HEART DISEASE PREDICTION USING ML ALGORITHMS

Project report submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

(RIYA GUPTA (191425))

Under the supervision

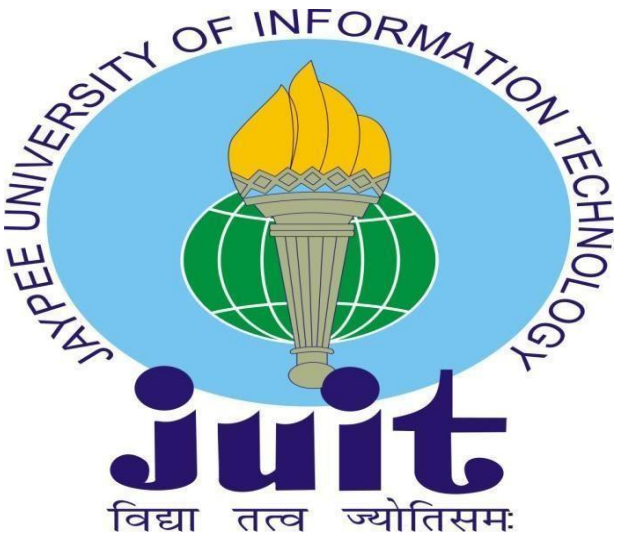
of

(Dr.Sunil Datt Sharma)
Assistant Professor, SG

(Dr. Yugal Kumar)

Associate Professor

to



Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234
Himachal Pradesh

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Heart Disease Prediction using Machine learning algorithms"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of **(**Dr. Sunil Datt Sharma**)** (Assistant Professor,SG (ECE))and Dr.Yugal Kumar(Associate Professor,(CSE)).I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Cloud Computing.**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Riya Gupta,191425

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor 1 Signature)
Dr. Sunil Datt Sharma
Assistant Professor, SG
ECE
Dated:

(Supervisor 2 Signature)
Dr. Yugal Kumar
Associate Professor
CSE
Dated:

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Sunil Datt Sharma,** Assistant Professor (SG), Department of ECE and **Dr.Yugal Kumar**(Associate Professor,(CSE),Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of " **Machine learning"** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Sunil Datt Sharma,** Department of ECE and **Dr.Yugal Kumar**,Department of CSE for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non- instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Riya Gupta (191425)

Table of Content

List of Abbreviations

MFCC : Mel- Frequency cepstral coefficients

CNN : Convolutional Neural Networks

KNN: K Nearest neighbor

DT : Decision Trees

RF: Random Forest

RNN : Residual Neural Networks

UCI : UC Irvine Machine Learning Repository

DFT: Discrete fourier transform

NB: Naïve bayes

VGG16 : Visual geometry group

IRB :Institutional Review Board

WHO: World Health Organization

List of Figures

List of Graphs

List of Tables

Abstract

We all know that heart is the organ which pumps blood throughout our body. It is the primary organ of our circulatory system. If it fails to work properly, then the brain and various other organs will stop working, and within a few seconds the person will die. In our project Our Aim is to build the best predictive model using various machine learning algorithms or neural networks which gives the highest accuracy in predicting whether a patient has a heart disease or not using the patient's data. We designed two approaches: In our first approach we are using Cleveland Heart Disease dataset which is a standardized dataset taken from the University of California Irvine (UCI) Repository. This dataset contains a total of 76 attributes but all previous research and published experiments refer to using a subset of 14 of them. we implemented some of the machine learning algorithms like Random Forest, Naïve Bayes, Decision trees, KNN. So far Random Forest Classifier has given most significant results with accuracy of 93.84%, this was achieved by the tuning of the hyperparameters of the algorithms with metric='Manhattan', n_neighbors=13, weights='distance', n_jobs=-1 at last these parameters values were used for extraction of the results. Now here comes our second method, we introduced heart disease detection based on heart sounds. The proposed method employs three successive stages, like spectrogram generation, deep feature extraction, and classification. In the spectrogram generation stage, the heart sounds are converted to spectrogram images by using time–frequency transformation. Our method also aims to classify an abnormal heartbeat with a normal heartbeat based on audio data recorded from a stethoscope. Audio data used is of wav type (Waveform Audio File Format). The classification is conducted using Convolutional Neural Network. Here we also studied Mel spectrogram and MFCC. In this, The epoch values used were 100,150, 200, 250 and 300.

