A PROJECT REPORT

ON

IDENTIFICATION OF CARDIOVASCULAR DISEASE IN ECG IMAGES

Submitted in the partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

P.Hariharan	T.Bhavyasree	K.Shanmukh
21BK5A0516	20BK1A05G6	20BK1A05H9

Under the guidance of
Dr.V.VAITHEESHWARAN Ph.D.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

St. Peter's Engineering College (UGC Autonomous)

Approved by AICTE, New Delhi, and NAAC Accrediated with 'A' Grade,

Affiliated to JNTU, Hyderabad, Telangana

2020-2024

Giving Wings to Thoughts
ESTD : 2007

St. PETER'S
ENGINEERING COLLEGE
UGC - AUTONOMOUS

Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programe Accredited (EEE, CSE, ECE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that a Project entitled **"IDENTIFICATION OF CARDIOVASCULAR DISEASE IN ECG IMAGE"** is carried out by **P.Hariharan (21BK5A0516), T.Bhavyasree (20BK1A05G6) & K.Shanmukh (20BK1A05H9),** in partial fulfilment for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work done by them under my supervision during the academic year 2023– 2024.

GUIDE	**HEAD OF THE DEPARTMENT**
Dr.V.VAITHEESHWARAN, Ph.D	**Dr.M. Dileep Kumar, M.Tech, Ph.D**
Assosiate Professor	**Professor& HOD**
Department of CSE	Department of CSE

PROJECT COORDINATOR	**EXTERNAL EXAMINER**
Mr. A. Senthil Murugan,M.E.,M.B.A.,(Ph.D)	
Assistant Professor	
Department of CSE	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ACKNOWLEDGEMENT

We sincerely express our deep sense of gratitude to **Dr.V.VAITHEESHWARAN,** for his valuable guidance, encouragement and cooperation during all phases of the project.

We are greatly indebted to our Project Coordinator **Mr. Senthil Murugan,** for providing valuable advice, constructive suggestions and encouragement without whom it would not been possible to complete this project.

It is a great opportunity to render our sincere thanks to **Dr. M. Dileep Kumar,** Head of the Department, Computer Science and Engineering for his timely guidance and highly interactive attitude which helped us a lot in successful execution of the Project.

We are extremely thankful to our Principal **Dr. K. SREE LATHA,** who stood as an inspiration behind this project and heartfelt for her endorsement and valuable suggestions.

We respect and thank our Academic Director **Mrs. T. SAROJA REDDY** and secretary **Sri.T.V.REDDY,** for providing us an opportunity to do the project work at **St. PETER'S ENGINEERING COLLEGE** and we are extremely thankful to them for providing such a nice support and guidance which made us to complete the project.

We also acknowledge with a deep sense of reverence, our gratitude towards our parents, who have always supported us morally as well as economically. We also express gratitude to all our friends who have directly or indirectly helped us to complete this project work. We hope that we can build upon the experience and knowledge that we have gained and make a valuable contribution towards the growth of the society in coming future.

P.Hariharan (21BK5A0516)

T.Bhavyasree (20BK1A05G6)

K.Shanmukh (20BK1A05H9)

Giving Wings to Thoughts
ESTD : 2007

St. PETER'S ENGINEERING COLLEGE

UGC - AUTONOMOUS

Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programe Accredited (EEE, CSE, ECE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE VISION

To be a renowned Educational Institution that moulds Students into Skilled Professionals fostering Technological Development, Research and Entrepreneurship meeting the societal needs.

INSTITUTE MISSION

IM1: Making students knowledgeable in the field of core and applied areas of Engineering to innovate Technological solutions to the problems in the Society.

IM2: Training the Students to impart the skills in cutting edge technologies, with the help of relevant stake holders.

IM3: Fostering conducive ambience that inculcates research attitude, identifying promising fields for entrepreneurship with ethical, moral and social responsibilities.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT VISION

To be a vibrant nodal center for Computer Science & Engineering Education, Research that makes the students contribute to technologies for IT, IT-Enabled Services; to involve in innovative research on thrust areas of industry and academia; to establish start-ups supporting major players in the industry.

DEPARTMENT MISSION

DM1: Emphasize project-based learning by employing state-of-art technologies, and algorithms in software development for the problems in interdisciplinary avenues.

DM2: Involve stakeholders to make the students industry ready with training in skill-oriented computer application software.

DM3: Facilitate learning the theoretical nuances of Computer Science, and Computer Engineering courses and motivate to carry out research in both core and applied areas of CSE.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: Graduates shall involve in research & development activities in industry and government arenas to conceive useful products for society.

PEO 2: Graduates shall be entrepreneurs contributing to national development in the fields of Computer Science based technologies.

PEO 3: Graduates shall be team leaders working for software development, and maintenance in the fields of the software industry and government agencies

Giving Wings to Thoughts
ESTD : 2007

St. PETER'S
ENGINEERING COLLEGE
UGC - AUTONOMOUS
Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programe Accredited (EEE, CSE, ECE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

<u>PROGRAM OUTCOMES (POs)</u>

Engineering Graduates will be able to:

1: ENGINEERING KNOWLEDGE: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2: PROBLEM ANALYSIS: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using the first principles of mathematics, natural sciences, and engineering sciences.

3: DESIGN/DEVELOPMENT OF SOLUTIONS: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and the cultural, societal, and environmental considerations.

4: CONDUCT INVESTIGATIONS OF COMPLEX PROBLEMS: Use research-based knowledge and research methods including design of experiments, analysis, interpretation of data, and synthesis of the information to provide valid conclusions.

5: MODERN TOOL USAGE: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6: THE ENGINEER AND SOCIETY: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to the professional engineering practice

7: ENVIRONMENT AND SUSTAINABILITY: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8: ETHICS: Apply ethical principles and commit to professional ethics and, responsibilities and norms of the engineering practice.

9: INDIVIDUAL AND TEAM WORK: Function effectively as an individual, and as a member or leader in diverse teams, and multidisciplinary settings.

10: COMMUNICATION: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and draft effective reports and design documentation, make an effective presentation, give, and receive clear instructions.

11: PROJECT MANAGEMENT AND FINANCE: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's work, as a member and leader in a team, to manage projects and in a multidisciplinary environment.

12: LIFE-LONG LEARNING: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadcast context of technological changes.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM SPECIFIC OBJECTIVES (PSO'S)

PSO-1 Design and develop computing subsystems for data storage, communication, information processing, and knowledge discovery.

PSO-2 Design algorithms for real-world problems focusing on execution, and complexity analysis considering the security, cost, quality, and privacy parameters in software development.

Giving Wings to Thoughts
ESTD : 2007

St. PETER'S ENGINEERING COLLEGE

UGC - AUTONOMOUS

Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programe Accredited (EEE, CSE, ECE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We declare that a Project entitled "**IDENTIFICATION OF CARDIOVASCULAR DISEASE IN ECG IMAGE**" is an Original Work submitted by the following group members who have actively contributed and submitted in partial fulfilment for the award of degree in **"Bachelor of Technology in Computer Science and Engineering",** at **St. Peter's Engineering College**, Hyderabad, and this project work has not been submitted by me to any other college or university for the award of any kind of degree.

Date Submitted:

Name	**Roll Number**	**Signature**
P.Hariharan	**21BK5A0516**	
T.Bhavyasree	**20BK1A05G6**	
K.Shanmukh	**20BK1A05H9**	

ABSTRACT

Cardiovascular diseases (CVDs) stand as a significant global health concern, necessitating timely detection and intervention for optimal patient outcomes. Leveraging advancements in machine learning and medical imaging, this project proposes a solution to predict the presence of CVDs from electrocardiogram (ECG) images. The system encompasses various modules, including data preprocessing, feature extraction, model development, and evaluation, to automate the analysis of ECG images. Through the integration of advanced image processing techniques and machine learning algorithms, the system aims to provide healthcare professionals with a reliable tool for early identification and intervention in CVD cases.

The proposed solution entails preprocessing ECG images to enhance clarity and extract relevant features indicative of cardiovascular health. These features are then utilized to train a machine learning model capable of accurately predicting the presence of CVDs. By deploying the trained model within a user-friendly web application interface, healthcare practitioners can conveniently upload ECG images for automated analysis. This system aims to streamline the diagnostic process, empowering healthcare professionals with an efficient and accurate tool for early detection and intervention in cardiovascular diseases, thereby improving patient outcomes and reducing healthcare burdens associated with CVDs.

TABLE OF CONTENTS

LIST OF FIGURES

1.Introduction

Cardiovascular diseases (CVDs) represent a formidable challenge in global healthcare, responsible for a significant portion of morbidity and mortality worldwide. Timely diagnosis and intervention are paramount for effectively managing CVDs and reducing associated health risks. In this context, the use of electrocardiogram (ECG) imaging presents a promising avenue for early detection and assessment of cardiovascular health. This project endeavors to harness the power of machine learning and image processing techniques to develop a system capable of predicting the presence of CVDs from ECG images with high accuracy.

The proposed system aims to address the limitations of traditional diagnostic methods by automating the analysis of ECG images, thereby enabling healthcare professionals to make informed decisions swiftly. By leveraging advanced algorithms for data preprocessing, feature extraction, and machine learning model development, the system seeks to extract meaningful insights from ECG images, facilitating the early identification of cardiovascular abnormalities. Through the seamless integration of cutting-edge technology and medical expertise, this project endeavors to provide a reliable and efficient tool for enhancing cardiovascular health assessment and improving patient outcomes.

The objective of this project is to develop an automated system for predicting cardiovascular diseases (CVDs) from electrocardiogram (ECG) images using machine learning and image processing techniques. The primary aim is to create a reliable tool that can accurately analyze ECG images, extract relevant features indicative of cardiovascular health, and predict the presence of CVDs with high accuracy. By integrating advanced algorithms for data preprocessing, feature extraction, and model development, the system seeks to streamline the diagnostic process, enabling early detection and intervention in CVD cases. Ultimately, the project aims to improve patient outcomes by providing healthcare professionals with an efficient and effective tool for cardiovascular health assessment.

2. LITERATURE SURVEY

Automated Diagnosis of Cardiovascular Diseases Using ECG Images:

- Various studies have explored the use of machine learning techniques for automated diagnosis of cardiovascular diseases from ECG images. Researchers have investigated different feature extraction methods, including wavelet transforms and convolutional neural networks (CNNs), to capture relevant patterns indicative of cardiac abnormalities.

Image Processing Techniques for ECG Preprocessing:

- Literature also covers a range of image processing techniques tailored specifically for preprocessing ECG images. These techniques include noise reduction, baseline correction, and artifact removal, aimed at enhancing the quality and clarity of ECG signals prior to feature extraction and analysis.

