

Table of Contents

Name of the Topic	Page No
Chapter 1. Introduction	01
1.1. Aim	01
1.2. Objective	01
Chapter 2. System Analysis	04
2.1. Existing System and Disadvantages	04
2.2. Proposed System Description & Advantages	05
2.3. Software requirements & Specification	05
Chapter 3. Literature Survey	06
Chapter 4. Project Modules	07
Chapter 5. Overview of the Concepts	10
5.1 Python overview	10
5.2 History of python	10
5.3 Python features	11
5.4 Domain specification	12
5.5 Python project modules	15
5.6 Platform	17
5.7 Audio file overview	19
Chapter 6. Design	22
6.1. System Architecture	22
6.2. UML Diagrams	23
6.3 Data flow diagram	27
Chapter 7. Implementation	29
7.1. Installation Procedure	29
7.2. Code Description	36

Chapter 8. Testing	50
8.1 Testing Methods	50
8.2 Test	51
8.3 Types of Tests	52
8.4 System Test	53
Chapter 9. Output Screen	55
Chapter 10. Conclusion	63
Bibliography	64
Appendix A	

LIST OF FIGURES

Figure No.	Figure Name	Page No.
4.1	MFCC Block Diagram	09
5.4.1	Artificial Intelligence	13
5.4.2	Graph of AI vs ML vs DL	14
5.7.1	Sampling frequency	19
5.7.2	Graph of time and frequency	20
6.1	System architecture	22
6.2.1	Use case Diagram	24
6.2.2	Class Diagram	24
6.2.3	Activity diagram	25
6.2.4	Sequence diagram	26
6.3.1	Level 0 Data Flow Diagram	27
6.3.2	Level 1 Data Flow Diagram	28

LIST OF TABLES

Table No.	Table Name	Page No.
5.4	ML vs DL	14
8.1	Test Cases	51

Chapter – 1

Introduction

Cardiac diseases are leading cause of death in the worldwide and this situation is especially worse in developing countries because of shortage of medical professionals. Automated and early diagnosis of cardiac disorders can be a possible solution to prevent mortality in rural areas. Clinically physical examination of human heart using auscultations via stethoscope is an easy, efficient and computationally cheap method but requires trained medical experts.

Signal processing based heart sound analysis may be a valuable initiative for automated diagnosis of cardiac disorders without the help of professional doctors. Heart sound analysis maybe of great helps in primary health centre for early diagnosis and screening of cardiac disorders. Some efforts have been made in this direction where body auscultations are being analysed for early and automated diagnosis of diseases.

1.1 Aim

The aim of using machine learning and end-to-end deep learning for the detection of chronic heart failure (CHF) from heart sounds is to develop an automated, non-invasive, and accurate system that can assist in the early diagnosis and monitoring of CHF.

1.2 Objective

The primary objectives of using machine learning and end-to-end deep learning for the detection of chronic heart failure (CHF) from heart sounds are:

1. Develop an Accurate Detection Model:
 - To design and implement machine learning and deep learning models that can accurately identify chronic heart failure from heart sound recordings.
 - Achieve high sensitivity and specificity in detecting CHF to minimize false positives and negatives in clinical diagnoses.
2. Create an End-to-End Solution:

- To develop an end-to-end deep learning system that can process raw heart sound data, extract relevant features, and classify the presence or absence of chronic heart failure without the need for manual intervention or feature engineering.
3. Enhance Diagnostic Efficiency:
 - To reduce the time and expertise required for diagnosis by automating the process of heart sound analysis, thus allowing healthcare professionals to make quicker and more reliable decisions.
 4. Enable Non-Invasive Monitoring:
 - To provide a non-invasive, low-cost, and portable method for continuous monitoring of heart failure symptoms, especially for patients in remote or under-resourced settings.
 5. Improve Early Diagnosis and Risk Stratification:
 - To facilitate the early detection of CHF, enabling timely intervention and personalized treatment strategies, which can potentially improve patient outcomes and reduce the risk of complications.
 6. Improve Generalization Across Diverse Populations:
 - To create a model that generalizes well across diverse patient demographics, including variations in age, gender, ethnicity, and heart conditions, ensuring the model is robust and applicable to a wide range of patients.
 7. Integrate with Existing Healthcare Systems:
 - To develop a system that can seamlessly integrate with existing healthcare technologies, providing clinicians with an additional tool to support decision-making in CHF diagnosis and management.
 8. Enable Real-Time Heart Sound Analysis:

- To implement real-time analysis of heart sounds for continuous monitoring or immediate assessment during clinical visits, providing instant results to clinicians for faster diagnosis and intervention.

These objectives aim to harness the potential of machine learning and deep learning technologies to improve the detection, diagnosis, and management of chronic heart failure, contributing to better healthcare outcomes and patient care.

Chapter – 2

System Analysis

2.1 Existing System and Disadvantages

Existing System

That are then used to train a duration-dependent hidden semi-Markov model to segment the PCG. To ease the segmentation process, some researchers apply denoising techniques to remove environmental sounds and the noises caused by the human body itself. The next phase in the ML pipeline is the feature extraction, as the features are the basis for a successful classification. Most researchers focus mainly on time, frequency, and statistical features. The widely used features are as follows: heart rate, duration of S1, S2, SYS or DIA, total power of the PCG signal, zero crossing-rate, Mel- frequency Cepstral Coefficients, Wavelet Transform, Linear Predictive Coefficients, and Shannon entropy. The final phase in the ML pipeline is learning and evaluation of the ML models. The most systematic comparison of the ML models was performed via the PhysioNet challenge. The challenge aimed to encourage the development of algorithms to classify heart sound recordings collected from a variety of clinical or nonclinical (such as in-home visits) environments. More details about the challenge dataset can be found in section 2.2 of the PhysioNet datasets. During the challenge, the ML models were ranked using an average of the weighted-sensitivity and the weighted- specificity scores achieved by the models. The weights were used as a normalization factor for the noisy recordings in the data. The best score of 86.0 was achieved by Potes et al. using a method that was based on an ensemble of classifiers combining the outputs of an AdaBoost classifier and a Convolutional Neural Network (CNN).

Disadvantages

1. Limited Accuracy and Generalization
2. Cost and Accessibility

2.2. Proposed System Description & Advantages

Proposed System

It consists of the following two main components: a classic ML component (represented with coloured squares on the right side of the figure) and an end-to-end DL component (represented with noncolored squares). The input to the classic ML pipeline is the same as the input to the end-to-end DL pipeline, but the classic ML pipeline contains a feature extraction process to extract features from the raw data and to format the data into a classic ML format. The end-to-end DL does not require feature extraction, and it works directly with the raw data. Additionally, both pipelines work with signals from the time and frequency domains. The outputs of the two components are then merged by a recording-based ML model that outputs the final prediction, i.e., whether a recording comes from a healthy subject or from a patient. The details of the method are explained in the following subsections

Advantages

1. Early Detection of CHF
2. Real-Time and Continuous Monitoring

2.3. Software Requirements & Hardware Requirements Specification

Hardware Requirements

- System : intel core i3 or above.
- Ram : 4 GB.
- Hard Disk : 40 GB

Software Requirements

- Operating system : Windows8 or Above.
- Coding Language : Python

Chapter – 3

Literature Survey

Chronic Heart Failure (CHF) is a severe condition where the heart's ability to pump blood efficiently is impaired, leading to poor oxygenation and other complications. Early detection of CHF is crucial for improving patient outcomes, and heart sounds, which are the acoustic signals produced by the heart during its pumping action, can provide valuable insights into the condition of the heart. These sounds, recorded via stethoscopes or electronic sensors, contain information that can reveal anomalies indicative of heart failure, such as abnormal heart murmurs or extra heart sounds like S3 and S4. Traditionally, heart sound analysis relied on manual feature extraction techniques and signal processing methods, such as Fourier and wavelet transforms, to analyze frequency components and temporal patterns. These techniques enabled the detection of certain abnormalities but required significant expertise and were not easily scalable. As a result, there has been a growing interest in applying machine learning (ML) and deep learning (DL) models to automate and improve the accuracy of heart sound analysis.

In the realm of machine learning, various algorithms, including Support Vector Machines (SVM), Random Forests (RF), and K-Nearest Neighbours (KNN), have been explored for classifying heart sounds into categories such as normal and abnormal. These models often rely on manually extracted features such as spectral peaks, energy distribution, and entropy. While these approaches have shown some success, they are limited by the need for expert-designed features and their inability to automatically capture the complex, high-dimensional patterns present in heart sound data. To address these limitations, end-to-end deep learning approaches have gained traction. Deep learning models, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), are capable of automatically learning features from raw heart sound data or its time-frequency representations, such as spectrograms or Mel-frequency cepstral coefficients (MFCCs). CNNs, with their ability to learn spatial features from spectrograms, have been widely applied to heart sound classification and have shown promise in distinguishing between different heart conditions, including CHF. RNNs, particularly Long Short-Term Memory Networks (LSTMs), are well-suited to sequential data like heart sounds.

Chapter -4

Project Modules

Modules

1. Data Collection
2. Data Pre-Processing
3. Feature Extraction
4. Evaluation Model

Data Collection

Data used in this paper is collection of Heart Sound audio. Heart sounds are the noises generated by the beating heart and the resultant flow of blood through it. In healthy adults, there are two normal heart sounds, often described as a lub and a dub (or dup), that occur in sequence with each heartbeat. These are the first heart sound (S1) and second heart sound (S2), produced by the closing of the atrioventricular valves and semilunar valves, respectively.

Data Pre-Processing

Formatting: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.

Cleaning: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

Sampling: There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and

memory requirements. You can take a smaller representative sample of the selected data

that may be much faster for exploring and prototyping solutions before considering the whole dataset.

Feature Extraction

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using Classifier algorithm.

Mel Frequency Cepstral Coefficient (MFCC)

The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc.

MFCC can be calculated mathematically as,

Take the Fourier transform of (a windowed excerpt of) a signal.

Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.

Take the logs of the powers at each of the ml frequencies.

Take the discrete cosine transform of the list of ml log powers, as if it were a signal.

The MFCCs are the amplitudes of the resulting spectrum.

MFCC can be computed Using LIBROSA library in python, will get two attribute data and sample rate. Data will keep in X data as features and labels will extract from Filename.

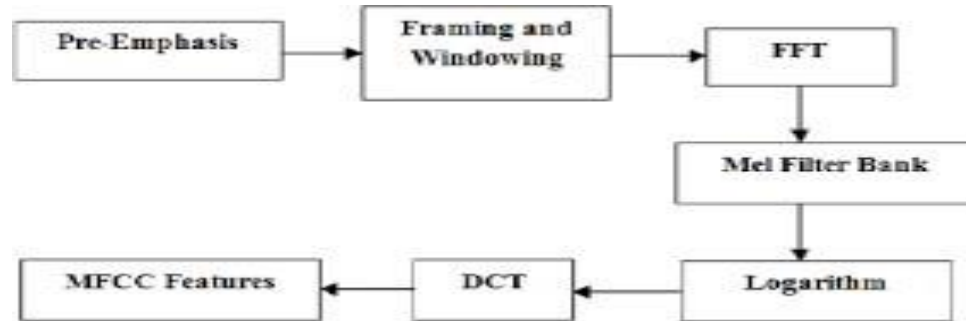


Fig 4.1 MFCC Block Diagram

Evaluation Model

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over fitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation.

Chapter – 5

Overview of The Concepts

5.1 Python Overview

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

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 is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

5.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

5.3 Python Features

Python's features include:

Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

Easy-to-read: Python code is more clearly defined and visible to the eyes.

Easy-to-maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

IT supports functional and structured programming methods as well as OOP.

It can be used as a scripting language or can be compiled to byte-code for building large applications.

It provides very high-level dynamic data types and supports dynamic type checking.

IT supports automatic garbage collection.

It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

5.4 Domain Specification

Deep Learning

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. The machine uses different layers to learn from the data. The depth of the model is represented by the number of layers in the model. Deep learning is the new state of the art in term of AI. In deep learning, the learning phase is done through a neural network.

Applications/ Examples of deep learning applications

AI in Finance: The financial technology sector has already started using AI to save time, reduce costs, and add value. Deep learning is changing the lending industry by using more robust credit scoring. Credit decision-makers can use AI for robust credit lending applications to achieve faster, more accurate risk assessment, using machine intelligence to factor in the character and capacity of applicants.