The best results were obtained with 300 epochs at 0.001 learning rate applied on batch size of 128. The training accuracy is 89.73%, while the testing accuracy rate is 82%.

Chapter-1

INTRODUCTION

1.1 Introduction

The body's main pump for moving blood around is the heart which is the primary organ of our circulatory system.If it fails to function properly, the brain and other organs will stop operating, and the individual will die within a few seconds. Possible symptoms include chest pain, loss of appetite, shortness of breath, fainting, and severe weakness. If you are having these symptoms, we recommend that you see a doctor to avoid heart failure. Work- related stress and poor eating habits also contribute to the rise in the prevalence of numerous heart diseases. Cardiovascular Diseases (CVDs) kill 17.9 million people each year, according to WHO, and constitute a major hazard in today's society.[3] Heart failure accounts for more than five out of every six CVD deaths, with one-third of these deaths occurring in adults under the age of 70.

One of the greatest burdens of cardiovascular disease (CVD) is found in India. The annual mortality from cardiovascular diseases in India is expected to increase from 2.2 million in 1990 to 4.7 million in 2025 (2020). In recent years, the incidence of coronary artery disease in India has increased from 1.6% to 7.4% in the provincial population and further increased from 1% to 13.2% in the metropolitan population.[3]

Interheart research found that Indians are more likely to have cardiovascular risk factors such as abdominal fat. High blood pressure and diabetes than other ethnic groups, even at a young age. In recent years, the prevalence of cardiovascular risk factors in India has expanded rapidly, particularly in the metropolitan areas. The reasons for the problem of high risk factors are unclear and not fully studied. Complementary studies thus provide unbiased assessments of gating-response associations and advance our understanding of factors that lead to cardiovascular disease.[4]

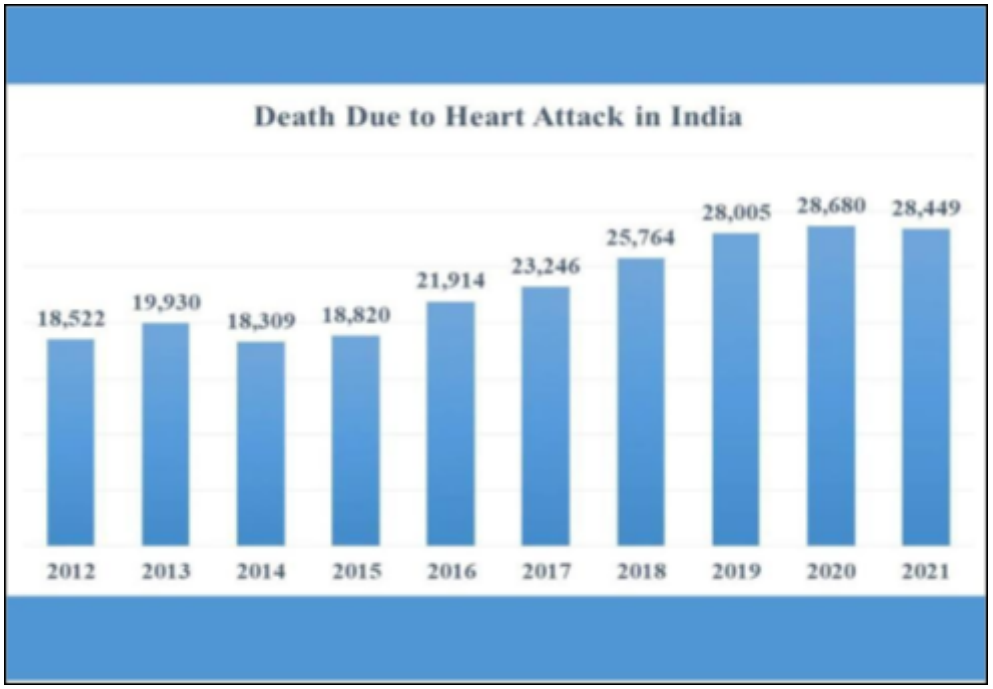


Figure 1. Heart attack deaths

Heart attack deaths over the years

In 2021 there was no significant increase in deaths from coronary heart failure more than in 2019. The number of heart attacks in 2021 will be 1.6% higher than in 2019. The number of deaths from respiratory failure in the. In 2021 it was lower than in 2020. Between 2017 and 2021, the number of deaths from respiratory arrest increased by 22%.