Feature Extraction Methods for Cardiovascular Disease Prediction:

- Several studies have compared and evaluated various feature extraction methods for predicting cardiovascular diseases from ECG images. Techniques such as principal component analysis (PCA), discrete wavelet transform (DWT), and morphological analysis have been explored to extract discriminative features from ECG signals.

Machine Learning Models for Disease Prediction:

- Researchers have employed a variety of machine learning models, including support vector machines (SVM), random forests, and deep learning architectures, to predict the presence of cardiovascular diseases from ECG images. Comparative studies have assessed the performance of these models in terms of accuracy, sensitivity, and specificity.

2.1 EXISTING SYSTEM:

Identification of cardiovascular disease (CVD) in electrocardiogram (ECG) images, diagnosis relied heavily on visual interpretation by trained medical professionals. This process involved examining paper printouts of ECG tracings, which were produced by the ECG machine, and analyzing various waveform patterns and intervals.

Here's an overview of the traditional process:

- Recording ECGs
- Printing ECG Tracings
- Visual Analysis
- Manual Measurement
- Interpretation and Diagnosis

2.2 PROPOSED SYSTEM:

Proposed solution involves developing a system that uses machine learning to analyze ECG images and predict cardiovascular diseases. ECG images will be preprocessed to enhance clarity, then divided into sections for feature extraction. Key features will be extracted using advanced techniques, and a machine learning model will be trained on these features. The model will be deployed through a user-friendly web application, allowing users to upload ECG images for analysis. By automating the analysis process, our system aims to provide healthcare professionals with a reliable tool for early detection and intervention in cardiovascular diseases.

3. ANALYSIS

3.1 INTRODUCTION

SYSTEM STUDY FEASIBILITY STUDY

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

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

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

TECHNICAL FEASIBILITY

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

SOCIAL FEASIBILITY

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

3.2 HARDWARE REQUIREMENTS:

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system does and not how it should be implemented.

• Processor	-	i3 or above
• Speed	-	1.1 GHz
• Ram	-	6GB
• Hard Disk	-	20 GB

3.3 SOFTWARE REQUIREMENTS:

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

• Operating System	-	Windows / MacOs
• Coding Language	-	Python
• IDE	-	VsCode

3.4 FUNCTIONAL REQUIREMENTS:

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behaviour, The proposed method comprised of three primary steps. Detection of moving regions from the video sequence, extraction of HOG features of moving regions, and utilizing SVM to classify the moving regions. Similarly, Qixiang presented an Error Correcting Output Code (ECOC) based on manifold clustering strategy for human detection. The proposed technique worked efficiently with multi-view and multi-posture problems. Dewei introduced an online expectation-maximization (EM) algorithm in order to estimate foreground and A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behaviour, The proposed method comprised of three primary steps. Detection of moving regions from the video sequence, extraction of HOG features of moving regions, and utilizing SVM to classify the moving regions. Similarly, Qixiang presented an Error Correcting Output Code (ECOC) based on manifold clustering strategy for human detection. The proposed technique worked efficiently with multi-view and multi-posture problems. Dewei introduced an online expectation-maximization (EM) algorithm in order to estimate foreground and background. Later, the human samples are cropped from the estimated foreground for HOG feature extraction.

3.5 NON-FUNCTIONAL REQUIREMENTS:

The major non-functional Requirements of the system are as follows

Usability The system is designed with completely automated process hence there is no or less user intervention.

Reliability The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using python is more reliable.

Performance This system is developing in the high-level languages and using the advanced frontend and backend technologies it will give response to the end user on client system with in very less time.

Supportability The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is built into the system.

Implementation The system is implemented in web environment using Django framework. The server is used as the web server and windows xp professional is used as the platform. Interface the user interface is based on Django provides web application.

4. DESIGN

4.1 GENERAL:

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

4.1 UML DIAGRAMS :

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

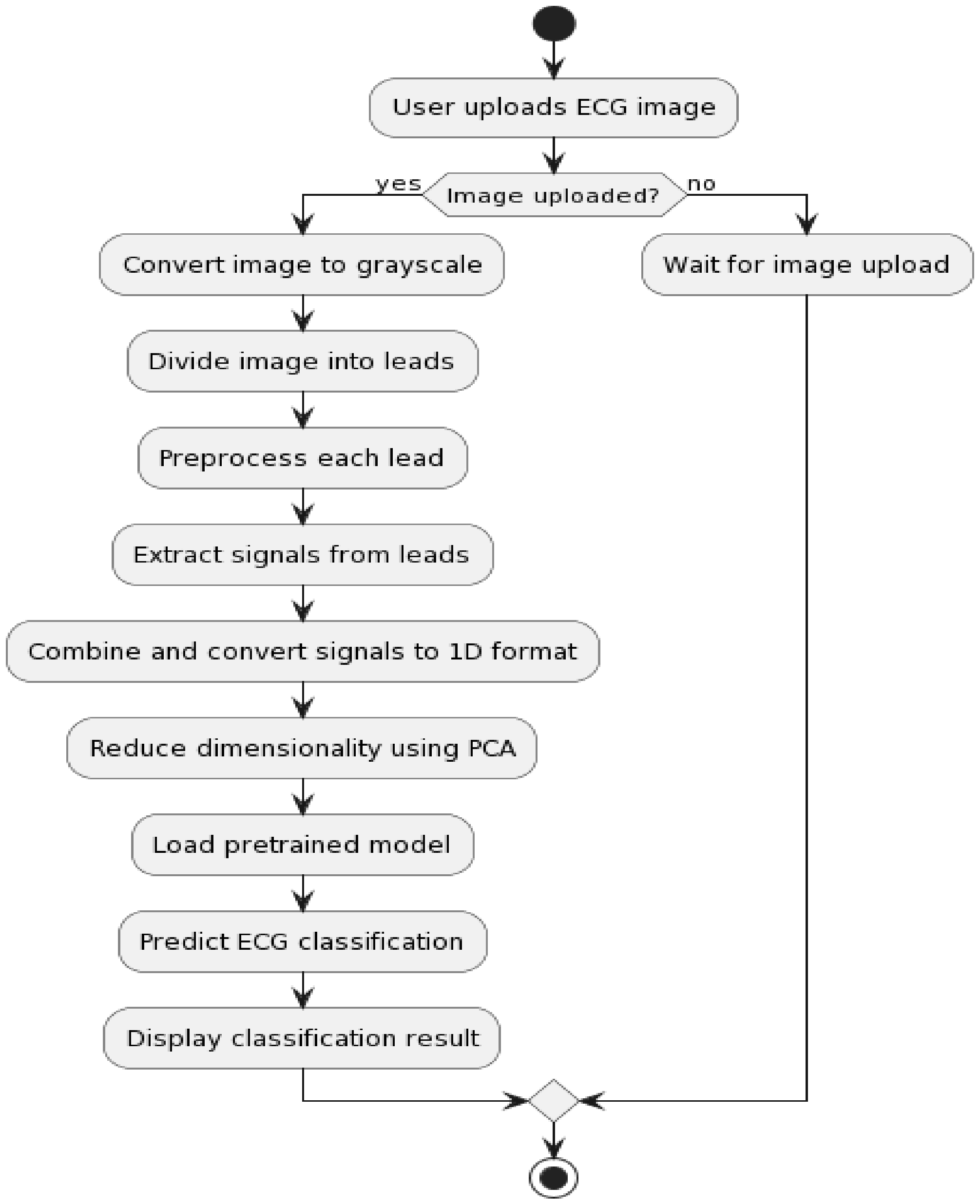
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

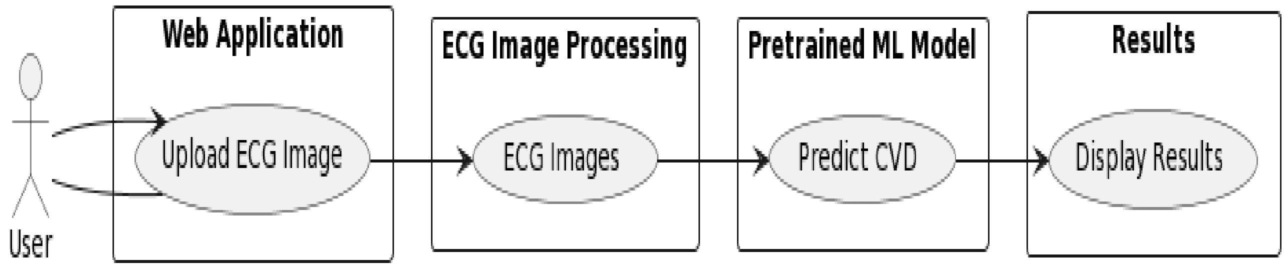
- **ACTIVITY DIAGRAM**

An activity diagram is a type of Unified Modeling Language (UML) flowchart that shows the flow from one activity to another in a system or process. It's used to describe the different dynamic aspects of a system and is referred to as a 'behavior diagram' because it describes what should happen in the modeled system.


User uploads ECG image
yes
Image uploaded?
no
Convert image to grayscale
Wait for image upload
Divide image into leads
Preprocess each lead
Extract signals from leads
Combine and convert signals to 1D format
Reduce dimensionality using PCA
Load pretrained model
Predict ECG classification
Display classification result