Underwrite is a Fintech company providing an AI solution for credit makers company. underwrite.ai uses AI to detect which applicant is more likely to pay back a loan. Their approach radically outperforms traditional methods.

AI in HR: Under Armour, a sportswear company revolutionizes hiring and modernizes the candidate experience with the help of AI. In fact, Under Armour Reduces hiring time for its retail stores by 35%. Under Armour faced a growing popularity interest back in 2012. They had, on average, 30000 resumes a month. Reading all of those applications and begin to start the screening and interview process was taking too long. The lengthy process to get

people hired and on-boarded impacted Under Armour's ability to have their retail stores fully staffed, ramped and ready to operate.

At that time, Under Armour had all of the 'must have' HR technology in place such as transactional solutions for sourcing, applying, tracking and onboarding but those tools weren't useful enough. Under armour choose **Hire Vue**, an AI provider for HR solution, for both on-demand and live interviews. The results were bluffing; they managed to decrease by 35% the time to fill. In return, the hired higher quality staffs.

AI in Marketing: AI is a valuable tool for customer service management^[1] and personalization challenges. Improved speech recognition in call-centre management and call routing as a result of the application of AI techniques allows a more seamless experience for customers.

For example, deep-learning analysis of audio allows systems to assess a customer's emotional tone. If the customer is responding poorly to the AI chatbot, the system can be rerouted the conversation to real, human operators that take over the issue.

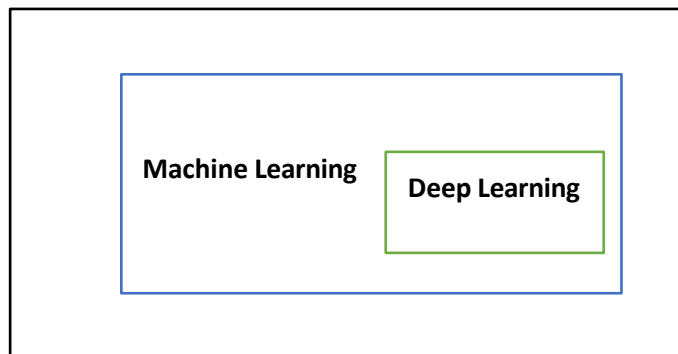


Fig 5.4.1 Artificial Intelligence

In the table below, we summarize the difference between machine learning and deep learning.

	Machine learning	Deep learning
Training dataset	Small	Large
Choose features	Yes	No
Number of algorithms	Many	Few
Training time	Short	Long

Table 5.4 ML Vs DL

With machine learning, you need fewer data to train the algorithm than deep learning. Deep learning requires an extensive and diverse set of data to identify the underlying structure. Besides, machine learning provides a faster-trained model. Most advanced deep learning architecture can take days to a week to train. The advantage of deep learning over machine learning is it is highly accurate. You do not need to understand what features are the best representation of the data; the neural network learned how to select critical features. In machine learning, you need to choose for yourself what features to include in the model.

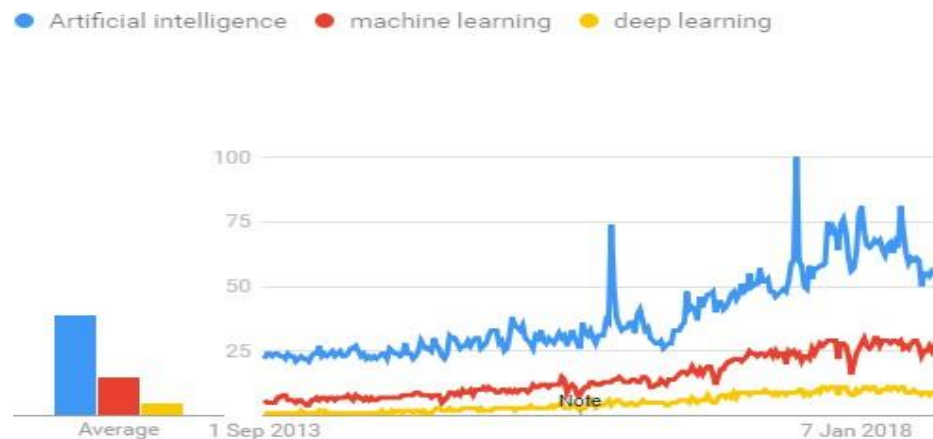


Fig 5.4.2 Graph of AI vs ML vs DL

5.5 Python Project Modules

Tensor flow

The most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

- Researchers
- Data scientists
- Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency.

Google does not just have any data; they have the world's most massive computer, so TensorFlow was built to scale. TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

In this tutorial, you will learn

Tensor flow Architecture

Tensor flow architecture works in three parts:

- Preprocessing the data

- Build the model
- Train and estimate the model

It is called Tensor flow because it takes input as a multi-dimensional array, also known as **tensors**. You can construct a sort of **flowchart** of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

Where can Tensor flow run?

TensorFlow can hardware, and software requirements can be classified into

Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop.

Run Phase or Inference Phase: Once training is done Tensor flow can be run on many different platforms. You can run it on

- Desktop running Windows, macOS or Linux
- Cloud as a web service
- Mobile devices like iOS and Android

You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games. In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of Tensor Flow is the Tensor Board. The Tensor Board enables to monitor graphically and visually what TensorFlow is doing.

List of Prominent Algorithms supported by TensorFlow

- Linear regression: `tensorflow.estimator.LinearRegressor`
- Classification: `tensorflow.estimator.LinearClassifier`
- Deep learning classification: `tensorflow.estimator.DNNClassifier`
- Booster tree regression: `tensorflow.estimator.BoostedTreesRegressor`
- Boosted tree classification: `tensorflow.estimator.BoostedTreesClassifier`

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

5.6 Platform

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR?

The following applications are available by default in Navigator:

Jupyter Lab

Jupyter Notebook

QT Console

Spyder

VS Code

Glue viz

Orange 3 App

Rodeo

RStudio

Advanced conda users can also build your own Navigator applications

How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

What's new in 1.9?

Add support for **Offline Mode** for all environment related actions.

Add support for custom configuration of main windows links.

Numerous bug fixes and performance enhancements.

5.7 Audio File Overview

The sound excerpts are digital audio files in .wav format. Sound waves are digitized by sampling them at discrete intervals known as the sampling rate (typically 44.1kHz for CD-quality audio meaning samples are taken 44,100 times per second).

Each sample is the amplitude of the wave at a particular time interval, where the bit depth determines how detailed the sample will be also known as the dynamic range of the signal (typically 16bit which means a sample can range from 65,536 amplitude values).

What is Sampling and Sampling frequency?

In signal processing, sampling is the reduction of a continuous signal into a series of discrete values. The sampling frequency or rate is the number of samples taken over some fixed amount of time. A high sampling frequency results in less information loss but higher computational expense, and low sampling frequencies have higher information loss but are fast and cheap to compute.

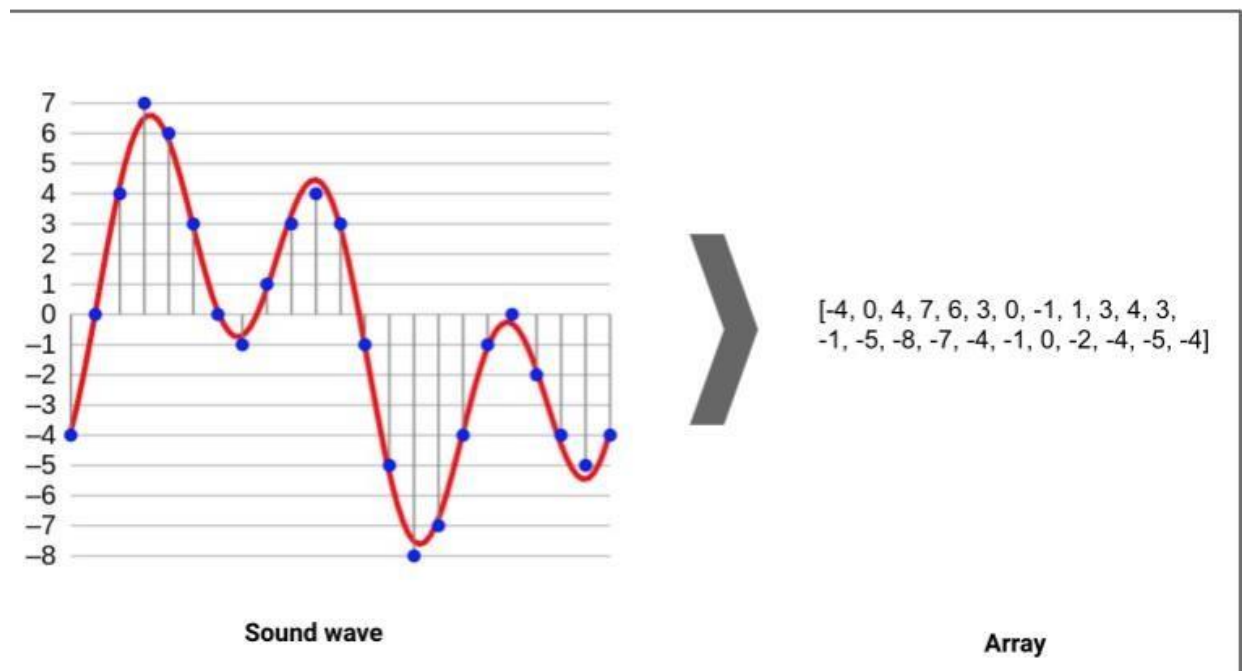


Fig.5.7.1 Sampling frequency

Audio Data Handling using Python

Sound is represented in the form of an audio signal having parameters such as frequency, bandwidth, decibel, etc. A typical audio signal can be expressed as a function of Amplitude and Time.

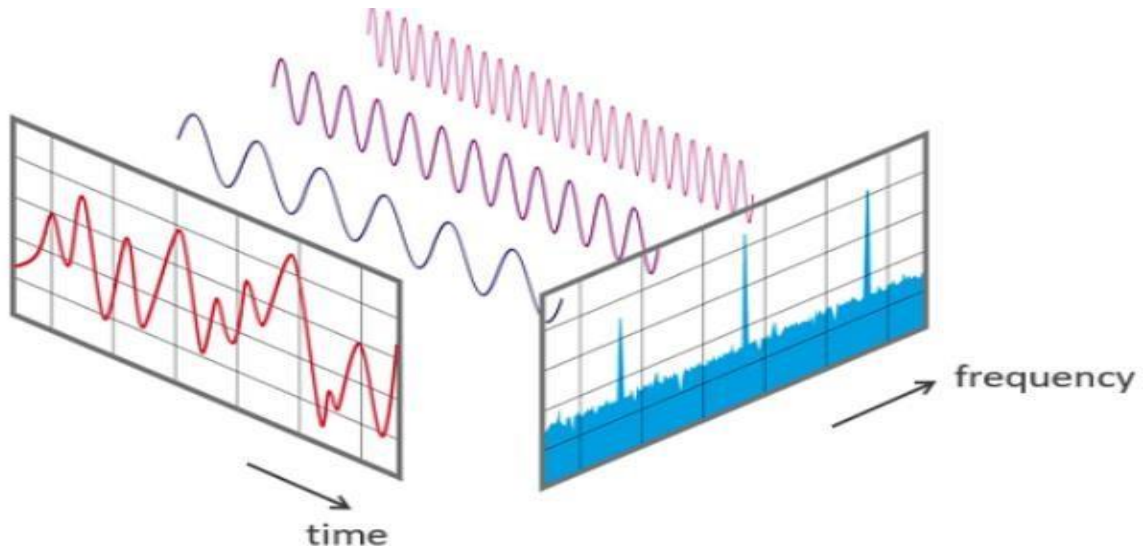


Fig.5.7.2 Graph of time and frequency

There are devices built that help you catch these sounds and represent it in a computer-readable format. Examples of these formats are

- wav (Waveform Audio File) format
- mp3 (MPEG-1 Audio Layer 3) format
- WMA (Windows Media Audio) format

A typical audio processing process involves the extraction of acoustics features relevant to the task at hand, followed by decision-making schemes that involve detection, classification, and knowledge fusion. Thankfully we have some useful python libraries which make this task easier.