According to the NCRB, in the last few years, from 2012 to 2021, there was a 54% increase in cases of respiratory failure. AI is a subclass of man-made intelligence (A), i.e. H. he focuses on planning programs that input information from him and work on his presentation without being able to. The AI calculations used to create the model use distinctive pattern and element tests using abundant accessible information. Forecasts and decisions are made based on this information.[4]

By applying the best calculations and preparing them properly, accuracy rates and execution can be increased. Specialists and doctors use different tests to analyze different diseases, which can be confusing, prone to human error, and

tedious. To solve these problems, an emotional clinical decision support network can be used that uses information mining and AI techniques such as Irregular Treeline, K-Nearest Neighbor (K-NN), Guileless Bayes, and Choice Trees for diseases with significant diseases and for minimal time. Price.

Due to the healthcare industry's critical expansion and growth, there is a wealth of information available that may be used to achieve these goals and collect helpful data to aid in making the right decisions.Various research papers have used machine learning calculations and predictive models to anticipate heart problems. This search proposes an AI model to predict the likelihood of coronary artery disease in a person using different techniques with varying accuracy rates. Blood flowing through the heart's chambers produces cardiovascular sounds. During the cardiovascular cycle, the heart valves simply open and close. Vibrations of these structures caused by blood flow produce audible cardiovascular sounds. Various stethoscopes are available to check for concussions in the heart, lungs or other organs. Auscultation is a convenient condition for this type of item. The division of heart sounds is the most problematic part of heart sound characterization. As clinical students and unskilled specialists made several times more errors than experienced specialists when examining heart sounds, calculations based on machine learning began to make the auscultation system programmed and reliable. In order to apply AI calculations on a PC, PKG or ECG information must be examined and broken down.

Since we are working with signals, we use signal data sets. Along those lines, to use this we create a spectrogram, which is a visual representation of a signal's signal strength, or "noise," over time at various frequencies contained within a given waveform. In reality, spectrograms are generally used to address the frequencies of sound waves produced by people, devices, and other objects that are picked up by receivers.[5] In order to identify and group specific types of earthquakes on Earth, spectrograms are increasingly being used to examine the repeat content of consistent signals recorded by individual seismometers.

1.2 Problem Statement

The aim of this research is to find the most important outcomes for heart disease prediction utilising multiple Machine Learning approaches. On the dataset, we used techniques such as KNN, Logistic Regression, Random Forest, Naive Bayes, and Decision Tree to achieve this goal. This paper discusses data mining and machine learning methodologies. These findings will aid medical professionals and patients in the future by avoiding costly and time-consuming examinations.

The second problem statement is that we are essentially performing three tasks: signal to image conversion, deep feature extraction, and classification. Librosa will be used to analyse signals. It is an open-source Python audio analysis package. This type of library can provide us with data as well as the sample rate. The sample rate is the number of audio samples per second. For example, if we have a 30 second audio file, we extract values every 10 seconds.

We are unable to render the audio file in its original format, which is difficult to decipher. As a result, for a better understanding of this data, a standard representation approach must be used. For the generation of the input visuals, the spectrogram of the input heart sound waveforms is created. It is a method of converting ECG impulses into coloured visuals. The spectrogram images are created using the Short Time Fourier Transform (STFT).[5]

1.3 Objectives

- To build the best predictive model using various machine learning algorithms which gives the highest accuracy in predicting that a patient has a heart disease or not using the patient's data.
- We will be testing our algorithms on different training-testing samples.
- To analyse different feature selection methods on the dataset.
- To assess the importance of distinct features for predicting the disease
- The objective of our project is to develop and deep learning classification models based on spectrogram generation.
- We will be studying the use of the fourier transform which is a very much important characteristic of a signal;it transforms the signal sampled in time.
- To study different types of spectrograms like mel spectrogram,MFCC,Mel Scale.
- To study the use of signal processing with machine learning,which will basically handle information content in signals.
- The classification of normal vs abnormal heart sounds is proposed so as to classify heart audio into one of the following categories like artifact,normal,murmur,extrasystole,extra heart sound.