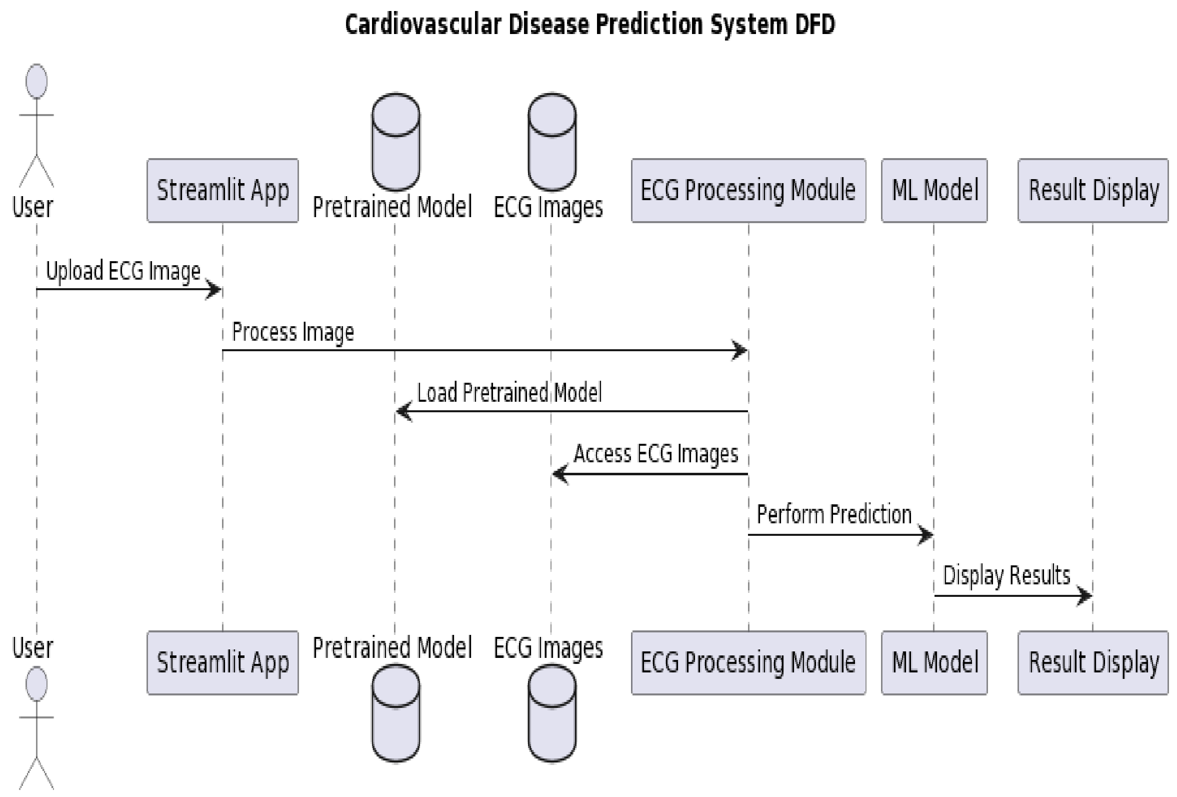
- **USE CASE DIAGRAM**

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



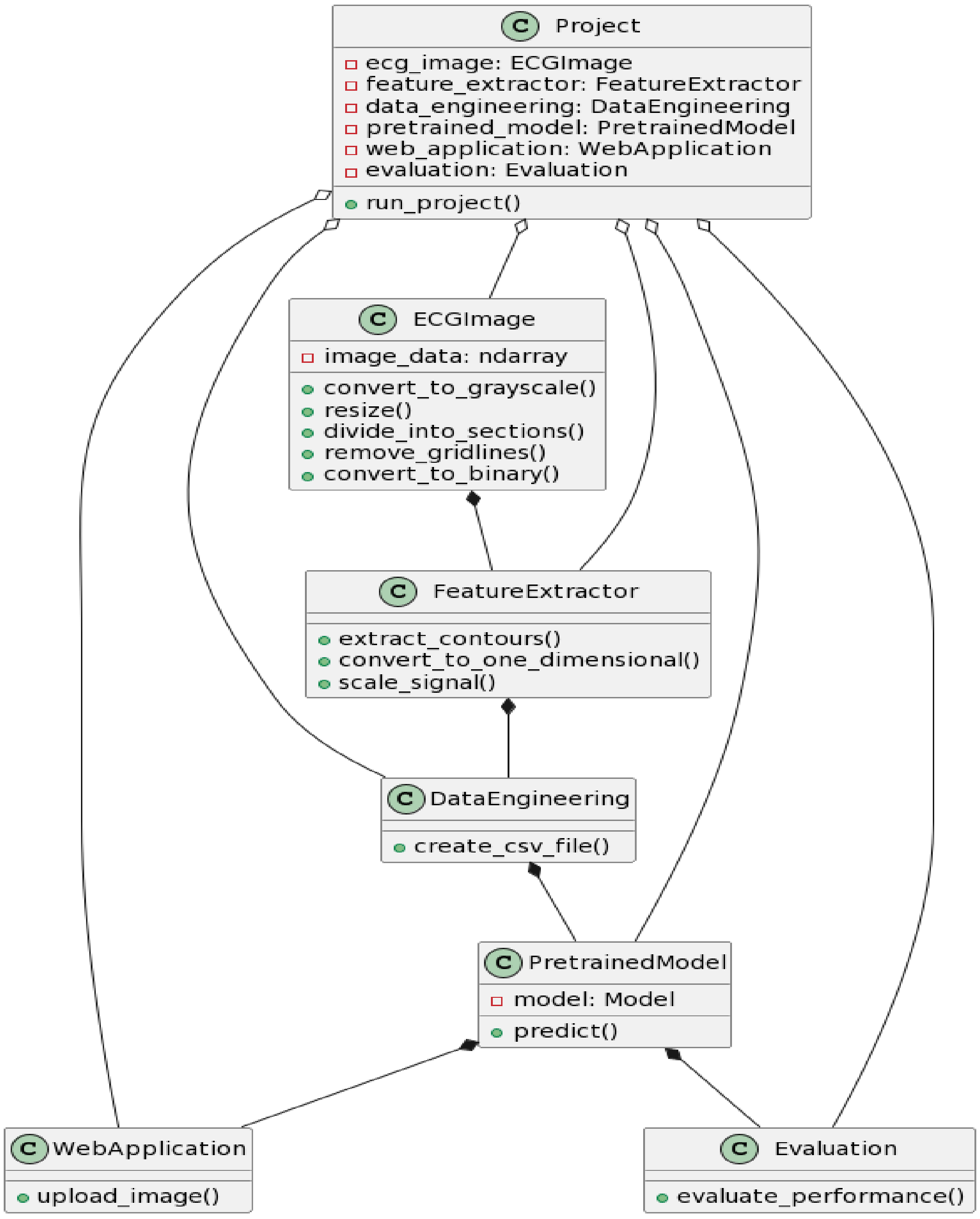
- **DATAFLOW DIAGRAM**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.



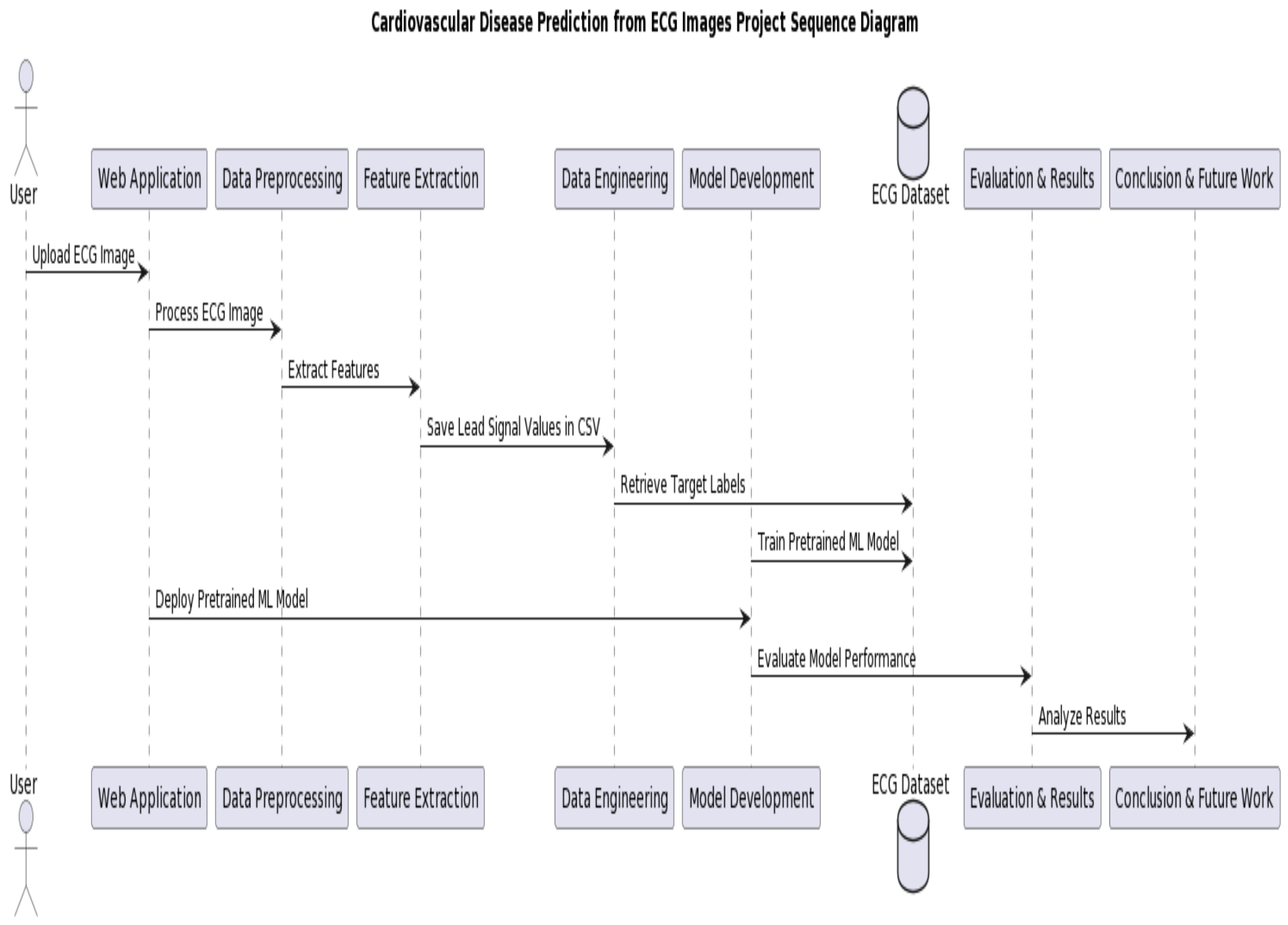
• CLASS DIAGRAM:

