as creating rectangular zones of interest for lead extraction and picture smoothing.

The model's testing accuracy of 88% indicates a noticeable drop when applied to fresh, unknown data, despite its remarkable training accuracy. This decline can be explained by the fact that each patient's ECG result is distinct and highly personalized. The KNN algorithm, which uses similarity metrics to categorize fresh data points, is severely challenged by the diversity in ECG rhythms. This result emphasizes the intricacy and variability of ECG signals as well as the inherent challenge of developing a universally applicable model.

The practical technique of classifying ECG data into three risk levels (low, medium, and high) based on the first four categories improves the model's use in a clinical environment. Because of this stratification, it is simpler for patients and healthcare professionals to comprehend and act upon the risk assessments. It also simplifies the interpretation of data. A useful tool for continuing patient treatment, the mobile application's graphical depiction of risk levels over time helps with early diagnosis and monitoring as well.

Despite the testing accuracy decline, the model's efficacious analysis of ECG pictures and risk level prediction emphasizes its prospective practical applications. The results, however, indicate that in order to increase the model's generalizability, more improvement and refining are required. The observed performance difference between training and testing accuracies may be resolved by adding more representative and varied samples to the training dataset and investigating advanced or hybrid machine learning approaches.

3.3 Discussion

This study's main goal was to create and evaluate several machine learning models for analyzing ECG data and forecasting the likelihood of upcoming heart attacks. Support Vector Machine (SVM), Logistic Regression, and k-Nearest Neighbors (KNN) are some of the models investigated. The efficacy of each model was assessed by looking at how well it could generalize from training data to test data that hadn't been seen before. The findings suggest that although all models have potential, they all have certain advantages and disadvantages.

The KNN model was created with scikit-learn with an extensive pipeline configuration. The model successfully learnt the patterns in the training dataset, as evidenced by its remarkable 96% training accuracy. But the testing accuracy fell to 88%, indicating that the model may have trouble applying fresh data. The intrinsic variability in ECG signals between patients is the cause of this accuracy reduction. Because ECG data is high-dimensional and even tiny variations can have clinically significant effects, the KNN algorithm, which uses proximity metrics to categorize new occurrences, faces difficulties.

Another machine learning model used in this study, logistic regression, performed well, demonstrating a testing accuracy of 96.2%. In order to predict categorical outcomes, this model optimizes a logistic function by a linear approach to ECG data classification. Using GridSearchCV, the hyperparameters were adjusted, paying particular attention to the regularization strength (C) and penalty type (l2). Because it can clearly understand the model's predictions in terms of probability, logistic regression is a useful tool for clinical decision-making. Because of its more straightforward and easily understood model structure, it appears to have good generalization to new data based on its high accuracy on the test set.

Strong performance was also shown by the Support Vector Machine (SVM) model, which attained a testing accuracy of almost 92.7%. Strong vector machines (SVMs) are capable of efficiently handling high-dimensional data by determining the ideal hyperplane that optimizes the margin between distinct classes. The regularization parameter (C) and the kernel coefficient (gamma) of the SVM model were carefully adjusted in this work. The SVM is a good option for ECG classification jobs due to its resilience in addressing complicated, non-linear correlations in the data, even if it is somewhat less accurate than Logistic Regression.

After analyzing the three models, it was found that Logistic Regression was the most accurate on the test set, surpassing both KNN and SVM models by a little margin. Because of its resilience against overfitting and its ability to handle linear connections with efficiency—especially when the data is well-regularized—Logistic Regression has a better accuracy rate. In addition, clinical applications, where comprehending the reasons behind a prediction is critical, find Logistic Regression to be a compelling option because of its interpretability and simplicity.

Despite having a high training accuracy, the KNN model's poor testing accuracy draws attention to the overfitting problem. The distance measure and parameter k selection have a major impact on KNN's performance. Although GridSearchCV was used in this work to determine the ideal k, the model's performance on the test set still declined significantly. This implies that, even if KNN can accurately represent the intricate details of the training set, it could struggle to generalize to new, unobserved data, particularly when there is a lot of fluctuation and noise present in the ECG signals.

However, the SVM model has advantages in handling high-dimensional feature spaces and non-linear interactions, albeit obtaining a little lower accuracy than Logistic Regression. It is possible to translate data into higher dimensions using kernel functions in support vector machines (SVMs), which facilitates the separation of classes that are not linearly separable in the original feature space. This feature is especially helpful for ECG data, which frequently shows intricate, non-linear patterns. SVMs may, however, be computationally demanding, and careful tuning is necessary since the selection of hyperparameters affects how well they function.