1.4 Methodology

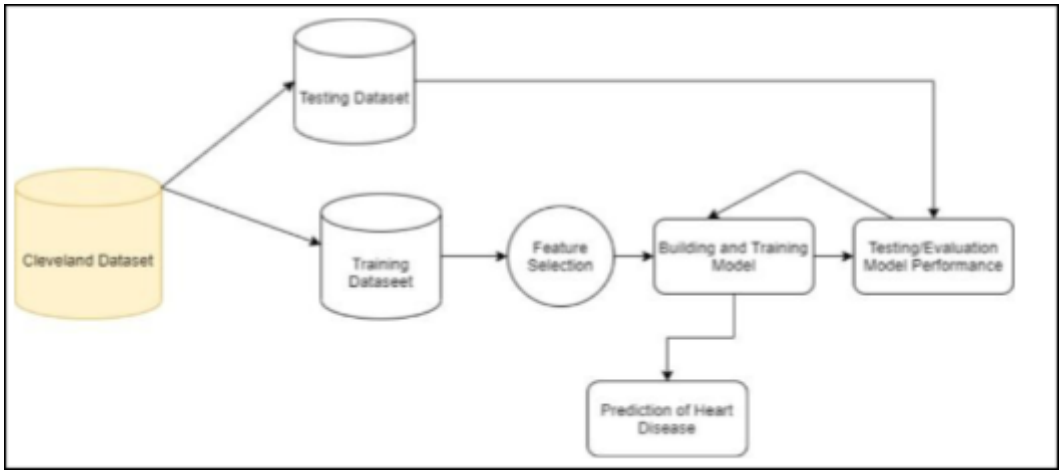


Figure 2: Methodology-ML algos

The first step is to achieve the best results, the dataset will be sampled. Following that, the dataset will be pre-processed. This covers cleaning the data set of missing and null values with various Python-coded procedures. Following the splitting of the dataset into training and testing datasets, the third step will be standardisation of the dataset.

After that, the model is trained on the training set and tested on the test set. We will get the optimum settings for the best results after modifying the dataset's hyperparameters. We will next retrain the model with those parameter values and examine the results. The flow of the dataset is depicted in the image below.

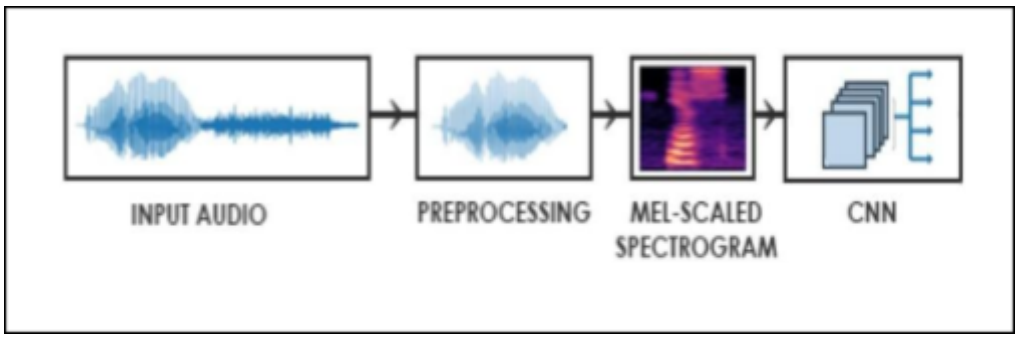


Figure 3. Framework for CNN and audio

Overview of the proposed sound-based heart disease recognition system The system is made up of three major components: input heart sound, processing, and spectrogram creation. Various deep feature set combinations are investigated in order to determine which deep feature set concatenation is best for effective heart sound classification. For the generation of the input images, the spectrogram of the input heart sound waveforms is calculated. The CNN model is used for feature extraction.

- Spectrogram images are the input for the feature extraction architecture.
- Convolutional layer (CONV) calculates the neuron outputs that are associated with the local regions in the input images. Each convolutional operation calculates a dot product between their weights and a small region.The output size of the layer is (width×height×N), depending on the number of filters (N) used in that layer.
- The ReLu layer is an activation function in which the CONV output is applied to the max(0,x) function and the output size of the ReLu layer does not change according to the output size of the CONV layer.