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



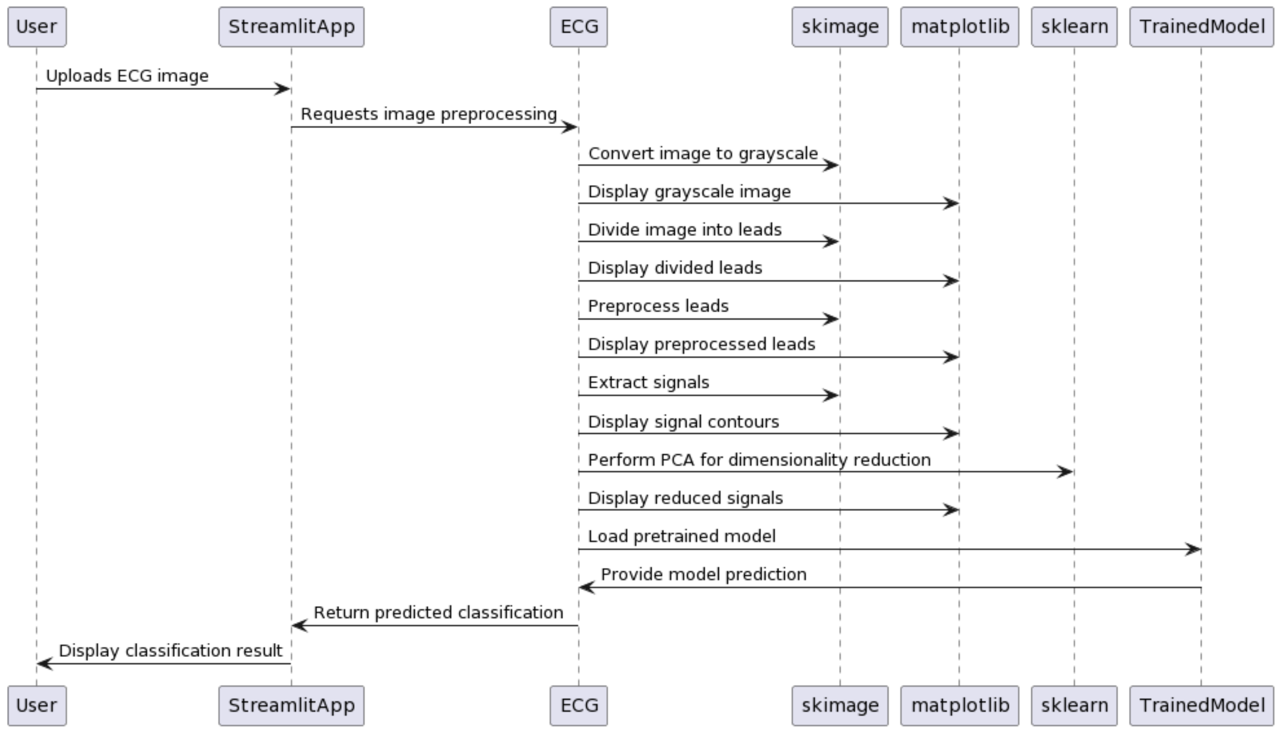
- **SEQUENCE DIAGRAM:**

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



• COLLABRATION DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



5. IMPLEMENTATION

5.1 MODULES:

Data Acquisition Module:

This module is responsible for collecting ECG images from various sources such as medical databases or research repositories.

Data Preprocessing Module:

Conversion to grayscale

Resizing

Division into 12 sections

Gridline removal

Conversion to binary images

Feature Extraction Module:

Contour technique for signal extraction

Conversion of waveform to one-dimensional signal

Data Engineering Module:

Scaling of signal values

CSV file creation for storing lead signal values and corresponding target labels

Model Development Module:

Training of machine learning models (e.g., decision trees, random forests, SVM, neural networks)

Ensemble technique implementation for model stacking

Web Application Interface Module:

User-friendly interface for uploading ECG images

Integration with the preprocessing and model deployment modules

Model Deployment Module:

Deployment of the trained machine learning model for real-time prediction

Integration with the web application interface for seamless user experience

Evaluation and Results Module:

Performance evaluation of the machine learning models using metrics like accuracy, precision, recall, and F1-score

Presentation of results to stakeholders

5.2 SOFTWARE ENVIRONMENT :

5.2.1 WHAT IS PYTHON :

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

5.2.2 ADVANTAGES OF PYTHON :

Let's see how Python dominates over other languages.

1. Extensive Libraries:

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible:

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable:

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity:

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities:

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

6. Readable:

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

7. Object-Oriented:

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

8. Free and Open-Source:

Like we said earlier, Python is **freely available.** But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

9. Portable:

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

10. Interpreted:

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

5.2.3 ADVANTAGES OF PYTHON OVER OTHER LANGUAGES :

1. Less Coding:

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable:

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

3.Python is for Everyone:

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

5.2.4 DISADVANTAGES OF PYTHON :

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations:

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers:

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions:

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers:

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple:

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

5.2.5 History of Python :

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde&Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voorWiskundeen Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because

I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

5.3 MACHINE LEARNING

5.3.1 WHAT IS MACHINE LEARNING :

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

5.3.2 CATEGORIES OF MACHINE LEANING :

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

5.3.3 NEED FOR MACHINE LEARNING :

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

5.3.4 CHALLENGES IN MACHINES LEARNING :

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

5.3.5 APPLICATIONS OF MACHINES LEARNING :

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition

- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term **"Machine Learning"** in 1959 and defined it as a **"Field of study that gives computers the capability to learn without being explicitly programmed".**

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus:

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics:

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data.
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python:

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML .It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning:

- **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label) –** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training –** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction –** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning:

- **Supervised Learning –** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning –** This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

ADVANTAGES OF MACHINE LEARNING :

1. Easily identifies trends and patterns:

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation):

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement :

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data:

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications:

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

DISADVANTAGES OF MACHINE LEARNING :

1. Data Acquisition:

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources:

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results:

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility:

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time.

PYTHON DEVELOPMENT STEPS :

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit

Purpose :

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project :

- **Tensorflow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

- **Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

- **Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

- **Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

- **Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive** − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac :

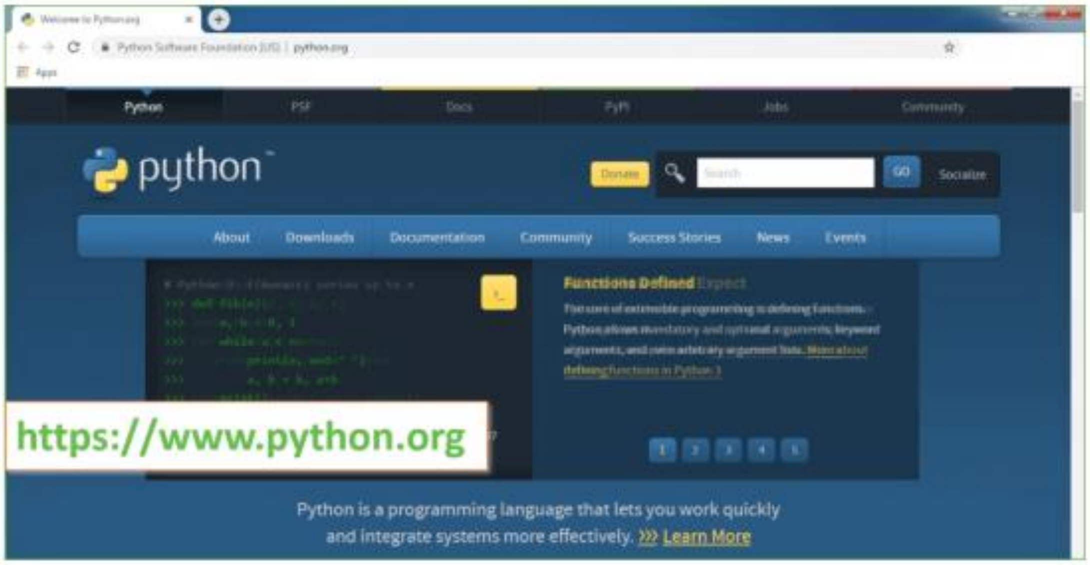
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

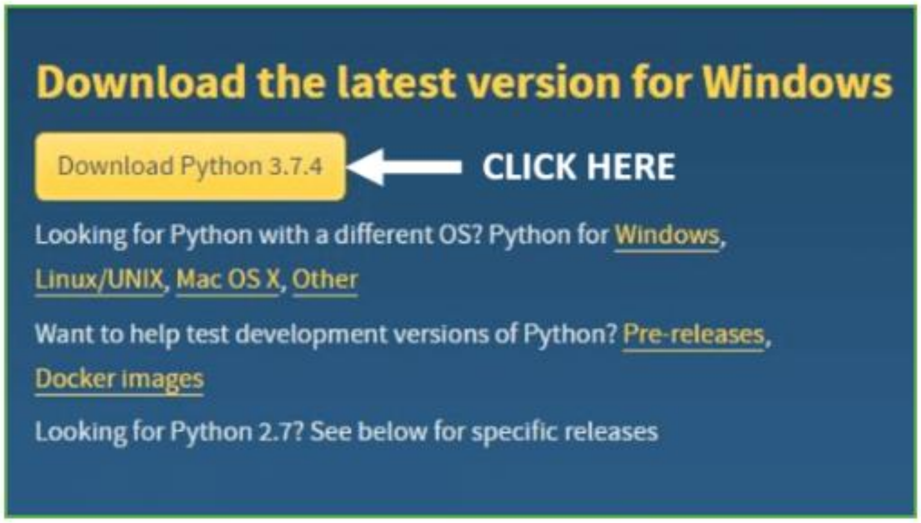
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

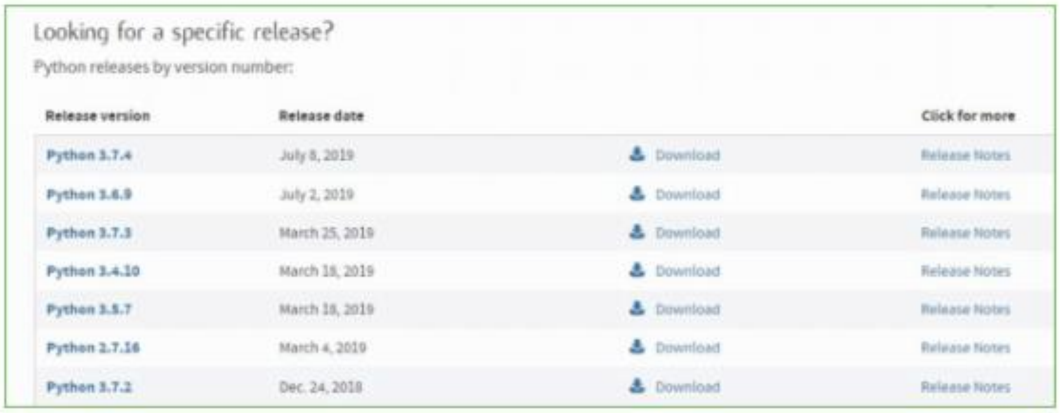

python
About
Downloads
Documentation
Community
Success Stories
News
Events
https://www.python.org
Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.