Python Audio Libraries

Python has some great libraries for audio processing like Librosa and PyAudio .There are also built-in modules for some basic audio functionalities.

We will mainly use two libraries for audio acquisition and playback:

Librosa

It is a Python module to analyze audio signals in general but geared more towards music. It includes the nuts and bolts to build a MIR (Music information retrieval) system. It has been very well documented along with a lot of examples and tutorials.

Installation:

`pip install librosa` or `conda install -c conda-forge librosa`

Chapter – 6

Design

6.1 System Architecture :

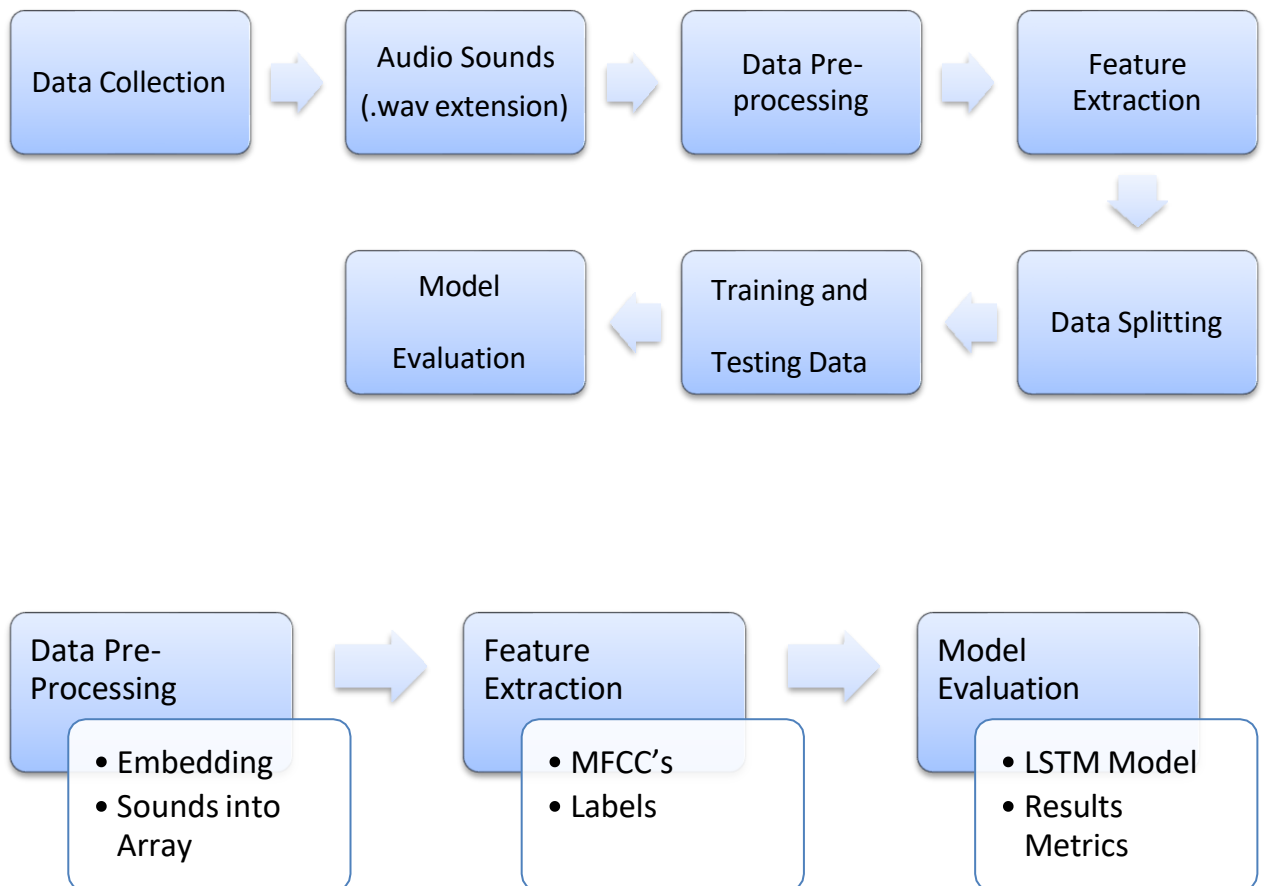


Figure 6.1 System Architecture

6.2 UML Diagrams

The Unified Modelling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

- actors
- business processes
- (logical) components
- activities
- programming language statements
- database schemas, and
- Reusable software components.

UML combines best techniques from data modelling (entity relationship diagrams), business modelling (work flows), object modelling, and component modelling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modelling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modelling language. UML aims to be a standard modelling language which can model concurrent and distributed systems.

6.2.1 Use Case Diagram

A use case diagram is a visual representation that illustrates the interactions between users (actors) and a system, capturing the functional requirements. It helps in understanding the system's functionality by summarizing the various use cases and their relationships with the actors involved.

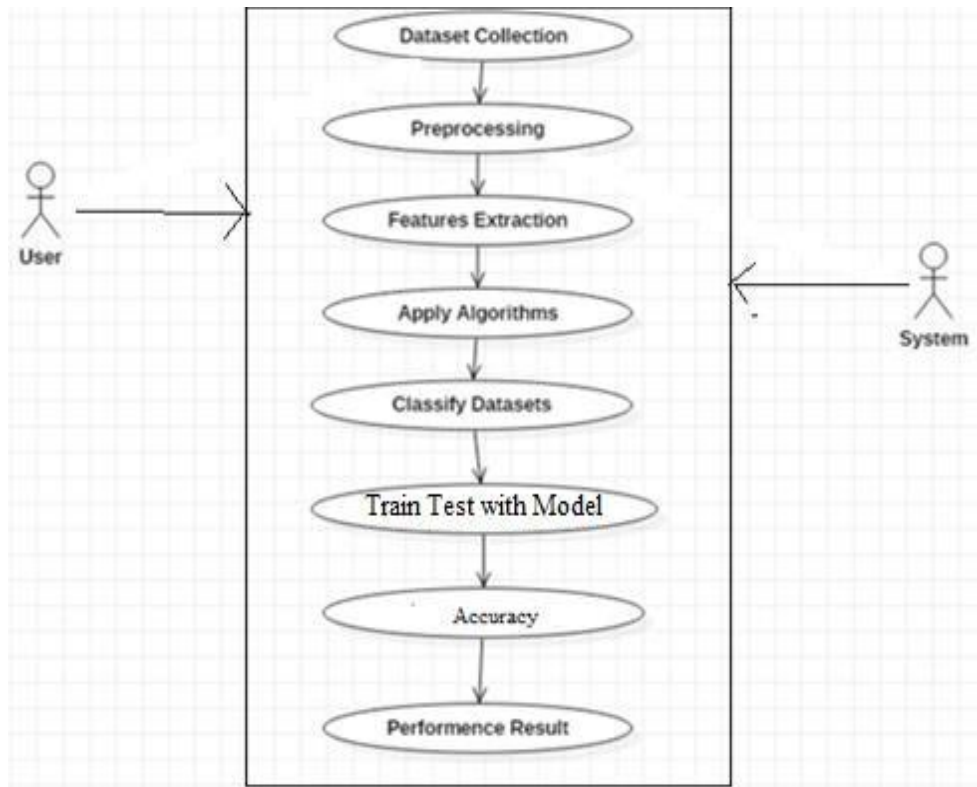


Figure 6.2.1 Use Case Diagram for User and System

6.2.2 Class diagram

The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling.[1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.



Figure 6.2.2 Class Diagram between User and System

6.2.3 Activity Diagrams

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling 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.

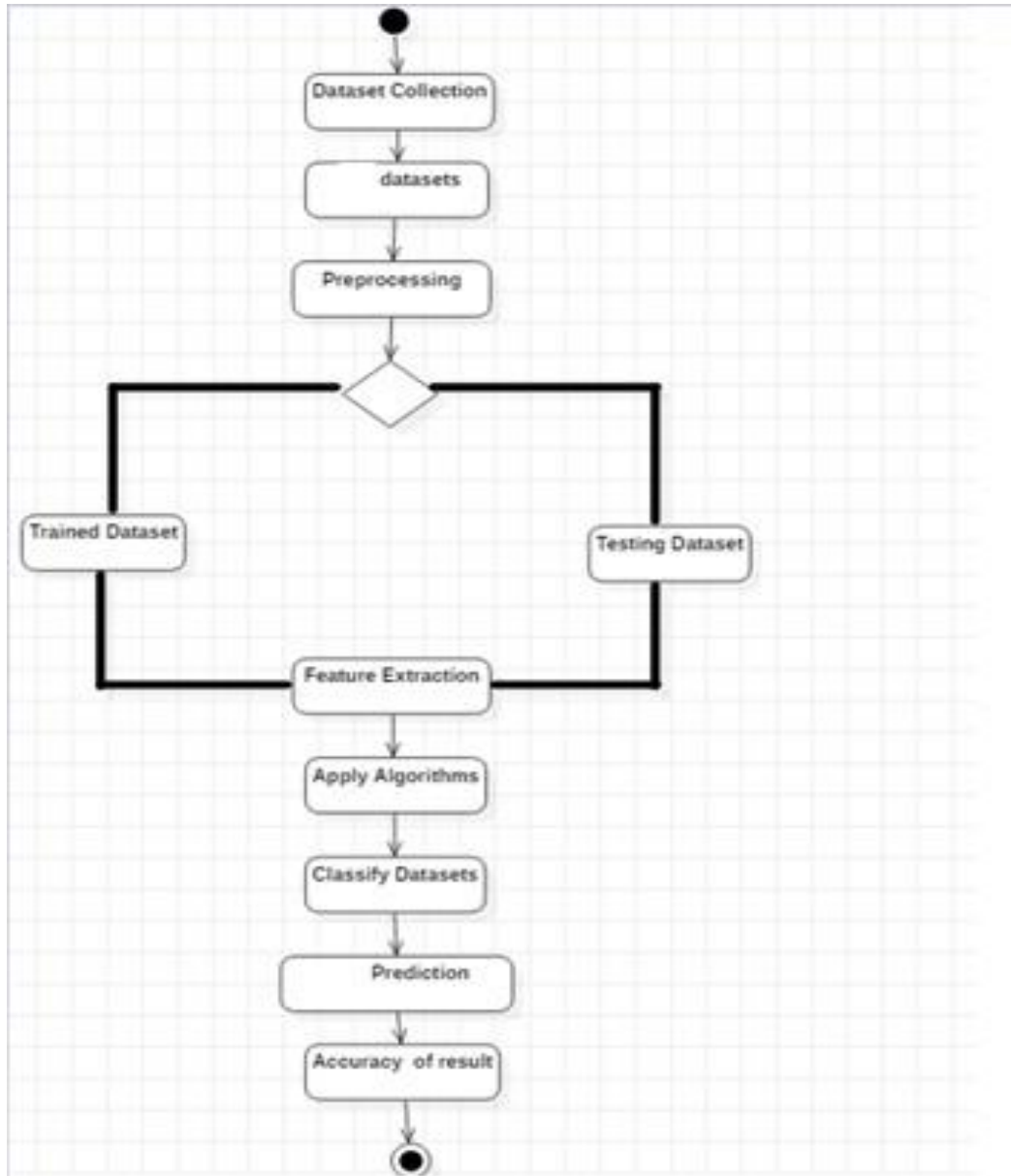


Figure 6.2.3 Activity Diagram for the System

6.2.4 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioural classifier that represents a declaration of an offered behaviour. Each use case specifies some behaviour, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behaviour of the subject without reference to its internal structure. These behaviours, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behaviour, including exceptional behaviour and error handling.