1.5 Organization

we have organized our project into two parts-

i) **In our first approach**, we have used ml models to predict the accuracy on Cleveland dataset of heart patients to be able to recognize whether the patient is having heart disease or not. In this we are doing these steps as follows-

Data Pre-processing

Real-world data contains large amounts of data, missing data, and noisy data. These data have been preprocessed in order to avoid problems and produce reliable forecasts. When cleaning collected data, noise and missing values are prevalent. These data need to be cleaned of noise and have any missing values filled in order to draw a reliable and accurate conclusion. To make data more understandable, transformation modifies the format of data from one type to another. It entails operations like aggregation, normalisation, and smoothing.

Feature Sampling

We chose 14 of the 76 variables included in the Cleveland Dataset that were previously employed by researchers for the building of the same model. By picking these values, we will be able to compare the observations fairly.

Data Cleaning

After we have finished feature sampling the dataset, we will clean it of missing and null values. According to the description for the UCI Repository for Heart Disease Cleveland Dataset, missing values are denoted as -9 in the dataset. To fill in the blanks, we'll utilize the carry forward approach, which fills in the values with the previous values in the column.

Standardization of Data

Standardization is a technique that is commonly performed on datasets before using Machine Learning Algorithms to standardise the range of the data. The dataset contains some features that are only available in a specified range, such as Chest Pain Type (1,2,3,4), and some values, such as Fasting Blood Sugar, that range from lowest to maximum. So, in order to achieve better results, we applied the Standardization approach for this dataset. After preprocessing, the data we used is standardised using the Python Sklearn Library. The standard score of a sample x can be determined by the formula:

z = (x - mu) / sigma

where mu is the mean of the training samples and sigma is the standard deviation of the training samples.

Training Testing Split

To train and test a model, we need two datasets: the training dataset to train the model and the test dataset to test the model. The optimal split ratio is not defined; it strongly depends on the data set, the problems, etc. However, be careful and avoid overfitting the data. We use a 70- 30,80-20,90-10 split of all the data we have. For example 70% of the data is used for training. while 30% is used for testing. To do this we used python train_test_split().

Methods and Techniques

After the data has been preprocessed, it is the time to apply the machine learning approach to the dataset. This will be accomplished with the Python Sklearn package. Machine Learning Algorithms are used to forecast numerous medical features; in this section, we will apply various machine learning algorithms to extract the optimum method that delivers the most accurate and precise prediction. The demand for an optimal model that produces meaningful outcomes is at an all-time high. To implement our model, we use a variety of Machine Learning approaches. Begin by importing the data collection and preparing it. First and foremost, in order to run these Machine Learning methods, we imported all of the major libraries that would be needed in the model's construction.

In our **second approach**, we will convert audio inputs to images before employing our cnn model. Following that, we shall categorise heart sounds into regular and pathological heartbeats.

Mr Peter Bentley [2] utilised this dataset in a Machine Learning challenge for the classification of pulse sounds. The dataset is separated into two sets based on the sources from which it was gathered. Set A (set a.csv) data was obtained from the general population using the iStethoscope Pro iPhone app, and Set B (set b.csv) data was acquired from a clinical trial in hospitals using the DigiScope digital stethoscope.

In this dataset there are 4 classes of heartbeat sounds:

1. Normal: healthy heart sounds

2. Murmur: extra sounds that occur when there is turbulence in blood flow that causes the extra vibrations that can be heard

3. Extrahls: heartbeats with an additional sound

4. Extrasystoles: are additional heartbeats that occur outside the physiological heart rhythm and can cause unpleasant symptoms