The significance of feature engineering and data pretreatment in ECG analysis is further shown by the comparison of various models. In order to prepare the data for machine learning analysis, it was essential to extract 12 leads from the ECG pictures, convert them to grayscale, smooth them down to eliminate high-frequency noise, and create a 13th lead. These signals were successfully entered into the models after being converted into a structured CSV file, highlighting the need for careful data preparation.

Furthermore, the models' clinical relevance was guaranteed by the incorporation of expert information during the machine learning pipeline's development. The cooperation of domain experts and data scientists is essential to creating models that function well statistically and offer valuable insights in a therapeutic setting. The models' practical usability is further enhanced by the classification of ECG data into danger categories and the utilization of a mobile application to show these hazards.

FUTURE WORKS

The KNN, SVM, and logistic regression models provide encouraging results, demonstrating the promise of machine learning in ECG analysis and heart attack risk prediction. Nonetheless, more investigation and improvement are required in a few areas to improve the models' functionality and suitability for use in therapeutic contexts. Future research should concentrate on enhancing the quality of the data, growing the dataset, and investigating cutting-edge machine learning methods.

The addition of additional representative and diverse ECG samples to the training dataset is an important subject for future study. Despite its effectiveness, the present dataset does not fully capture the range of changes in ECG across various demographic groups and clinical situations. The generalizability and robustness of the models would be enhanced by expanding the dataset to include ECG records from a wider patient population, including various age groups, ethnicities, and comorbidities. Collaboration with more hospitals and medical facilities might help achieve this.

More advanced preparation methods may be used to further enhance the quality of ECG data in addition to growing the dataset. Although the present methodology consists of noise reduction and grayscale conversion, future research might investigate more sophisticated signal processing techniques to improve feature extraction, such as Fourier analysis or wavelet transformations. By capturing more subtle patterns in the ECG data, these strategies may improve the performance of the model.

Investigating ensemble learning strategies is another direction that future research should go. Putting many models together, such SVM, Logistic Regression, and KNN, might maximize the positive aspects of each model while reducing its negative aspects. One may use strategies like bagging, boosting, or stacking to develop a prediction model that is more reliable and accurate. This strategy may enhance overall performance, especially when various models have characteristics that complement one another.

Deep learning techniques have a lot of potential for ECG analysis as well. Specifically, convolutional neural networks, or CNNs, operate well with image-based data and may be used to automatically identify hierarchical patterns in raw or preprocessed ECG pictures. For the analysis of ECG time series data, recurrent neural networks (RNNs) and long short-term memory (LSTM) networks—which function well with sequential data—might also be investigated. By identifying intricate patterns in the data, these sophisticated neural network designs have the potential to perform better than conventional machine learning models.

Moreover, adding more patient data and clinical context to the models may improve their prediction ability. A more complete picture of a patient's health state might be obtained, for instance, by combining ECG data with electronic health record (EHR) data, which includes patient history, prescription information, and lifestyle variables. Better clinical judgments and more precise risk assessments may result from this multimodal approach.

The interpretability of the model is still a major challenge, particularly in clinical applications where it is necessary to comprehend the reasoning behind the predictions. The development of explainable AI (XAI) methods that can yield visible and understandable model insights need to be the main emphasis of future research. To increase clinician acceptability and trust, techniques like SHAP (SHapley Additive exPlanations) values or LIME (Local Interpretable Model-agnostic Explanations) might be used to clarify the contributions of various characteristics to the model's predictions.

04. CONCLUSION

There is a lot of research and discovery to control the ever-increasing population of heart patients. It is not a process that can be completely stopped. But controlling heart disease is a practical process that any of us can do if we are interested. In the mobile app designed for heart disease sufferers, I developed ECG classification and analysis of ECG reports to predict risk levels is the main focus here.

Through the analysis of ECG data, the study successfully illustrates the application of a machine learning technique, namely the k-nearest neighbors (KNN) method, to forecast the likelihood of future heart attacks. The technique used included taking patient data from Maharagama Apeksha Hospital and extracting and processing the ECG signals. The ECG signals were classified into four categories: Normal Person Heartbeat, Abnormal Heartbeat, History of Myocardial Infarction, and Myocardial Infarction. These categories were then further divided into three risk levels for practical monitoring and prediction, thanks to this thorough approach.

This study's effective extraction and processing of ECG data is one of its major accomplishments. The original ECG picture was extracted into 12 separate leads, converted to grayscale, and then smoothed to eliminate high-frequency noise, ensuring that the data utilized for analysis was accurate and pure. An inventive move that improved the caliber of the input data for the KNN model was the construction of the 13th lead, which was achieved by merging the first leads and storing it as an image file for additional examination.