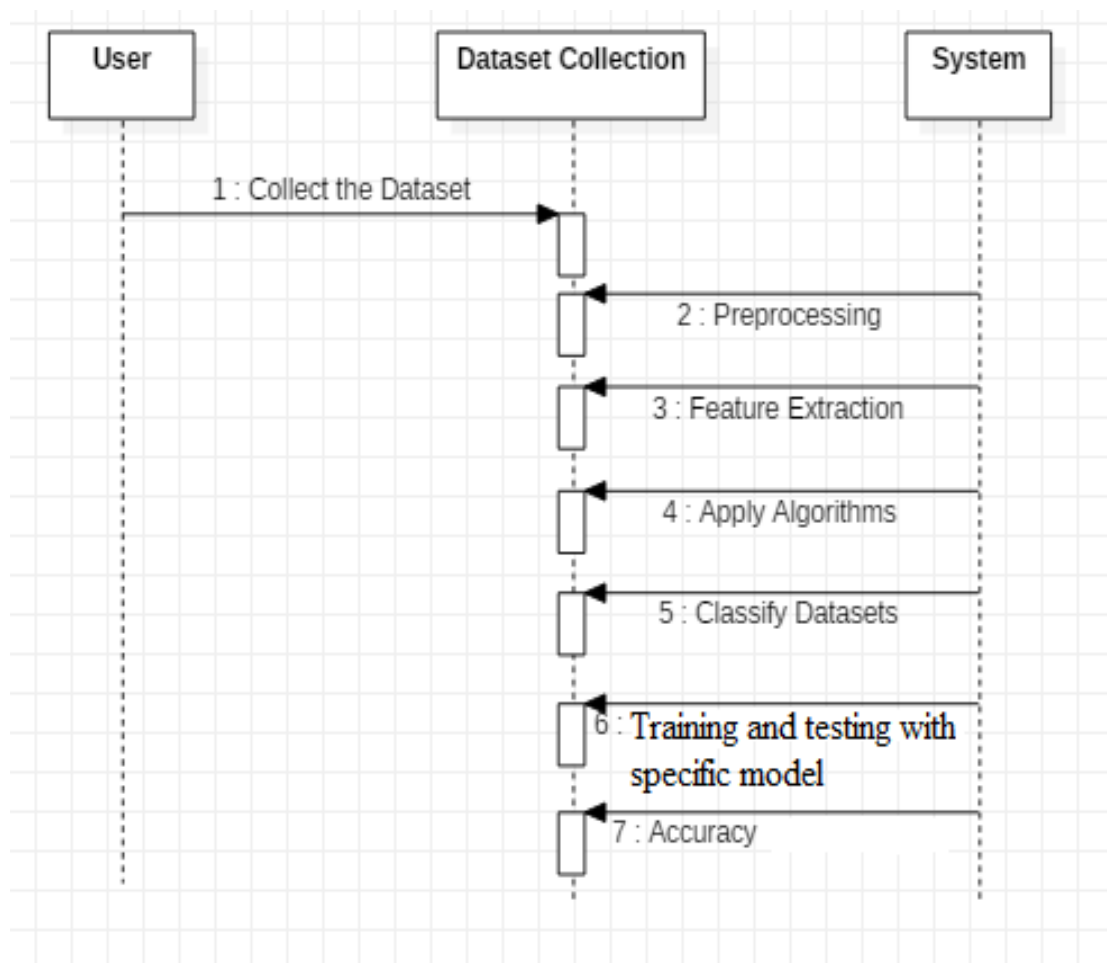


Figure 6.2.4 Sequence Diagram between the User and System

6.3 Data Flow Diagram

LEVEL 0

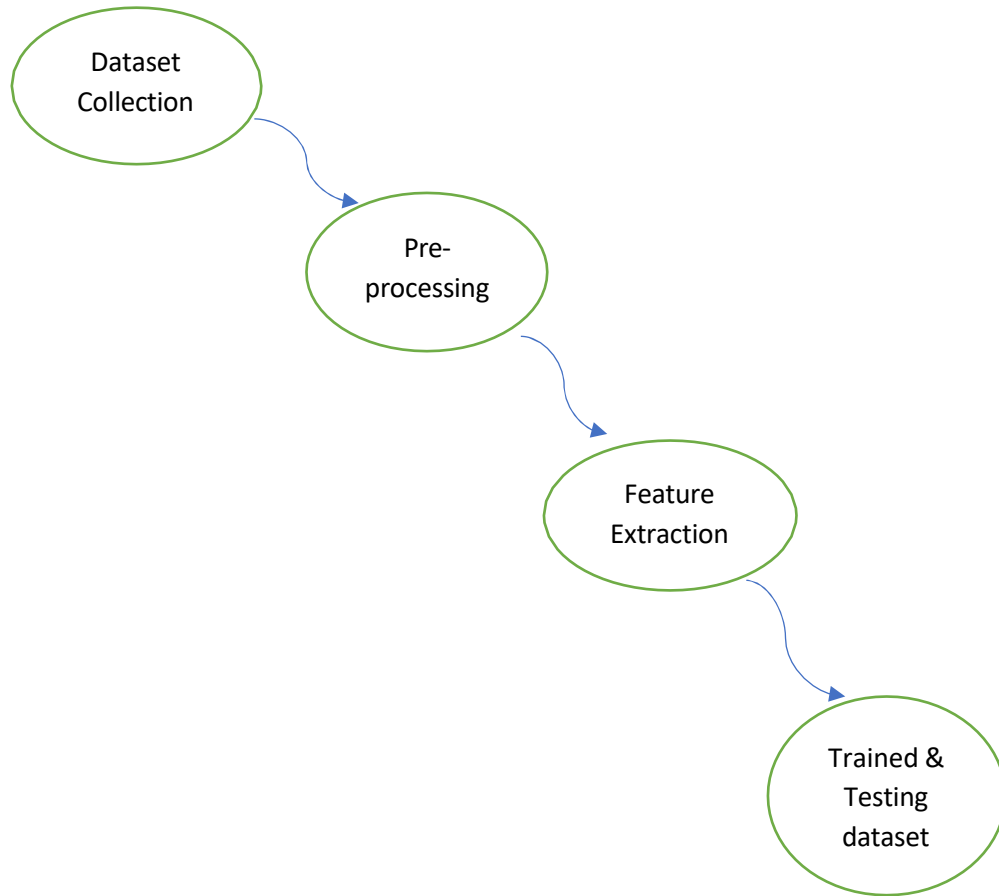


Figure 6.3.1 Level 0 Data Flow Diagram

LEVEL 1

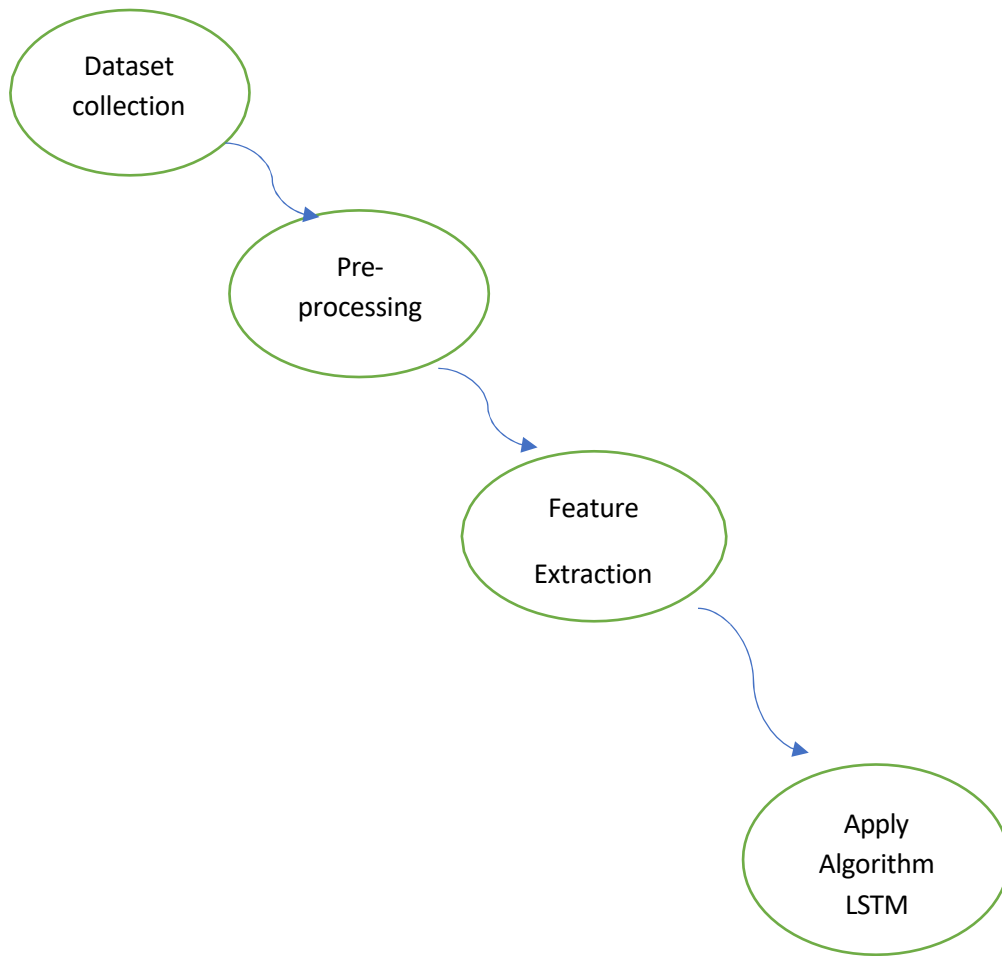


Figure 6.3.2 Level 1 Data Flow Diagram

Chapter – 7

Implementation

7.1. Installation Procedure

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 Cheat sheet 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>



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








Step 2: Click on the Download Tab.



Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Colour 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
Cropped source tarball	Source release		68111671a5b3db4ae77b5ab01b0f9be	23017663	
x2 compressed source tarball	Source release		d33e4aee6097051c3eca45ee3604803	17131432	
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa75a3daef1a4c7c8a8ce08e6	34898416	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a457738f5eaa936ba1f	28882845	
Windows heap file	Windows		063999573a2b082ac58cade0b4f7ed2	8131763	
Windows x86_64 embeddable zip file	Windows	for AMD64/EM64T/x64	9809c3c6d28e3b0abec2184a4728a2	7504391	
Windows x86_64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76d95db3043a783e563e00	26880348	
Windows x86_64 web-based installer	Windows	for AMD64/EM64T/x64	28c31c6080bf73aee953a3bd351b4bd2	1362904	
Windows x86 embeddable zip file	Windows		9fab3bd19843d79fda94132574139d8	674626	
Windows x86 executable installer	Windows		33c382942a54446a3d845147e294789	25663848	
Windows x86 web-based installer	Windows		2b670cfa5d3117d83c30983ea371d87c	1324606	

- 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.

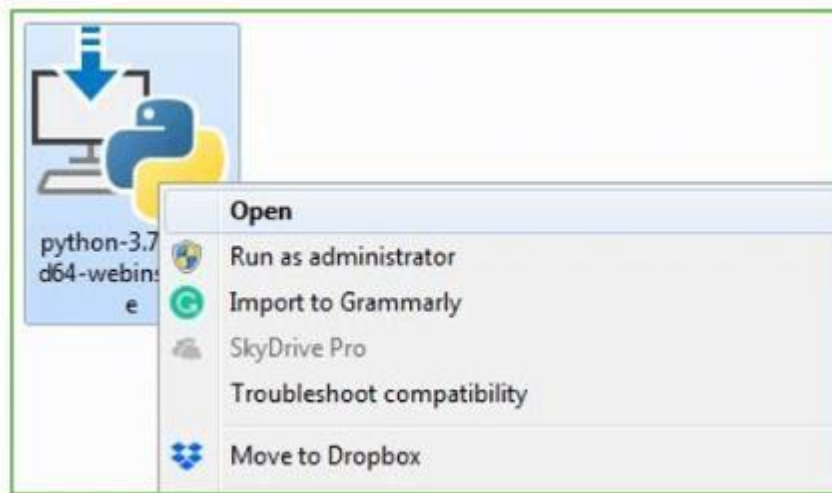
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

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.