Download the latest version for Windows
Download Python 3.7.4
CLICK HERE
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other
Want to help test development versions of Python? Pre-releases, Docker images
Looking for Python 2.7? See below for specific releases

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



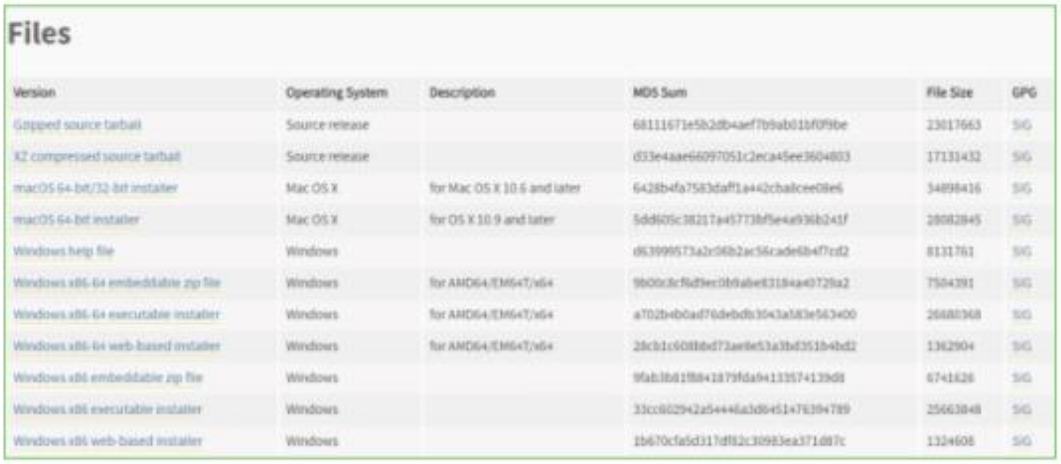
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

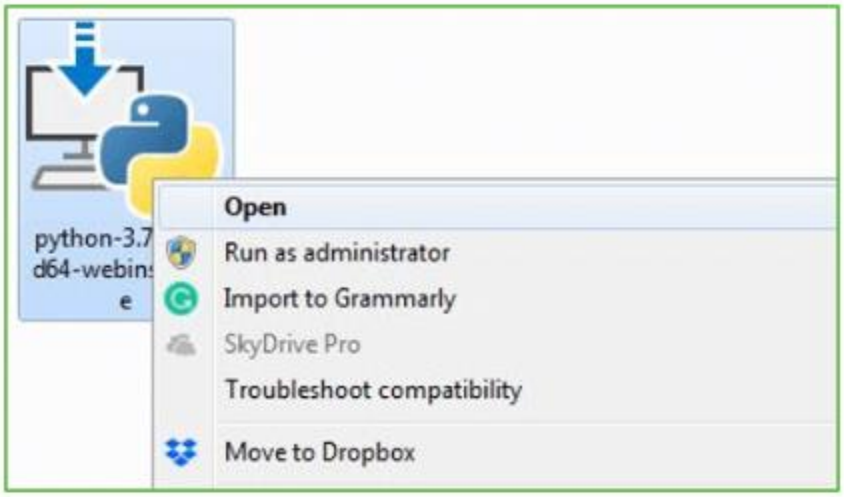
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

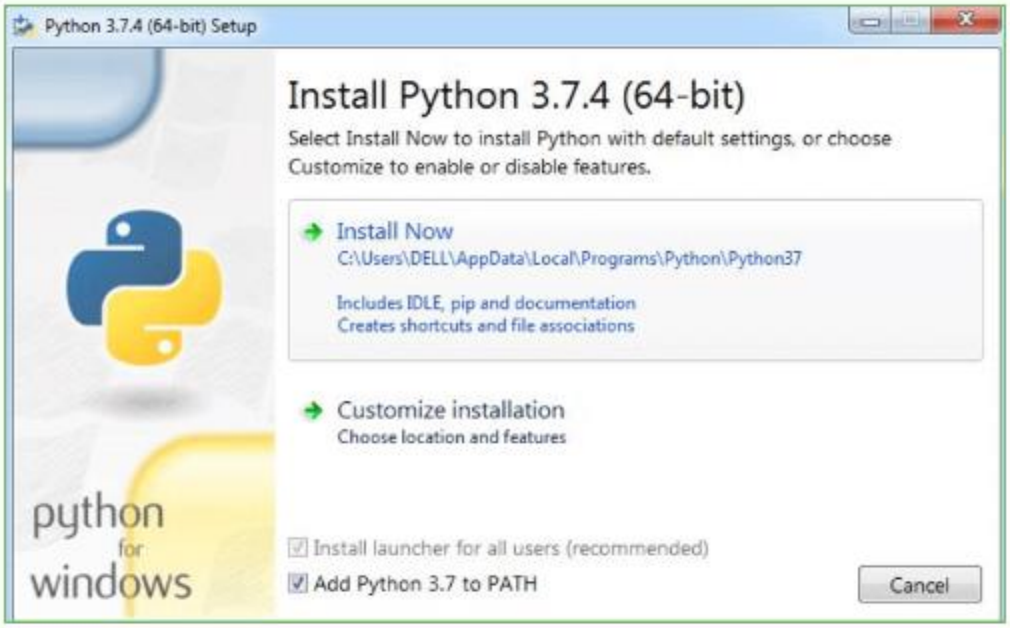
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

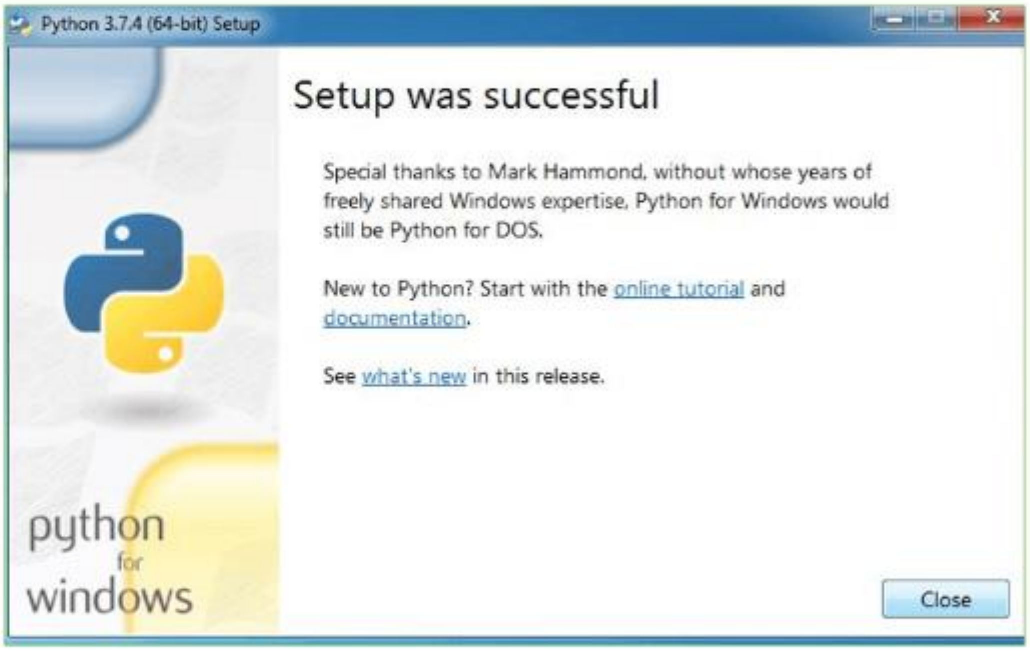
Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

[✓] Install launcher for all users (recommended)
[✓] Add Python 3.7 to PATH

Step 3: Click on Install NOW After the installation is successful. Click on Close.



Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

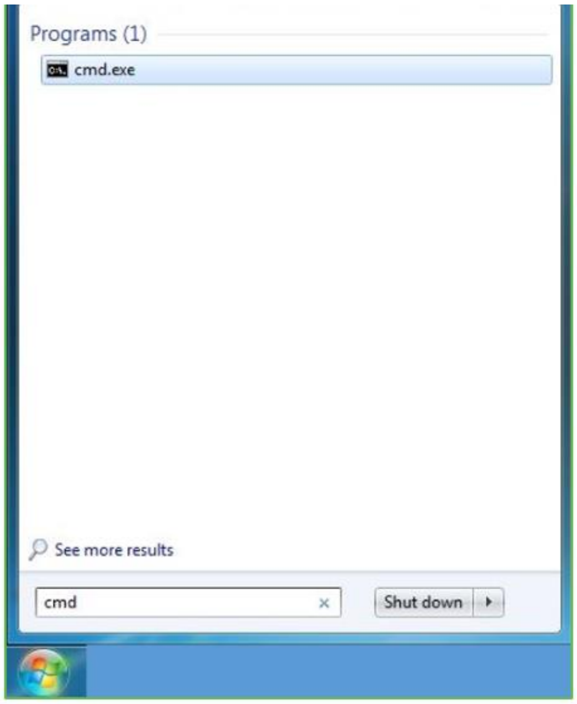
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

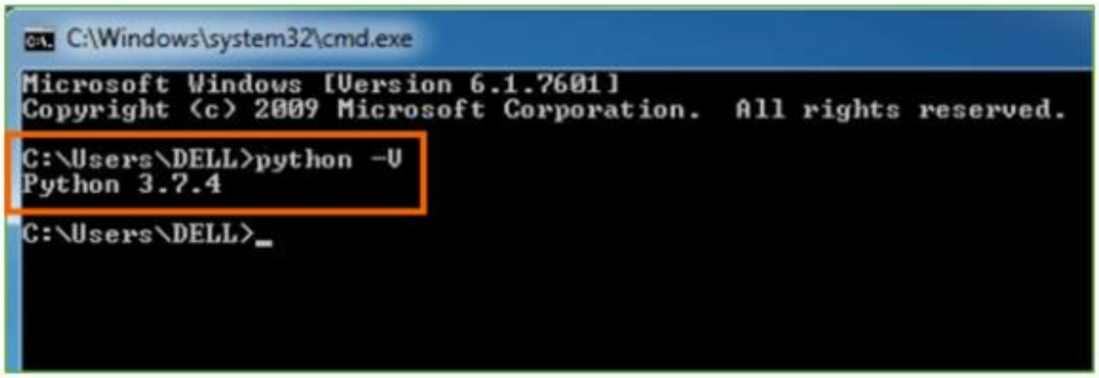
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

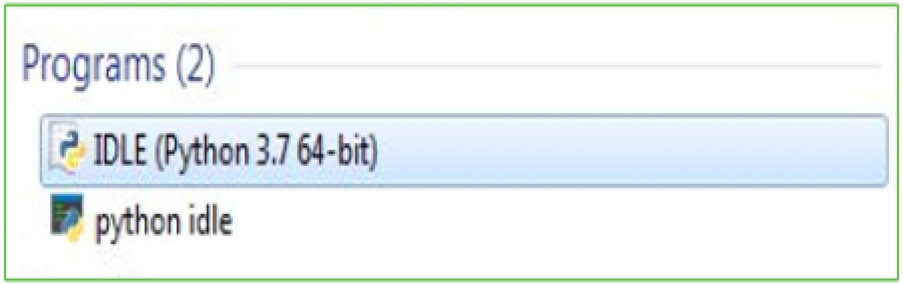
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