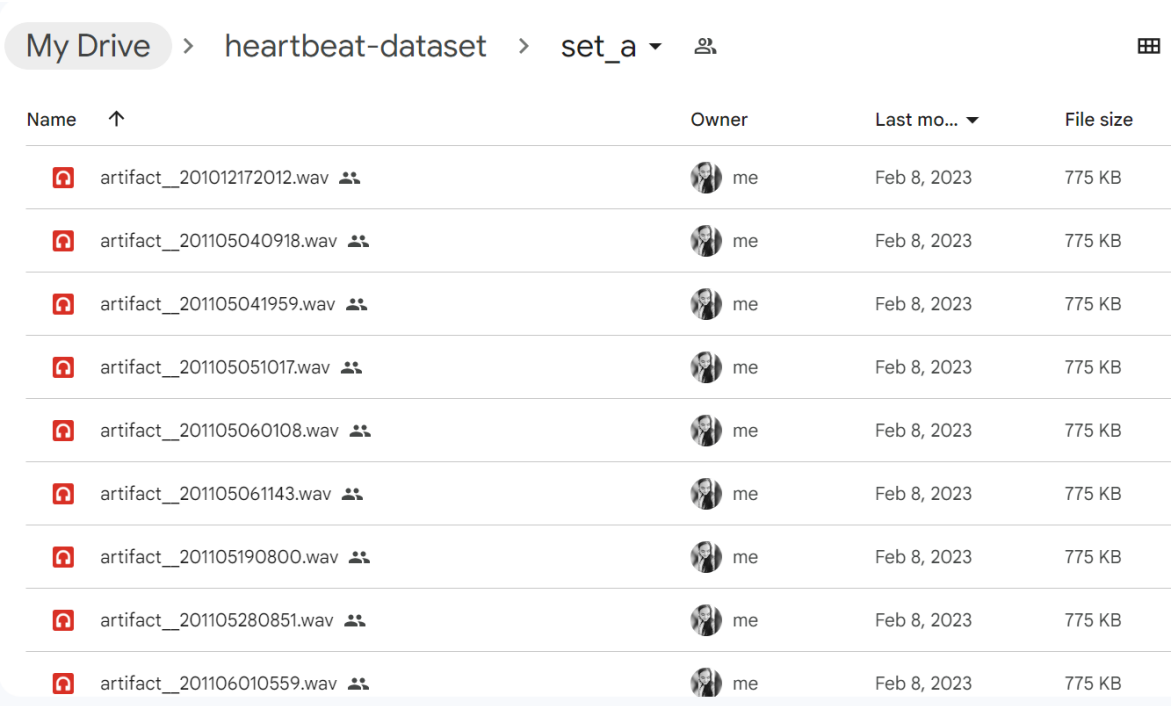


Figure 4. Data Collection

We were needed to collect heartbeats from nearby clinics and hospitals as part of the study in order to evaluate our model. However, the two hospitals we spoke with denied authorization on the basis of obtaining IRB approval. The Institutional Review Board (IRB) is in charge of reviewing behavioural research involving human subjects. The IRB approves and supervises all human-participant research to verify that the experiment or survey is structured to address legal and ethical concerns while posing no undue hazards to the participants. Before recruiting and data collection can begin, all proposed study involving "human subjects" must be evaluated and authorised by the IRB. The IRB ensures that the participants are adequately informed of what their participation entails (e.g.: risks and benefits). This includes written and signed informed consent in most cases. Here we will be dealing with different types of spectrograms like mel spectrogram,MFCC

Chapter-2

LITERATURE SURVEY

As previously stated, this study was divided into two categories, thus we employed several research articles to address the objectives. The works were published in the journals Health Information Science and Systems, Research Gate, SN Computer Science, and IEEE.

Heart disease is currently one of the leading causes of death worldwide. CVDs are responsible for one-third of all fatalities in 2019, according to the Journal of the American College of Cardiology. Heart disease was the biggest cause of death in India, Russia, the United States, and Indonesia last year. From 1990 to 2019, the prognosis of heart disease patients doubled, from 271 million to 523 million, and the overall number of fatalities from heart failure climbed from 12.1 million to 18.6 million.

Heart Disease Prediction using Machine Learning Techniques (SN Computer Science)

Heart disease, also known as cardiovascular disease, refers to a group of diseases that affect the heart and are the leading cause of death worldwide. Data mining is a technique for processing large amounts of data, which is often used in medicine. His study focuses on heart disease and his models are built using supervised learning techniques such as decision trees, K-nearest neighbors, naive Bayes, and random forest algorithms. Using a dataset from their ICU repository of cardiac patients from the Cleveland database. There are 303 instances and 76 attributes in the data set. Only 14 of the 76 characteristics were selected for the test. His research also attempts to map the likelihood of people developing heart disease. The results show that the nearest neighbors of K have the highest accuracy values.[6]