The necessity of efficient data preparation in machine learning applications is shown by the translation of ECG wave coordinates into a CSV file format and the use of these values as inputs for the KNN model. The excellent training accuracy of 96% was largely attributable to this strategy, which made it easier to enter the ECG data into the model in an organized and quantitative manner.

Although the training accuracy is great, the testing accuracy of 88% indicates a significant problem with the model's generalizability. The fact that each patient's ECG report is distinct might be the reason for the accuracy decline. The complexity of ECG signals and the requirement for models that can adjust to such fluctuation are highlighted by this discovery. In the future, sophisticated methods or hybrid models that might more effectively manage the variety of ECG patterns could be investigated.

A useful and therapeutically relevant method is the division of ECG signals into three risk levels: low, medium, and high, based on the initial four categories. Healthcare professionals and people can more easily perceive and make decisions thanks to this simplification into risk levels. The capacity of the mobile application to visually depict these risk levels over time makes it a useful tool for continuing observation and early identification of any cardiac problems.

The findings show that even though the KNN model is a reliable tool for risk assessment and ECG analysis, further improvement is clearly needed based on how well it performs with unknown data. Providing a training dataset that is more extensive and diverse may enhance the model's accuracy and generalizability when used to test data. This emphasizes how crucial it is to continuously validate models and change them in light of new information.

A key strength of the work is the technique, which relies on expert knowledge to direct the machine learning process. This partnership guarantees that the model's development is based on clinical relevance and accuracy, giving the analytical process a strong basis. In order to correctly interpret the ECG data and make sure that the model's predictions match clinical standards and observations, expert input is essential. This model's incorporation into a mobile application offers an innovative approach to healthcare. It provides patients with an easy-to-use interface to monitor their heart health, which may improve patient compliance with recommended treatment plans. This technological advancement makes sophisticated healthcare available to a wider audience by bridging the gap between intricate ECG analysis and routine patient care.

The study's findings demonstrate how machine learning has the potential to revolutionize heart attack risk assessment and ECG interpretation. Even though the KNN model performs well, especially during training, there is still room for improvement as seen by the decline in testing accuracy. Subsequent investigations ought to concentrate on augmenting the model's capacity to generalize over heterogeneous patient data, maybe by means of sophisticated algorithms or more extensive, varied datasets. To optimize this approach's therapeutic value and continue to refine it, technical improvements and expert knowledge must be continuously integrated.

In summary, this study establishes a strong foundation for forthcoming advancements in automated electrocardiogram analysis and heart attack risk assessment, therefore supporting the overarching objective of enhancing cardiovascular healthcare using inventive technology. Researchers and clinicians looking to use machine learning to improve health outcomes may learn a great deal from the findings and approaches presented.

05. REFERENCES

[1] M. Sandhya Pruthi, "Mayo Clinic," Mayo Foundation for Medical Education and Research (MFMER), [Online]. Available: https://www.mayoclinic.org/tests-procedures/ekg/about/pac-20384983.

[2] P. Slack, "Healio," Healio, [Online]. Available: https://www.healio.com/cardiology/learn-the-heart/ecg-review/ecg-interpretation-tutorial/qrs-complex.

[3] r. S. Denis, "Cardiologs - AI serving cardiologs," Cardiologs Technologies SAS, 2021. [Online]. Available: https://cardiologs.com/blog/ai-systems-powered-by-deep-learning/.

[4] W. H. ORGANIZATION, "SRI LANKA: CORONARY HEART DISEASE," WORLDLIFEEXPECTANCY, 2020. [Online]. Available: https://www.worldlifeexpectancy.com/sri-lanka-coronary-heart-disease.

[5] "AIA.com," AIA Insurance, 28 Octomber 2021. [Online]. Available: https://www.aialife.com.lk/en/our-products/Health.

[6] V. H. Buch, I. Ahmed and M. Maruthappu, "Artificial intelligence in medicine: Current trends and future possibilities," *British Journal of General Practice,* vol. 68, no. 668, pp. 143-144, 2018.

[7] K. Shameer, K. W. Johnson, A. Yahi, R. Miotto, L. I. Li, D. Ricks, J. Jebakaran, P. Kovatch, P. P. Sengupta, A. Gelijns, A. Moskovitz, B. Darrow, D. L. Reich and A. Kasarskis, "PREDICTIVE MODELING OF HOSPITAL READMISSION RATES USING ELECTRONIC MEDICAL RECORD-WIDE MACHINE LEARNING: A CASE-STUDY USING MOUNT SINAI HEART FAILURE COHORT," 2017.