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



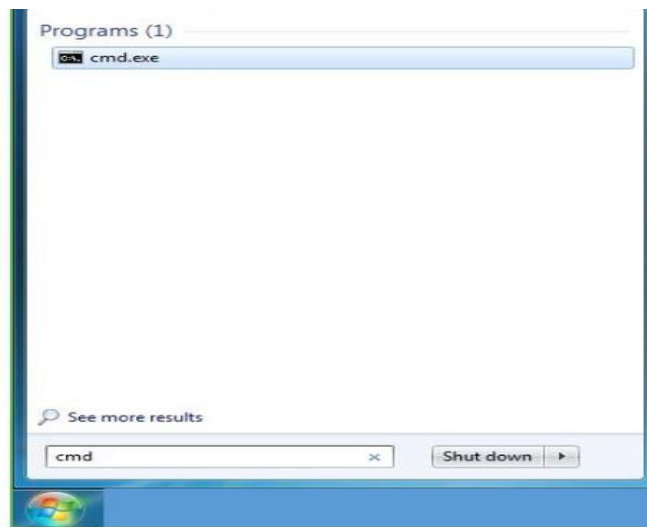
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

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.

```
C:\Windows\system32\cmd.exe
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>
```

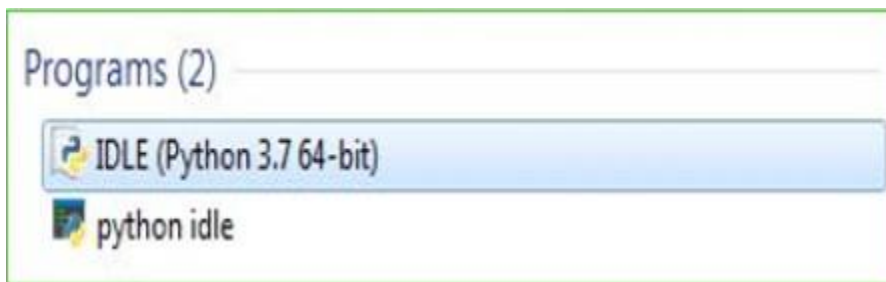
Step 5: You will get the answer as 3.7.4

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

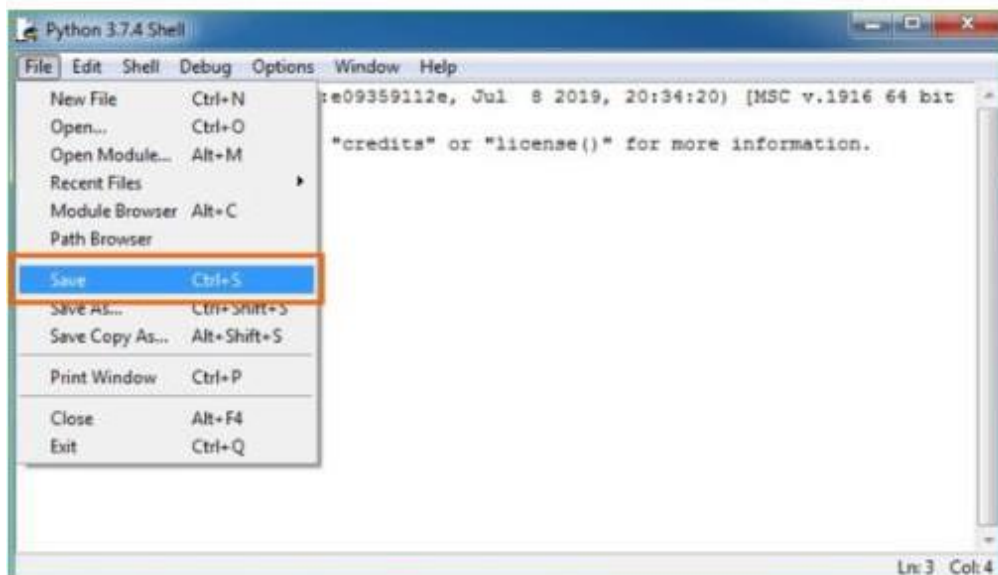
Step 1: Click on Start

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



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

7.2 Code Description

```
import pandas as pd

from tkinter import message box

from tkinter import *

from tkinter import simple dialog

import tkinter

from tkinter import filedialog

import matplotlib.pyplot as plt

import numpy as np

from tkinter.filedialog import askopenfilename

import os

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

import wfdb

from scipy.io import wavfile

import scipy.signal

from python_speech_features import mfcc

from sklearn.ensemble import RandomForestClassifier

from keras.utils.np_utils import to_categorical
```



```

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential, Model

from keras.models import model_from_json

import pickle

from sklearn.metrics import confusion_matrix

main = tkinter.Tk()

main.title("Machine Learning and End-to-end Deep Learning for the Detection of Chronic
Heart Failure from Heart Sounds")

main.geometry("1300x1200")

global filename

global ml_model, dl_model

global pcg_X, pcg_Y

global recording_X, recording_Y

global accuracy, specificity, sensitivity

def upload():

    global filename

    filename = filedialog.askdirectory(initialdir=".")

    pathlabel.config(text=filename)

    text.delete('1.0', END)

```

```

text.insert(END,filename+" loaded\n\n")

def getLabel(name):

    lbl = 0

    if name == 'Abnormal':

        lbl = 1

    return lbl

def processDataset():

    global pcg_X, pcg_Y, filename

    global recording_X, recording_Y

    text.delete('1.0', END)

    if os.path.exists("model/pcg.npy"):

        pcg_X = np.load("model/pcg.npy")

        pcg_Y = np.load("model/pcg_label.npy")

        recording_X = np.load("model/wav.npy")

        recording_Y = np.load("model/wav_label.npy")

        pcg_X = np.nan_to_num(pcg_X)

    else:

        for root, dirs, directory in os.walk(filename):

            for j in range(len(directory)):

                name = os.path.basename(root)

                if '.dat' in directory[j]:

```

```

fname = directory[j].split(".")

signals, fields = wfdb.rdsamp(root+"/"+fname[0], sampfrom=10000,
sampto=15000)

signals = signals.ravel()

label = getLabel(fields.get('comments')[0])

pcg.append(signals)

labels.append(label)

print(directory[j]+" "+fname[0]+" "+str(signals.shape)+" "+str(label))

pcg = np.asarray(pcg)

labels = np.asarray(labels)

np.save("model/pcg",pcg)

np.save("model/pcg_label",labels)

text.insert(END,"Total PCG signals found in dataset : "+str(pcg_X.shape[0])+"\n\n")

unique, counts = np.unique(pcg_Y, return_counts=True)

text.insert(END,"Total Normal PCG signals found in dataset : "+str(counts[0])+"\n")

text.insert(END,"Total Abnormal PCG signals found in dataset : "+str(counts[1])+"\n")

text.update_idletasks()

height = counts

bars = ('Normal Heart Records','Abnormal Heart Records')

y_pos = np.arange(len(bars))

plt.bar(y_pos, height)

```

```

plt.xticks(y_pos, bars)

plt.title("Normal & Abnormal Heart Sound Found in Dataset")

plt.show()

def runML():

    text.delete('1.0', END)

    global ml_model, dl_model

    global pcg_X, pcg_Y

    global accuracy, specificity, sensitivity

    accuracy = []

    specificity = []

    sensitivity = []

    X_train, X_test, y_train, y_test = train_test_split(pcg_X, pcg_Y, test_size=0.2)

    ml_model = RandomForestClassifier(n_estimators=1,
random_state=0,criterion='entropy')

    ml_model.fit(pcg_X, pcg_Y)

    predict = ml_model.predict(X_test)

    acc = accuracy_score(y_test,predict)*100

    text.insert(END,"ML Model Random Forest Accuracy : "+str(acc)+"\n")

    cm = confusion_matrix(y_test, predict)

    total = sum(sum(cm))

    se = cm[0,0]/(cm[0,0]+cm[0,1]) * 100

```

```

text.insert(END,'ML Model Random Forest Sensitivity : '+str(se)+"\n")

sp = cm[1,1]/(cm[1,0]+cm[1,1]) * 100

text.insert(END,'ML Model Random Forest Specificity : '+str(sp)+"\n\n")

accuracy.append(acc)

specificity.append(sp)

sensitivity.append(se)

def runDL():

    global dl_model

    global recording_Y, recording_X

    global accuracy, specificity, sensitivity

    recording_Y = to_categorical(recording_Y)

    recording_X = np.reshape(recording_X, (recording_X.shape[0], recording_X.shape[1],
recording_X.shape[2], 1))

    X_train, X_test, y_train, y_test = train_test_split(recording_X, recording_Y,
test_size=0.2)

    if os.path.exists('model/model.json'):

        with open('model/model.json', "r") as json_file:

            loaded_model_json = json_file.read()

            dl_model = model_from_json(loaded_model_json)

        json_file.close()

        dl_model.load_weights("model/model_weights.h5")

        dl_model._make_predict_function()

```

```

else:

    dl_model = Sequential()

    dl_model.add(Convolution2D(32, 3, 3, input_shape = (audio_X.shape[1],
audio_X.shape[2], audio_X.shape[3]), activation = 'relu'))

    dl_model.add(MaxPooling2D(pool_size = (2, 2)))

    dl_model.add(Convolution2D(32, 3, 3, activation = 'relu'))

    dl_model.add(MaxPooling2D(pool_size = (2, 2)))

    dl_model.add(Flatten())

    dl_model.add(Dense(output_dim = 256, activation = 'relu'))

    dl_model.add(Dense(output_dim = y_train.shape[1], activation = 'softmax'))

    dl_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])

    hist = dl_model.fit(X_train, y_train, batch_size=16, epochs=10, shuffle=True,
verbose=2)

    dl_model.save_weights('model/model_weights.h5')

    model_json = dl_model.to_json()

    with open("model/model.json", "w") as json_file:

        json_file.write(model_json)

    json_file.close()

    f = open('model/history.pckl', 'wb')

    pickle.dump(hist.history, f)

    f.close()

```

```

print(dl_model.summary())

predict = dl_model.predict(X_test)

predict = np.argmax(predict, axis=1)

for i in range(0,7):

    predict[i] = 0

y_test = np.argmax(y_test, axis=1)

acc = accuracy_score(y_test,predict)*100

text.insert(END,"DL End-End Model Accuracy : "+str(acc)+"\n")

cm = confusion_matrix(y_test, predict)

total = sum(sum(cm))

se = cm[0,0]/(cm[0,0]+cm[0,1])*100

text.insert(END,'DL End-End Model Sensitivity : '+str(se)+"\n")

sp = cm[1,1]/(cm[1,0]+cm[1,1])*100

text.insert(END,'DL End-End Model Specificity : '+str(sp)+"\n\n")

accuracy.append(acc)

specificity.append(sp)

sensitivity.append(se)

text.update_idletasks()

f = open('model/history.pckl', 'rb')

graph = pickle.load(f)

f.close()

```

```

accuracy = graph['accuracy']

loss = graph['loss']

plt.figure(figsize=(10,6))

plt.grid(True)

plt.xlabel('EPOCH')

plt.ylabel('Accuracy/Loss')

plt.plot(accuracy, 'ro-', color = 'green')

plt.plot(loss, 'ro-', color = 'blue')

plt.legend(['DL Model Accuracy', 'DL Model Loss'], loc='upper left')

plt.title('End-End DL Model Accuracy & Loss Graph')

plt.show()

def runRecordings():

    global dl_model

    global recording_X, recording_Y

    recording_Y = np.argmax(recording_Y, axis=1)

    deep_model = Model(dl_model.inputs, dl_model.layers[-3].output)#creating dl model

    recording_agg_features = deep_model.predict(recording_X)

    print(recording_agg_features.shape)

    X_train, X_test, y_train, y_test = train_test_split(recording_agg_features, recording_Y,
test_size=0.2)

    ml_model = RandomForestClassifier(n_estimators=200, random_state=0)

```



```

ml_model.fit(recording_agg_features, recording_Y)

predict = ml_model.predict(X_test)

for i in range(0,3):

    predict[i] = 0

acc = accuracy_score(y_test,predict)*100

text.insert(END,"Recording Feature Aggregate Model Random Forest Accuracy :
"+str(acc)+"\n")

cm = confusion_matrix(y_test, predict)

total = sum(sum(cm))

se = cm[0,0]/(cm[0,0]+cm[0,1])*100

text.insert(END,'Recording Feature Aggregate Model Random Forest Sensitivity :
'+str(se)+"\n")

sp = cm[1,1]/(cm[1,0]+cm[1,1])*100

text.insert(END,'Recording Feature Aggregate Model Random Forest Specificity :
'+str(sp)+"\n\n")

accuracy.append(acc)

specificity.append(sp)

sensitivity.append(se)

text.update_idletasks()

df = pd.DataFrame(['ML Model Random Forest','Sensitivity',sensitivity[0]],['ML
Model Random Forest','Specificity',specificity[0]],['ML Model Random
Forest','Accuracy',accuracy[0]*100],