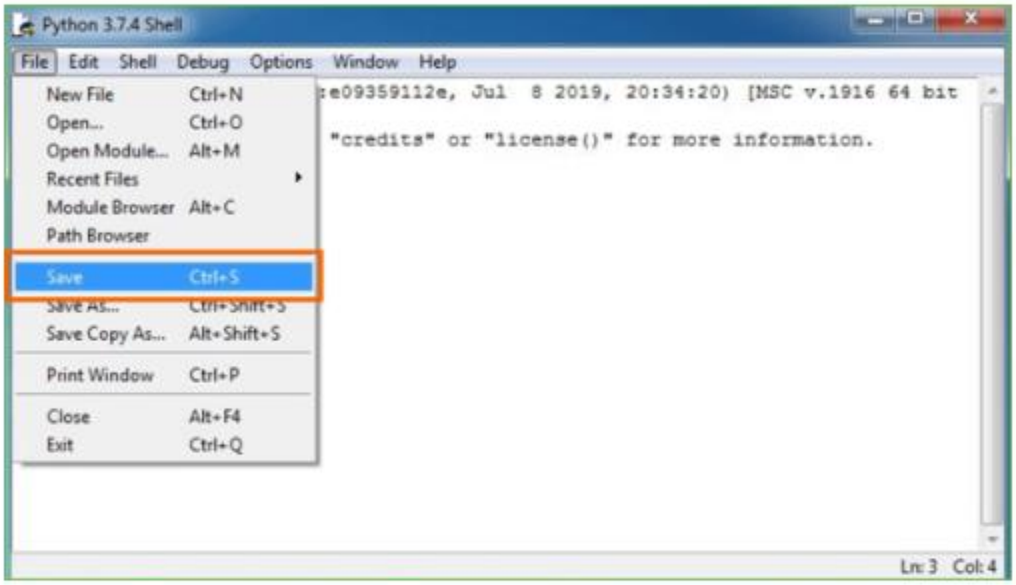
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File** > **Click on Save**



```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit

 "credits" or "license()" for more information.
```

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

5.4 SOURCE CODE

final_app.py:-

```
import streamlit as st
from Ecg import  ECG
#intialize ecg object
ecg = ECG()
#get the uploaded image
uploaded_file = st.file_uploader("Choose a file")

if uploaded_file is not None:
 """#### **UPLOADED IMAGE**"""
 # call the getimage method
 ecg_user_image_read = ecg.getImage(uploaded_file)
 #show the image
 st.image(ecg_user_image_read)

 """#### **GRAY SCALE IMAGE**"""
 #call the convert Grayscale image method
 ecg_user_gray_image_read = ecg.GrayImgae(ecg_user_image_read)
  #create Streamlit Expander for Gray Scale
 my_expander = st.expander(label='Gray SCALE IMAGE')
 with my_expander:
   st.image(ecg_user_gray_image_read)
  """#### **DIVIDING LEADS**"""
  #call the Divide leads method
 dividing_leads=ecg.DividingLeads(ecg_user_image_read)

 #streamlit expander for dividing leads
 my_expander1 = st.expander(label='DIVIDING LEAD')
 with my_expander1:
   st.image('Leads_1-12_figure.png')
   st.image('Long_Lead_13_figure.png')
  """#### **PREPROCESSED LEADS**"""
 #call the preprocessed leads method
 ecg_preprocessed_leads = ecg.PreprocessingLeads(dividing_leads)
```

```
  #streamlit expander for preprocessed leads
  my_expander2 = st.expander(label='PREPROCESSED LEAD')
  with my_expander2:
    st.image('Preprossed_Leads_1-12_figure.png')
    st.image('Preprossed_Leads_13_figure.png')
   """#### **EXTRACTING SIGNALS(1-12)**"""
  #call the sognal extraction method
  ec_signal_extraction = ecg.SignalExtraction_Scaling(dividing_leads)
  my_expander3 = st.expander(label='CONOTUR LEADS')
  with my_expander3:
    st.image('Contour_Leads_1-12_figure.png')
   """#### **CONVERTING TO 1D SIGNAL**"""
  #call the combine and conver to 1D signal method
  ecg_1dsignal = ecg.CombineConvert1Dsignal()
  my_expander4 = st.expander(label='1D Signals')
  with my_expander4:
    st.write(ecg_1dsignal)

  """#### **PERFORM DIMENSINALITY REDUCTION**"""
  #call the dimensinality reduction funciton
  ecg_final = ecg.DimensionalReduciton(ecg_1dsignal)
  my_expander4 = st.expander(label='Dimensional Reduction')
  with my_expander4:
    st.write(ecg_final)
   """#### **PASS TO PRETRAINED ML MODEL FOR PREDICTION**"""
  #call the Pretrainsed ML model for prediction
  ecg_model=ecg.ModelLoad_predict(ecg_final)
  my_expander5 = st.expander(label='PREDICTION')
  with my_expander5:
    st.write(ecg_model)
```

Ecg.py:-

```
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 numpy import asarray
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
from sklearn import linear_model, tree, ensemble
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression

class ECG:
  def  getImage(self,image):
      """
      this functions gets user image
      return: user image
      """
      image=imread(image)
      return image

  def GrayImgae(self,image):
      """
      This funciton converts the user image to Gray Scale
      return: Gray scale Image
      """
      image_gray = color.rgb2gray(image)
      image_gray=resize(image_gray,(1572,2213))
      return image_gray

  def DividingLeads(self,image):
      """
      This Funciton Divides the Ecg image into 13 Leads including long lead. Bipolar limb leads(Leads1,2,3).
Augmented unipolar limb leads(aVR,aVF,aVL). Unipolar (+) chest leads(V1,V2,V3,V4,V5,V6)
```

```
        return : List containing all 13 leads divided
        """
        Lead_1 = image[300:600, 150:643] # Lead 1
        Lead_2 = image[300:600, 646:1135] # Lead aVR
        Lead_3 = image[300:600, 1140:1625] # Lead V1
        Lead_4 = image[300:600, 1630:2125] # Lead V4
        Lead_5 = image[600:900, 150:643] #Lead 2
        Lead_6 = image[600:900, 646:1135] # Lead aVL
        Lead_7 = image[600:900, 1140:1625] # Lead V2
        Lead_8 = image[600:900, 1630:2125] #Lead V5
        Lead_9 = image[900:1200, 150:643] # Lead 3
        Lead_10 = image[900:1200, 646:1135] # Lead aVF
        Lead_11 = image[900:1200, 1140:1625] # Lead V3
        Lead_12 = image[900:1200, 1630:2125] # Lead V6
        Lead_13 = image[1250:1480, 150:2125] # Long Lead

        #All Leads in a list

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,Lea
d_13]
        fig , ax = plt.subplots(4,3)
        fig.set_size_inches(10, 10)
        x_counter=0
        y_counter=0

        #Create 12 Lead plot using Matplotlib subplot

        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

        #save the image
        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')

        return Leads

    def PreprocessingLeads(self,Leads):
        """
        This Function Performs preprocessing to on the extracted 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=1)
            #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')

        #plotting lead 13
        fig3 , ax3 = plt.subplots()
        fig3.set_size_inches(10, 10)
        #converting to gray scale
        grayscale = color.rgb2gray(Leads[-1])
        #smoothing image
        blurred_image = gaussian(grayscale, sigma=1)
        #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')
        fig3.savefig('Preprossed_Leads_13_figure.png')



    def SignalExtraction_Scaling(self,Leads):
        """
        This Function Performs Signal Extraction using various steps,techniques: conver to grayscale, apply gaussian
      filter, thresholding, perform contouring to extract signal image and then save the image as 1D signal
        """
        fig4 , ax4 = plt.subplots(4,3)
        #fig4.set_size_inches(10, 10)
```

```
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=0.7)
    #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))
    #finding contours
    contours = measure.find_contours(binary_global,0.8)
    contours_shape = sorted([x.shape for x in contours])[::-1][0:1]
    for contour in contours:
        if contour.shape in contours_shape:
            test = resize(contour, (255, 2))
    if (x+1)%3==0:
        ax4[x_counter][y_counter].invert_yaxis()
        ax4[x_counter][y_counter].plot(test[:, 1], test[:, 0],linewidth=1,color='black')
        ax4[x_counter][y_counter].axis('image')
        ax4[x_counter][y_counter].set_title("Contour {} image".format(x+1))
        x_counter+=1
        y_counter=0
    else:
        ax4[x_counter][y_counter].invert_yaxis()
        ax4[x_counter][y_counter].plot(test[:, 1], test[:, 0],linewidth=1,color='black')
        ax4[x_counter][y_counter].axis('image')
        ax4[x_counter][y_counter].set_title("Contour {} image".format(x+1))
        y_counter+=1

    #scaling the data and testing
    lead_no=x
```

```
            scaler = MinMaxScaler()
            fit_transform_data = scaler.fit_transform(test)
            Normalized_Scaled=pd.DataFrame(fit_transform_data[:,0], columns = ['X'])
            Normalized_Scaled=Normalized_Scaled.T
            #scaled_data to CSV
            if (os.path.isfile('scaled_data_1D_{lead_no}.csv'.format(lead_no=lead_no+1))):
                Normalized_Scaled.to_csv('Scaled_1DLead_{lead_no}.csv'.format(lead_no=lead_no+1),
mode='a',index=False)
            else:
                Normalized_Scaled.to_csv('Scaled_1DLead_{lead_no}.csv'.format(lead_no=lead_no+1),index=False)

        fig4.savefig('Contour_Leads_1-12_figure.png')



    def CombineConvert1Dsignal(self):
        """
        This function combines all 1D signals of 12 Leads into one FIle csv for model input.
        returns the final dataframe
        """
        #first read the Lead1 1D signal
        test_final=pd.read_csv('Scaled_1DLead_1.csv')
        location= os.getcwd()
        print(location)
        #loop over all the 11 remaining leads and combine as one dataset using pandas concat
        for files in natsorted(os.listdir(location)):
            if files.endswith(".csv"):
                if files!='Scaled_1DLead_1.csv':
                    df=pd.read_csv('{}'.format(files))
                    test_final=pd.concat([test_final,df],axis=1,ignore_index=True)

        return test_final

    def DimensionalReduciton(self,test_final):
        """
        This function reduces the dimensinality of the 1D signal using PCA
        returns the final dataframe
        """
        #first load the trained pca
        pca_loaded_model = joblib.load('PCA_ECG (1).pkl')
        result = pca_loaded_model.transform(test_final)
        final_df = pd.DataFrame(result)
        return final_df

    def ModelLoad_predict(self,final_df):
        """
        This Function Loads the pretrained model and perfrom ECG classification
        return the classification Type.
        """
        loaded_model = joblib.load('Heart_Disease_Prediction_using_ECG (4).pkl')
        result = loaded_model.predict(final_df)
        if result[0] == 1:
            return "You ECG corresponds to Myocardial Infarction"
        elif result[0] == 0:
            return "You ECG corresponds to Abnormal Heartbeat"
        elif result[0] == 2:
            return "Your ECG is Normal"
        else:
            return "You ECG corresponds to History of Myocardial Infarction"
```

6. TESTING & VALIDATION

6.1 SYSTEM TEST

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

6.1.1 TYPES OF TESTS

UNIT TESTING:

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

INTEGRATION TESTING:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

FUNCTIONAL TEST:

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

Functional testing is centered on the following items:

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

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

SYSTEM TEST:

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

WHITE BOX TESTING:

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

BLACK BOX TESTING:

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

UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

6.1.2 TEST STRATEGY AND APPROACH

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

Test objectives

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

Features to be tested

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

6.2 Integration Testing

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

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

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

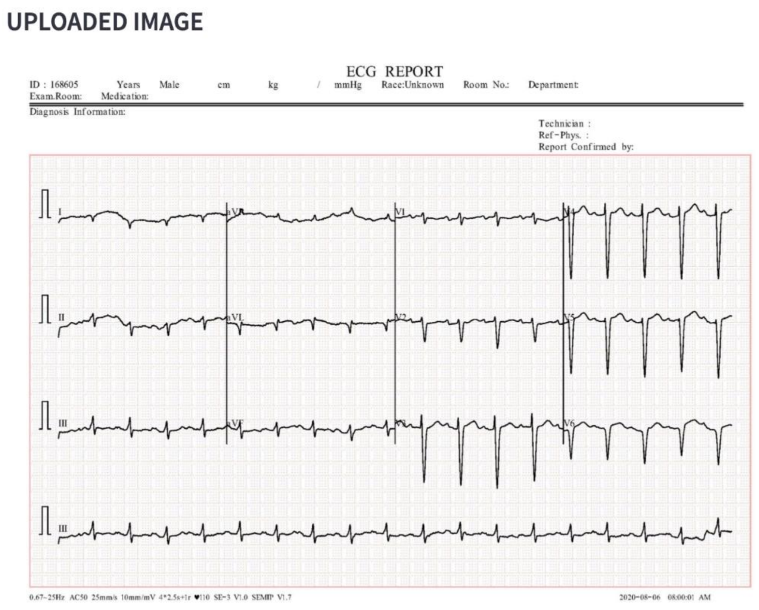
Acceptance Testing

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

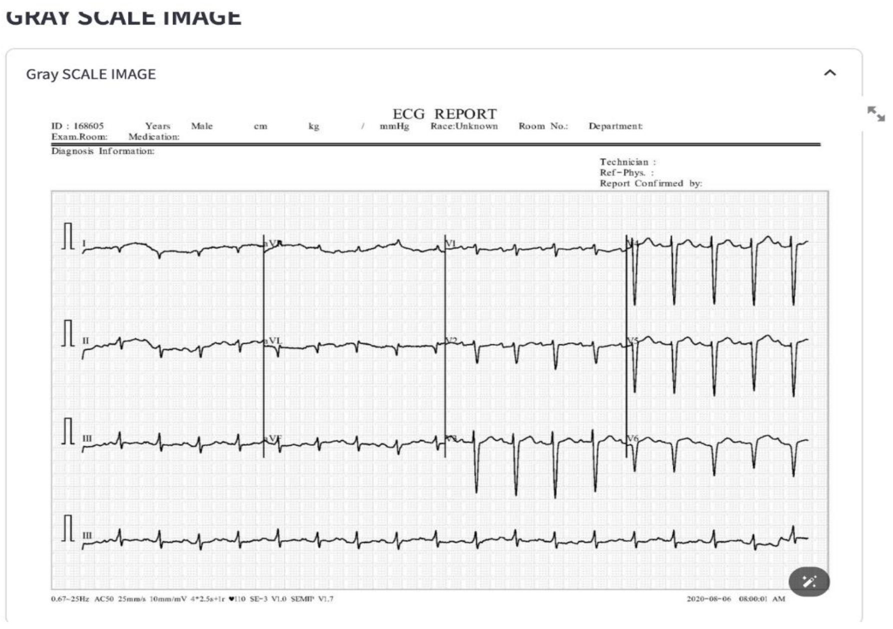
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9.RESULTS

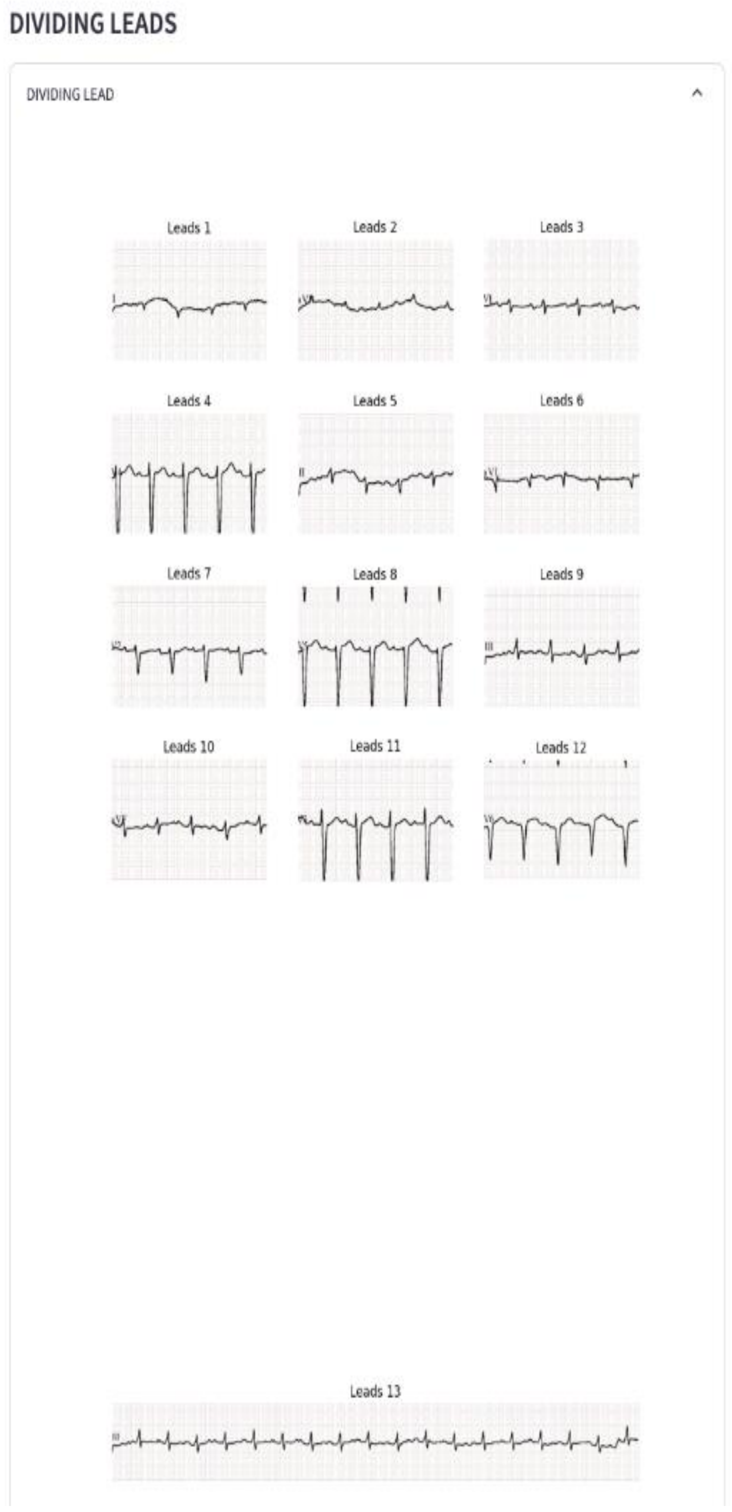
UPLOADED ECG IMAGE :



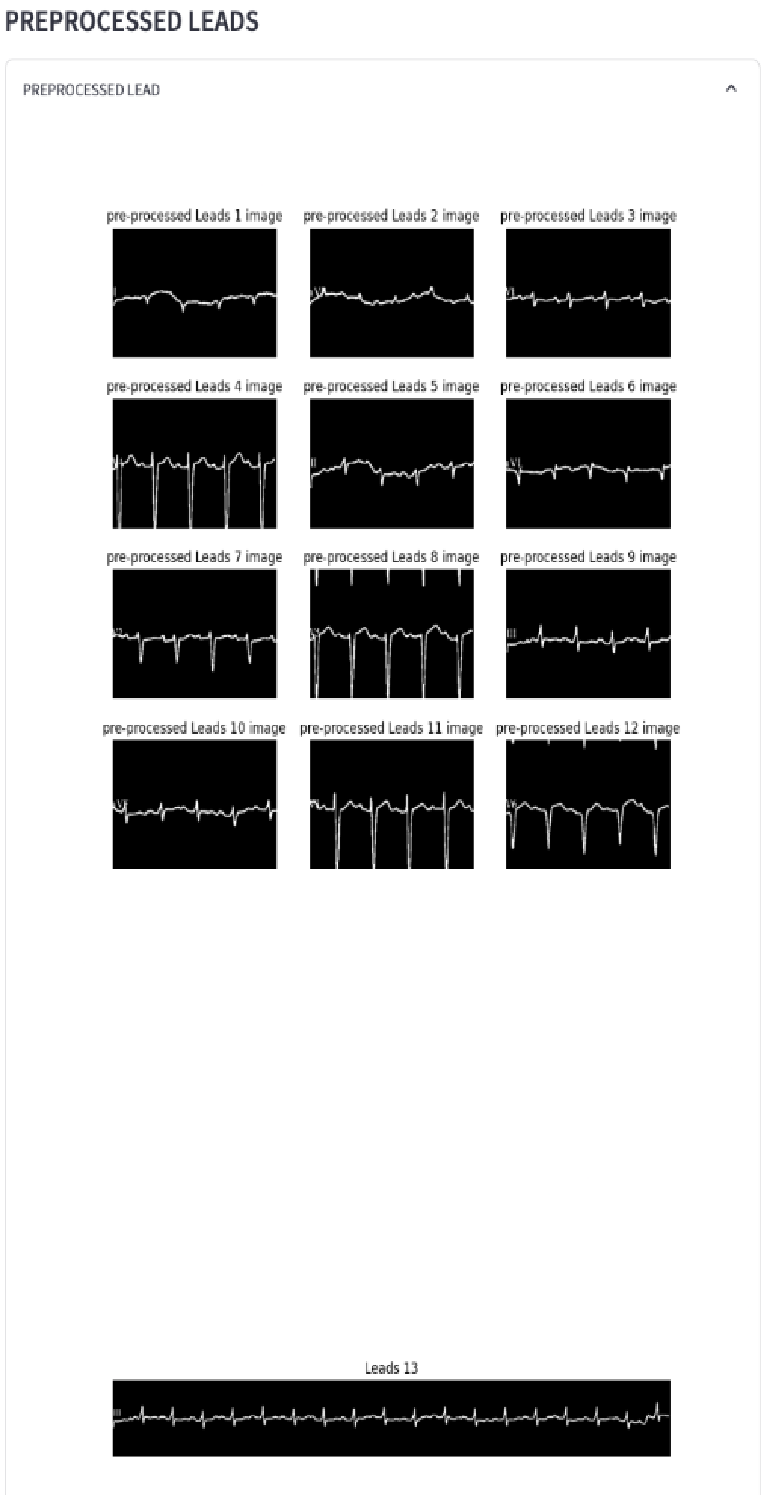
GRAY SCALE IMAGE :