[8] B. Nithya and D. V. Ilango, "Predictive Analytics in Health Care Using Machine Learning Tools and Techniques," in *International Conference on Intelligent Computing and Control Systems (ICICCS)*, Bangalore, 2017.

[9] E. G. Ross, K. Jung, J. T. Dudley, L. Li, N. J. Leeper and N. H. Shah, "Predicting Future Cardiovascular Events in Patients with Peripheral Artery Disease Using Electronic Health Record Data," *Circulation: Cardiovascular Quality and Outcomes,* vol. 12, no. 3, p. 10, 2019.

[10] A. Mincholé, J. Camps, A. Lyon and B. Rodríguez, "Machine learning in the electrocardiogram," in *Journal of Electrocardiology*, Maastricht, 2019.

[11] G. Battineni, G. G. Sagaro, N. Chinatalapudi and F. Amenta, "Applications of machine learning predictive models in the chronic disease diagnosis," *Journal of Personalized Medicine,* vol. 10, no. 2, 2020.

[12] N. A. Nayan, H. A. Hamid, M. Z. Suboh, R. Jaafar, N. Abdullah, N. A. M. Yusof, M. A. Hamid, N. F. Zubiri, A. S. K. Arifin, S. M. A. Daud and M. A. Kamaruddin,

"Cardiovascular disease prediction from electrocardiogram by using machine learning," *International journal of online and biomedical engineering,* vol. 16, no. 7, pp. 34-48, 2020.

[13] M. T. García-Ordás, M. Bayón-Gutiérrez, C. Benavides, J. Aveleira-Mata and J. A. Benítez-Andrades, "Heart disease risk prediction using deep learning techniques with feature augmentation," *Multimedia Tools and Applications,* vol. 82, no. 20, pp. 31759-31773, 2023.

[14] M. N. Islam, K. R. Raiyan, S. Mitra, M. . . Mannan, T. Tasnim, A. O. Putul and A. B. Mandol, "Predictis: an IoT and machine learning-based system to predict risk level of cardio-vascular diseases," *BMC Health Services Research,* vol. 23, no. 1, 2023.

[15] T. Habineza, A. H. Ribeiro, D. Gedon, J. A. Behar, A. L. P. Ribeiro and T. B. Schön, "End-to-end risk prediction of atrial fibrillation from the 12-Lead ECG by deep neural networks," *Journal of Electrocardiology,* vol. 81, pp. 193-200, 2023.

[16] M. A. Sarwar, N. Kamal, W. Hamid and M. A. Shah, "Prediction of Diabetes Using Machine Learning Algorithms in Healthcare," in *International Conference on Automation & Computing,* Newcastle, 2018.

[17] M. Ganesan and N. Sivakumar, "IoT based heart disease prediction and diagnosis model for healthcare using machine learning models," in *IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, Pondicherry, 2019.

06. GLOSSARY

Term	Definition
ECG or **EKG** (electrocardiogram)	An electrocardiogram (ECG or EKG) is one of the simplest and fastest tests used to evaluate the heart.
CVD (Cardiovascular)	Cardiovascular diseases (CVDs) are a group of disorders of the heart and blood vessels.
PCA (Principal component analysis)	Principal component analysis, or PCA, is a statistical procedure that allows you to summarize the information content in large data tables by means of a smaller set of "summary indices" that can be more easily visualized and analyzed.
DNN (Deep Neural Network)	Deep Neural Network (DNN), also called Deep Nets, is a neural network with a certain level of complexity.
PAD (Peripheral arterial disease)	Peripheral arterial disease (PAD) in the legs or lower extremities is the narrowing or blockage of the vessels that carry blood from the heart to the legs.
AI (Artificial intelligence)	Artificial intelligence (AI) refers to computer systems capable of performing complex tasks that historically only a human could do, such as reasoning, making decisions, or solving problems.
KNN (k-nearest neighbor)	kNN, or the k-nearest neighbor algorithm, is a machine learning algorithm that uses proximity to compare one data point with a set of data it was trained on and has memorized to make predictions.
SVM (support vector machine)	A support vector machine (SVM) is a supervised machine learning algorithm that classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space.
ML (Machine learning)	Machine learning (ML) is a branch of artificial intelligence (AI) and computer science that focuses on the using data and algorithms to enable AI to imitate the way that humans learn, gradually improving its accuracy.
MI (Myocardial infarction)	Myocardial infarction (MI), colloquially known as "heart attack," is caused by decreased or complete cessation of blood flow to a portion of the myocardium.

07. APPENDICES