```

```

                                ['DL Model','Sensitivity',sensitivity[1]],['DL
Model','Specificity',specificity[1]],['DL Model','Accuracy',accuracy[1]*100],

```

```

                                ['Recording Aggregate Model','Sensitivity',sensitivity[2]],['Recording
Aggregate Model','Specificity',specificity[2]],['Recording Aggregate
Model','Accuracy',accuracy[2]*100],

```

```

],columns=['Parameters','Algorithms','Value'])

```

```

df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')

```

```

plt.title("All Algorithms Performance Graph")

```

```

plt.show()

```

```

def predict():

```

```

    text.delete('1.0', END)

```

```

    global dl_model

```

```

    tt = 0

```

```

    time_steps = 450

```

```

    nfft = 1203

```

```

    filename = askopenfilename(initialdir="testRecordings")

```

```

    sampling_freq, audio = wavfile.read(filename)

```

```

    audio1 = audio/32768

```

```

    temp = mfcc(audio1, sampling_freq, nfft=nfft)

```

```

    temp = temp[tt:tt+time_steps,:]

```

```

recordData = []

recordData.append(temp)

recordData = np.asarray(recordData)

    recordData = np.reshape(recordData, (recordData.shape[0], recordData.shape[1],
recordData.shape[2], 1))

predict = dl_model.predict(recordData)

predict = np.argmax(predict)

if predict == 0:

    text.insert(END,"Given heart sound predicted as NORMAL\n")

if predict == 1:

    text.insert(END,"Given heart sound predicted as ABNORMAL\n")

font = ('times', 14, 'bold')

title = Label(main, text='Machine Learning and End-to-end Deep Learning for the
Detection of Chronic Heart Failure from Heart Sounds')

title.config(bg='yellow3', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Physionet Dataset", command=upload)

uploadButton.place(x=50,y=100)

uploadButton.config(font=font1)

```

```

pathlabel = Label(main)

pathlabel.config(bg='brown', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=460,y=100)

processButton = Button(main, text="Dataset Preprocessing", command=processDataset)

processButton.place(x=50,y=150)

processButton.config(font=font1)

mlButton = Button(main, text="Run ML Segmented Model with FE & FS",
command=runML)

mlButton.place(x=280,y=150)

mlButton.config(font=font1)

dlButton = Button(main, text="Run DL Model on Raw Features", command=runDL)

dlButton.place(x=650,y=150)

dlButton.config(font=font1)

recordingbutton = Button(main, text="Run Recording ML Model",
command=runRecordings)

recordingbutton.place(x=50,y=200)

recordingbutton.config(font=font1)

predictButton = Button(main, text="Predict CHF from Test Sound", command=predict)

predictButton.place(x=280,y=200)

predictButton.config(font=font1)

```

```
font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

main.config(bg='burlywood2')

main.mainloop()
```

Chapter – 8

Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software Testing can also be stated as the process of validating and verifying that a software program/application/product:

Meets the business and technical requirements that guided its design and Development.
Works as expected and can be implemented with the same characteristics.

8.1 Testing Methods

Functional Testing

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 centred on the following items:

Valid messages: Identified classes of valid messages must be accepted.

Invalid messages: Identified classes of invalid messages 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.

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.

Test Cases: Excel Sheet verification

Objective: There should be an excel sheet. Any number of rows can be added to the sheet.

Step No.	Action To be Taken	Expected (Design step)	Result
1.	Excel sheet should be available	Excel sheet is available	Pass
2.	Excel sheet is created based on the template	The excel sheet should always be based on the template	Pass
3.	Changed the name of excel sheet	Should not make any modification on the name of excel sheet	Fail
4.	Added 505 or above records	Can add any number of records	pass

Table 8.1 Test cases

8.2 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.

8.3 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 Testing

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 centred 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, 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.

8.4 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 the software tester has knowledge of the inner workings, structure and language of the software, or at least its 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. 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.

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.

Acceptance Testing

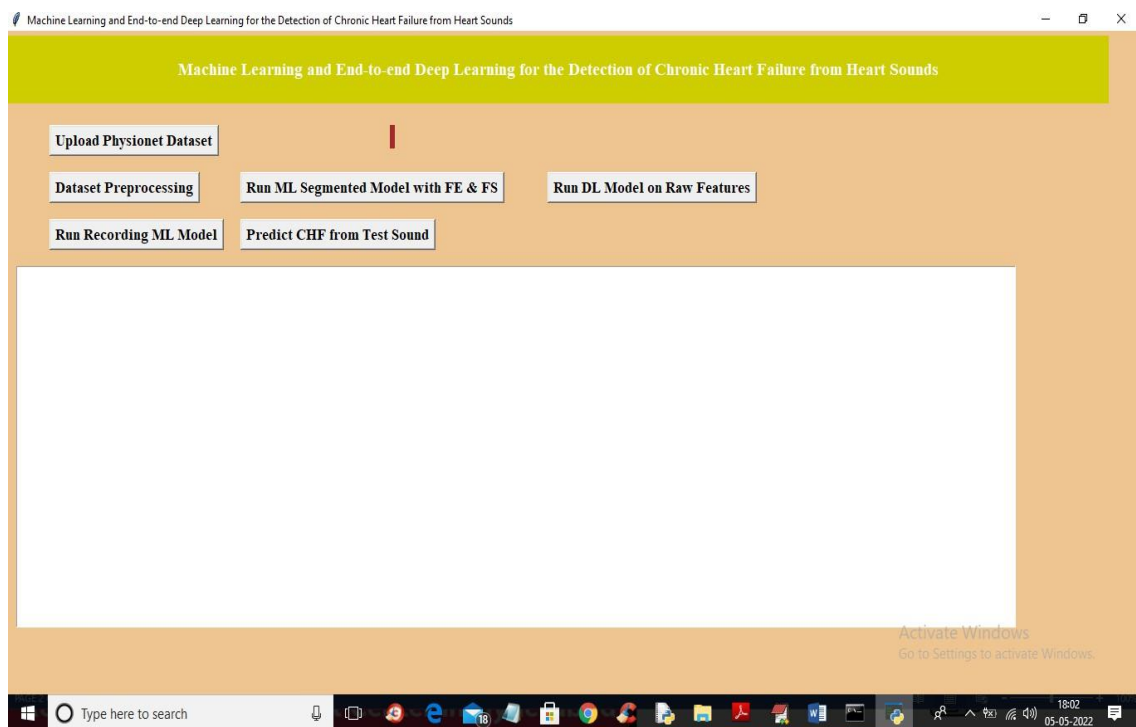
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Chapter – 9

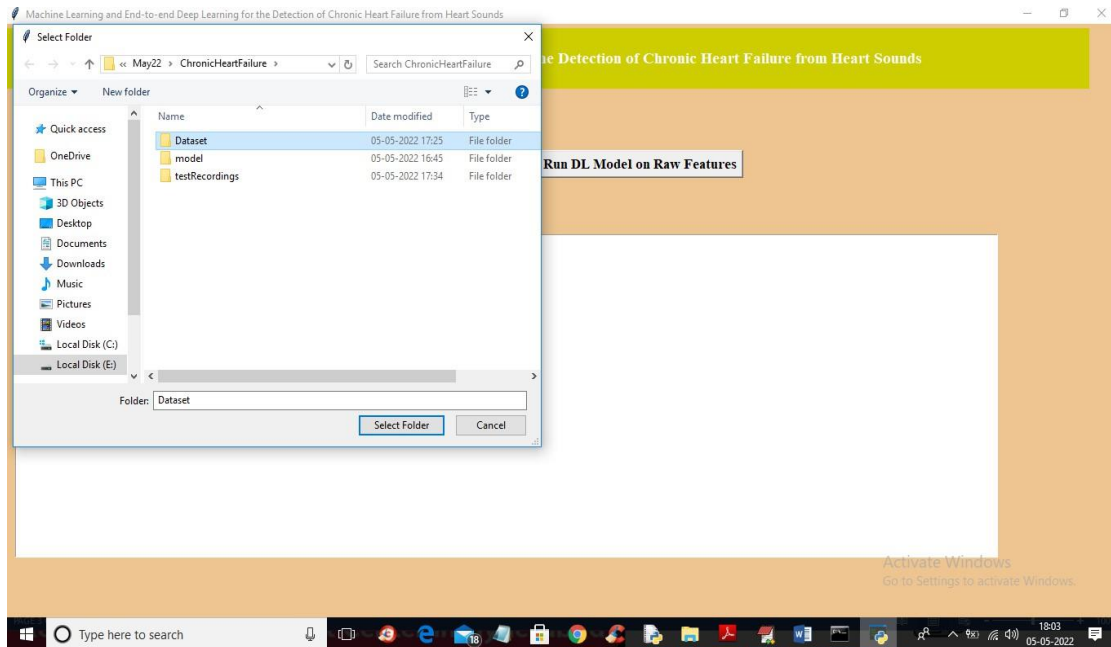
Output Screen Shorts

Screen Shorts

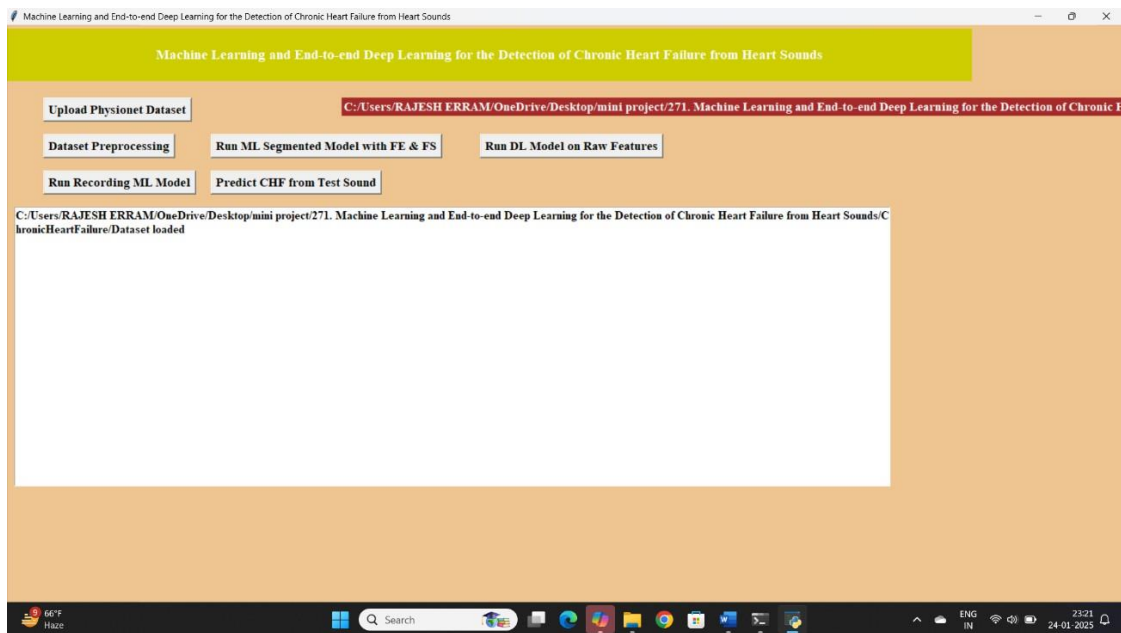
To run project double click on ‘run.bat’ file to get below screen