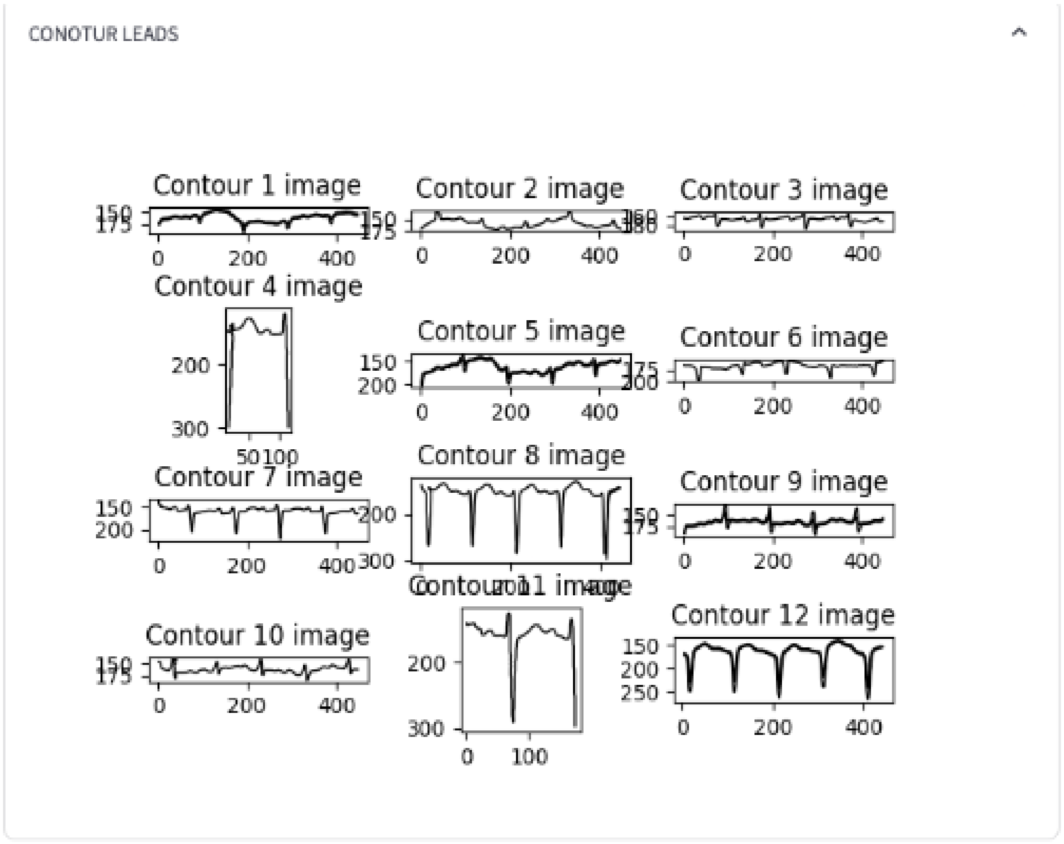
DIVIDING LEADS:



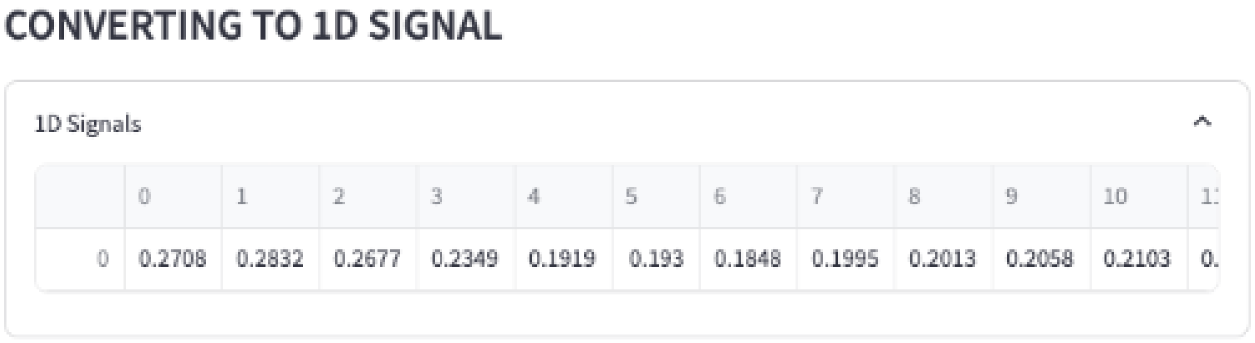
PREPROCESSED LEADS :


PREPROCESSED LEADS
PREPROCESSED LEAD
pre-processed Leads 1 image
pre-processed Leads 2 image
pre-processed Leads 3 image
pre-processed Leads 4 image
pre-processed Leads 5 image
pre-processed Leads 6 image
pre-processed Leads 7 image
pre-processed Leads 8 image
pre-processed Leads 9 image
pre-processed Leads 10 image
pre-processed Leads 11 image
pre-processed Leads 12 image
Leads 13

CONTOUR LEADS :



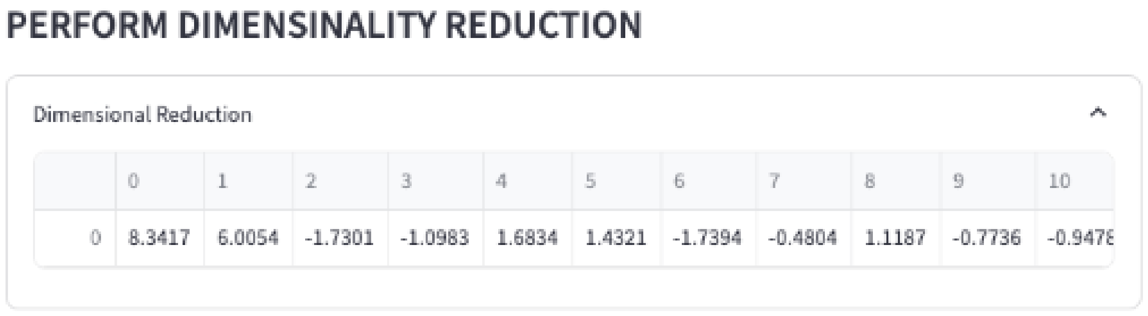
CONVERTING TO 1D SIGNALS :


CONVERTING TO 1D SIGNAL

1D Signals

	0	1	2	3	4	5	6	7	8	9	10	1:
0	0.2708	0.2832	0.2677	0.2349	0.1919	0.193	0.1848	0.1995	0.2013	0.2058	0.2103	0.

DIMENSIONALITY REDUCTION :

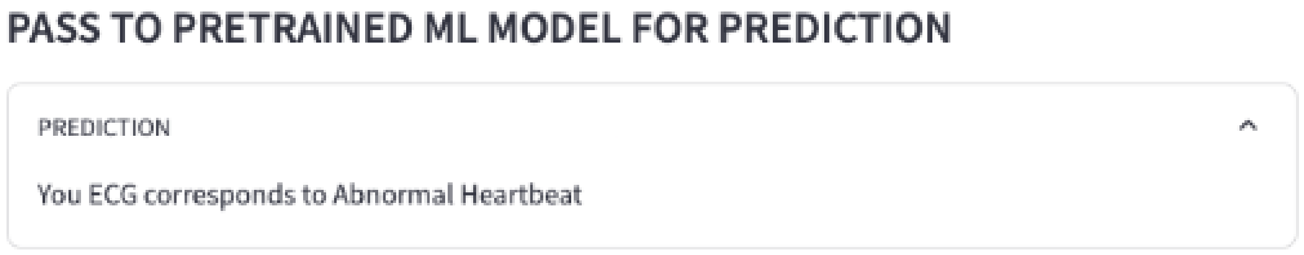


PERFORM DIMENSINALITY REDUCTION

Dimensional Reduction

	0	1	2	3	4	5	6	7	8	9	10
0	8.3417	6.0054	-1.7301	-1.0983	1.6834	1.4321	-1.7394	-0.4804	1.1187	-0.7736	-0.947

FINAL PREDICTION :



PASS TO PRETRAINED ML MODEL FOR PREDICTION

PREDICTION

You ECG corresponds to Abnormal Heartbeat

7. CONCLUSION

In conclusion, the development of an automated system for predicting cardiovascular diseases (CVDs) from electrocardiogram (ECG) images represents a significant advancement in healthcare technology. By leveraging machine learning and image processing techniques, we have successfully created a tool capable of analyzing ECG images and providing accurate predictions of CVD presence. This system streamlines the diagnostic process, enabling healthcare professionals to make timely interventions and improve patient outcomes.

Moving forward, further refinements and enhancements can be made to improve the system's performance and usability. This includes fine-tuning machine learning models, optimizing image preprocessing algorithms, and integrating additional clinical data sources for enhanced predictive accuracy. By continually iterating and improving upon our system, we can further empower healthcare professionals in their efforts to combat cardiovascular diseases and ultimately contribute to better overall public health.

8. REFERENCES

[1] World Health Organization (WHO), "Cardiovascular diseases," Jun. 11, 2021. Accessed: Dec. 27, 2021. [Online]. Available: https://www.who. int/health-topics/cardiovascular-diseases

[2] Government of Westren Australia, Department of Health, "Com- mon medical tests to diagnose heart conditions," Accessed: Dec. 29, 2021. [Online]. Available: https://www.healthywa.wa.gov.au/Articles/A_ E/Common-medical-tests-to-diagnose-heart-conditions

[3] M. Swathy and K. Saruladha, "A comparative study of classification and prediction of cardio-vascular diseases (CVD) using machine learning and deep learning techniques," ICT Exp., to be published, 2021. [Online].

Available: https://doi.org/10.1016/j.icte.2021.08.021

[4] R. R. Lopes et al., "Improving electrocardiogram-based detection of rare genetic heart disease using transfer learning: An application to phos- pholamban p.Arg14del mutation carriers," Comput. Biol. Med., vol. 131, 2021, Art. no. 104262. [Online]. Available: https://doi.org/10.1016/j. Compbiomed.2021.104262

[5] R. J. Martis, U. R. Acharya, and H. Adeli, "Current methods in electro- cardiogram characterization," Comput. Biol. Med., vol. 48, pp. 133–149, 2014. [Online]. Available: https://doi.org/10.1016/j.compbiomed.2014. 02.012