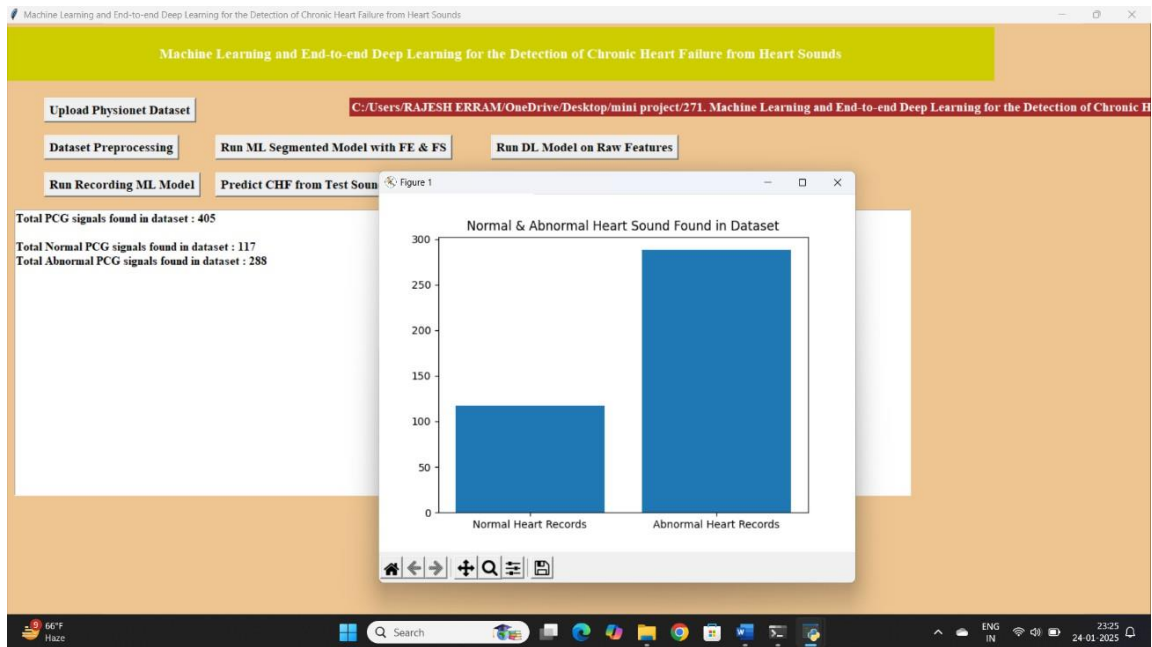
In above screen click on ‘Upload Physionet Dataset’ button to upload dataset



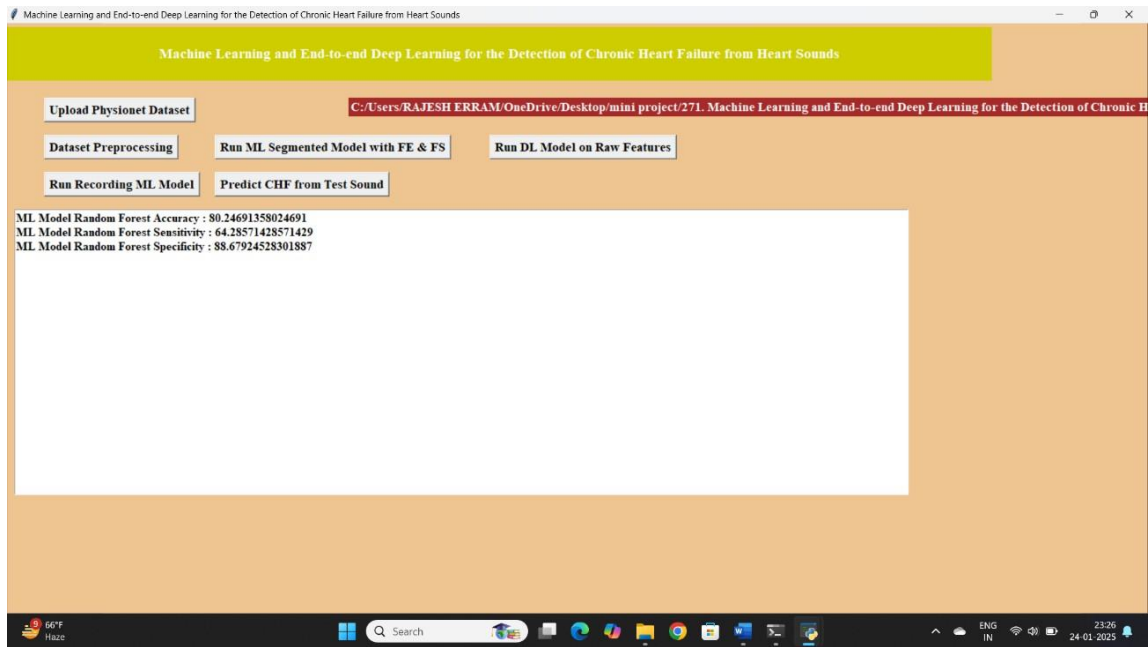
In above screen selecting and uploading 'Dataset' folder and then click on 'Select Folder' button to load dataset and to get below output



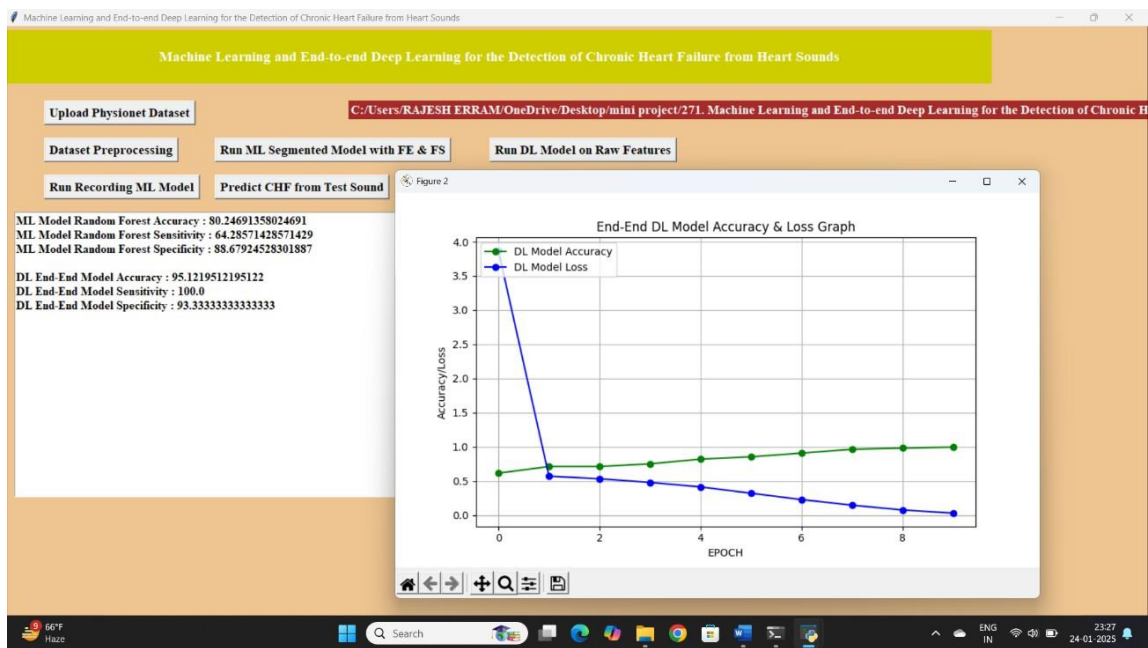
In above screen dataset loaded and now click on 'Dataset Preprocessing' button to read all dataset file and then extract features from it



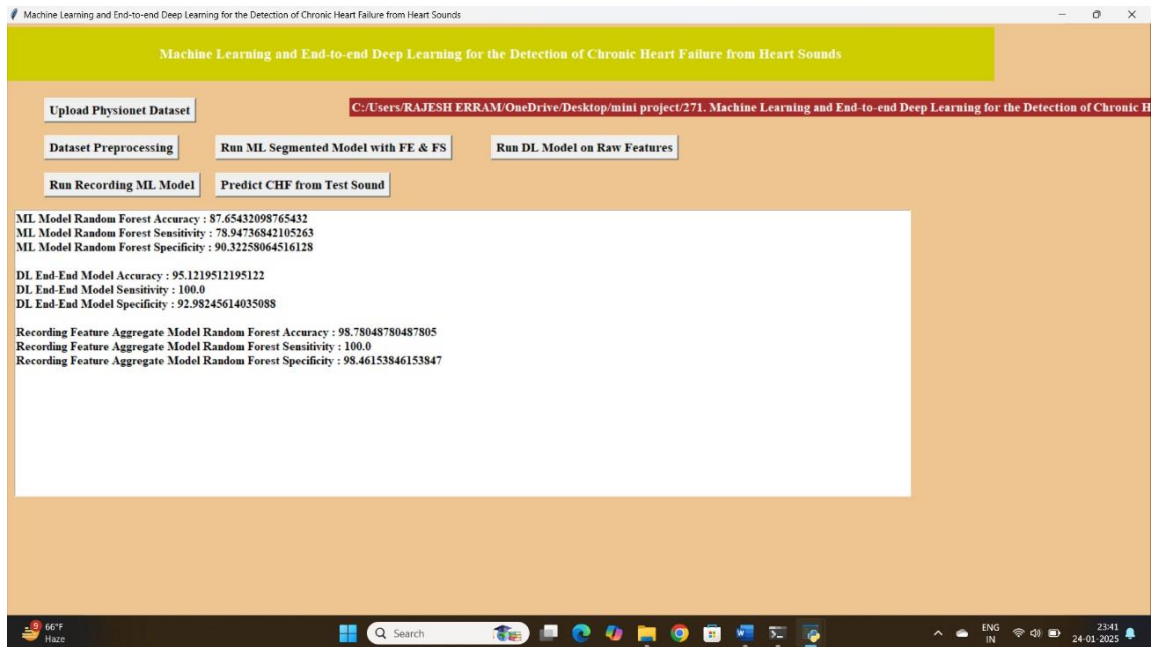
In above screen we can see dataset contains 405 heart sound files from 405 different person and 117 are the Normal sound and 288 are abnormal and in graph x-axis represents normal or abnormal and y-axis represents number of persons for normal or abnormal. Now close above graph and then click on 'Run ML Segmented Model with FE & FS' button to train Classic ML segmented model on above dataset and get below output



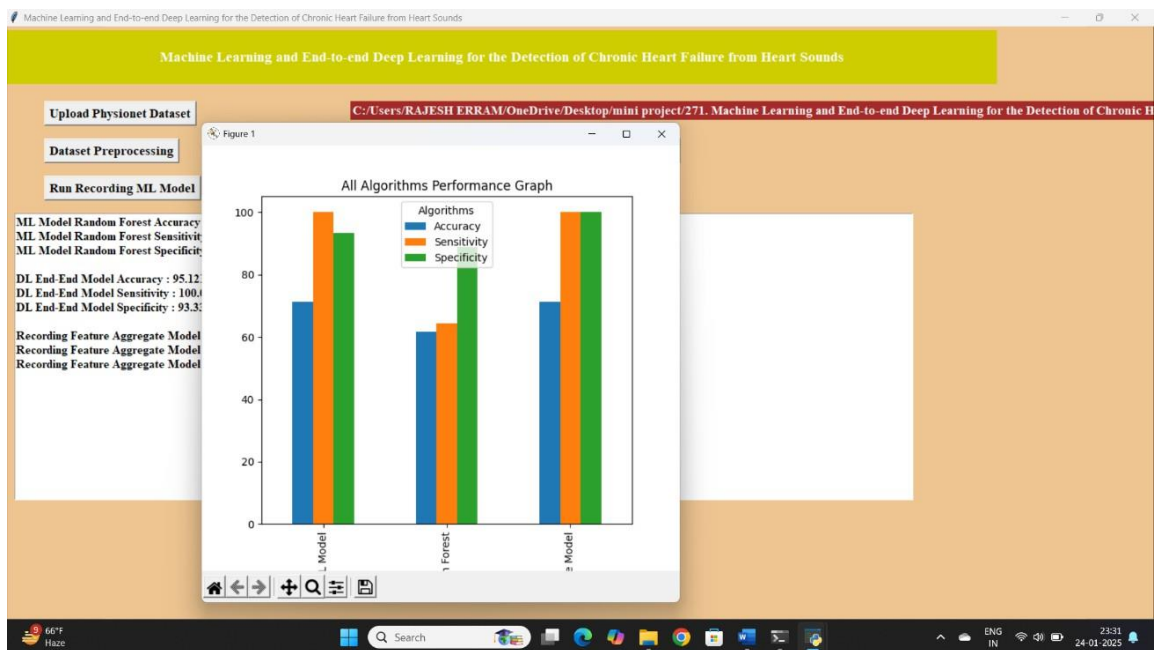
In above screen with Classic ML we got 90% accuracy and now click on ‘Run DL Model on Raw Features’ to get below output



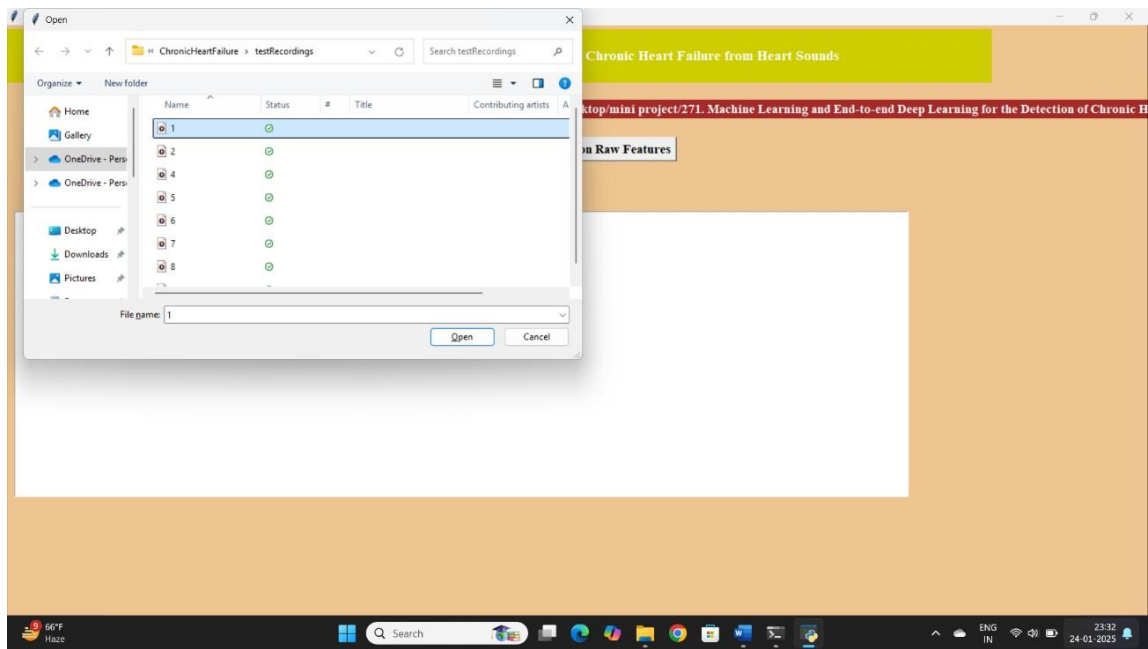
In above screen with DL model we got 93% accuracy and in graph x-axis represents epoch or iterations and y-axis represents accuracy or loss values and green line represents accuracy and blue line represents LOSS and we can see with each increasing epoch accuracy got increase and loss got decrease and now close above graph and then click on ‘Run Recording Model’ button to get below output



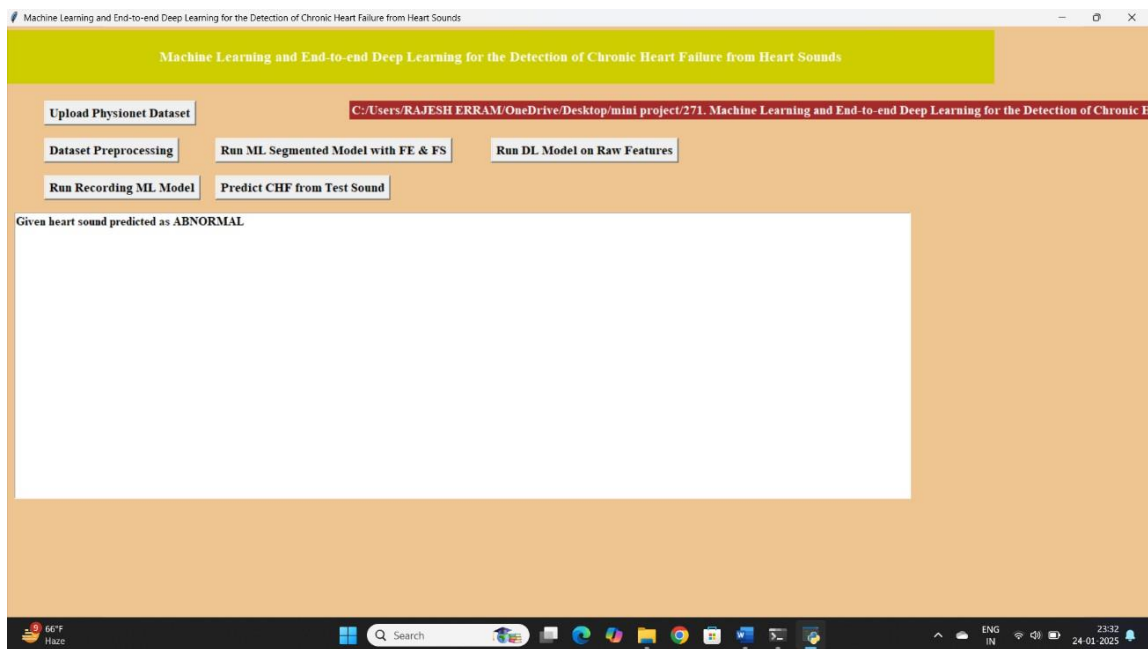
In above screen with recording model we got 96% accuracy and we can see all algorithms performance graph in below screen



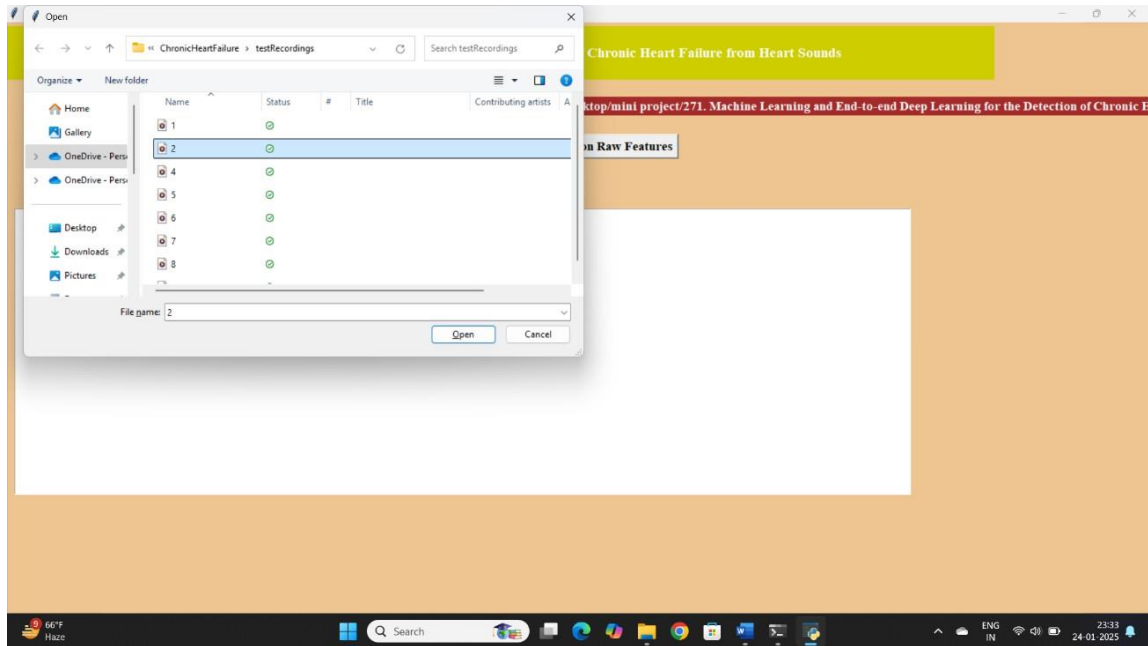
In above graph x-axis represents algorithm names and y-axis represents accuracy, sensitivity and specificity and in all algorithms Recording model has got high accuracy. Now close above graph and then click on 'Predict CHF from Test Sound' button to upload test sound file and get predicted output as Normal or Abnormal



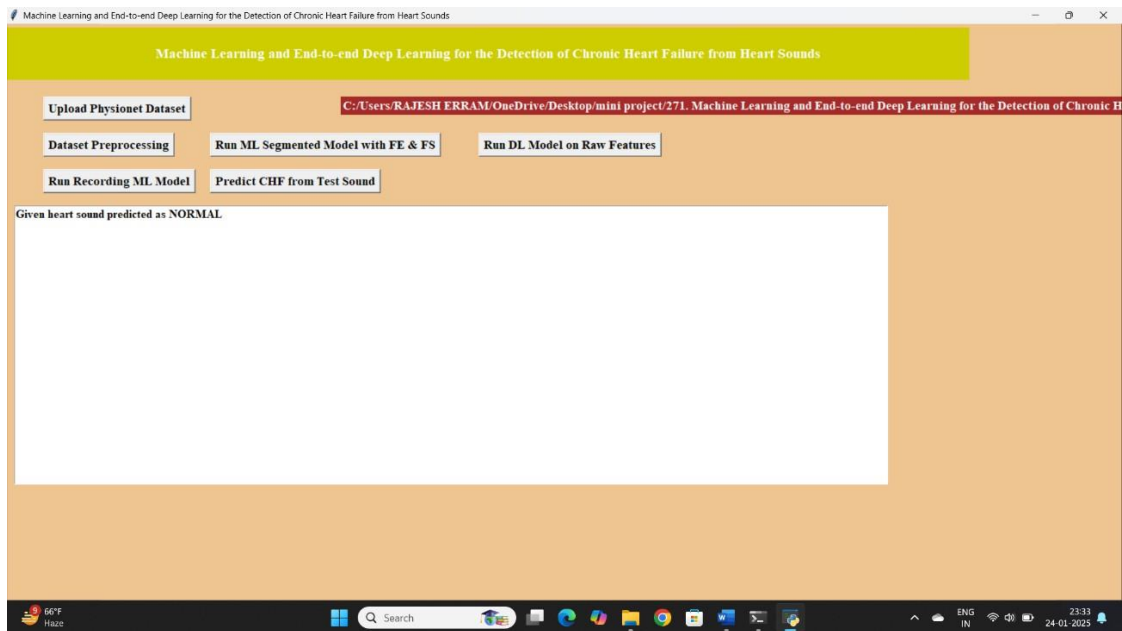
In above screen selecting and uploading '1.wav' file and then click on 'Open' button to get below output



In above screen uploaded heart sound file predicted as ABNORMAL and similarly you can upload other files and test



For '2.wav' file below is the output



Chapter-10

Conclusion

In this developed algorithm, a deep learning-based classification technique is applied to classify the PCG recorded heart sounds as normal/abnormal also in abnormal subtype as murmur, artifact and extrasystole. MFCC is used to remove the noise that gets recorded along with the heart sound features. Various statistical features are extracted from filtered heart sound signals. In the proposed work, the features were strategically considered which are likely to be affected during an abnormal heart functioning. Those features improved the performance of deep learning algorithms in terms of time complexity and accuracy. The proposed method achieved 94.4% accuracy for abnormal and normal heart sound classification when tested on heart sound database. The proposed model of automated diagnosis shows high response with low time complexity as compared to existing method of classification.

References

- [1]. Mendis S, Puska P, Norrving B, World Health Organization (2011) Global atlas on cardiovascular disease prevention and control (PDF). World Health Organization in collaboration with the World Heart Federation and the World Stroke Organization, pp 3–18. ISBN: 978-92-4-156437-3
- [2]. Wilks S (1883) Evolution of the stethoscope. Popular Science 22(28):488–491
- [3]. Leatham A (1975) Auscultation of the heart and phonocardiography. Churchill Livingstone, London
- [4]. Azmy MM (2015) Classification of normal and abnormal heart sounds using new mother wavelet and support vector machines. In: 2015 4th international conference on electrical engineering (ICEE), Boumerdes, pp 1–3. <https://doi.org/10.1109/intee.2015.7416684>. December 2015
- [5]. Vadicherla D, Sonawane S (2013) Decision support system for heart disease based on sequential minimal optimization in support vector machine. Int J Eng Sci Emerg Technol 4(2):19–26