Heart Disease Prediction using Machine Learning Techniques (IEEE)

Several studies have found that machine learning techniques have accelerated the health sector. As a result, the goal of this research is to develop an ML model for heart disease prediction based on the relevant parameters. For this study, we employed a benchmark dataset of UCI Heart illness prediction, which includes 14 different factors linked to Heart Disease. Models have been developed using machine learning algorithms such as Random Forest, Support Vector Machine (SVM), Naive Bayes, and Decision Tree. In our research, we also attempted to uncover correlations between the various qualities present in the dataset using standard Machine Learning methods and then use them efficiently in the prediction of the likelihood of Heart disease. The results reveal that, when compared to other ML approaches, Random Forest predicts with more accuracy in less time. As a decision support system, this model can be useful to medical practitioners in their clinic.[7]

Towards the classification of heart sounds based on convolutional deep neural network(Information Science and Systems)-{Fatih Demir, Abdulkadir Şengür)

They removed several deep features from the previously prepared CNN models and used SVM classifiers to productively characterize the heart sound in this scan. His strategy first fully shifts cardiovascular sound waves into different spectrogram images, which are then processed through CNN slices for highlight extraction. Three previously prepared CNN models are used in highlight extraction: AlexNet, VGG16 and VGG19. The combination of the determined component vectors results in vectors of productive elements.[8]

Abnormal Heart Rhythm Detection Based on Spectrogram of Heart Sound using Convolutional Neural Network (Research gate){ Satria Wibawa. I Md. Dendi Maysanjayaj}

They overlooked several fundamental qualities of previously constructed CNN models and used SVM classifiers to effectively group heart sounds in this study. His strategy first fully shifts cardiovascular sound waves into different spectrogram images, which are then processed through CNN slices for highlight extraction. Three previously prepared CNN models are used in

highlight extraction: AlexNet, VGG16 and VGG19. Joining the determined component vectors results in invalid component vectors.[9]

Table 1- Comaprison of studies

Author(s)	Journal/Conference, year	Published By (IEEE, Elsevier, Springer)	Methodology
Devansh shah Samir Patil	SN Computer Science,2020	Springer	KNN-90.78 DT-80.26 Random Forest-86.84% Naive Bayes-88.15%
Vijeta Sharma,Shrinkhala Yadav	2nd Conference,2020	IEEE	SVM Random Forest Decision Trees Naive Bayes
Fatih Demir Abdulkadir Sengür	2019	Springer	spectrogram generation deep feature extraction and classification. Used AlexNet,VGG16,VGG19
Made Satria Wibawa I Md. Dendi Maysanjaya	6th conference,2018	IEEE	Audio data used wav type classification is conducted using CNN performed many epochs like 25,50,75,100,125,150

CHAPTER-3
SYSTEM DESIGN AND DEVELOPMENT

The Cleveland Heart Disease dataset, which is a standardised dataset from the University of California Irvine (UC) Repository, was used in this investigation. This dataset has 76 properties in total, however all previous research and published trials have only used a subset of 14 of them. The primary purpose of using this dataset is to determine whether the patient has heart disease. Cleveland is a binary dataset that has 303 instances and 76 attributes. This dataset contains missing values as well, thus it must be pre-processed before prediction. The Cleveland dataset tests focused on identifying the presence of disease (value 1) from the absence of cardiac disease (value 0).[1]

Data Set Features

Attributes description of Cleveland Dataset

Cleveland Dataset consists of a total of 14 attributes with one target attribute which is used in diagnosis of heart disease.[1]

Table 2: Attributes description of heart dataset

S.No	Attribute	Code	S.No	Attribute	Code
1	Age	age	7	Resting ECG	restecg
2	Sex	sex	8	Maximum heart rate achieved	thalach
3	Type of chest pain	cp	9	Exercise-induced angina	exang
4	Resting blood pressure	trestbps	10	Old peak = ST depression induced by exercise relative to rest	oldpeak
5	Serum cholesterol	chol	11	Slope of the peak exercise ST segment	slope
6	Fasting blood pressure	fbs	12	Number of major vessels	ca
13	Thalassemia	thal	14	Diagnosis of heart disease	target

Brief description of attributes

- **Age**: This attribute defines the age of the individual person.
- **Sex**:This attribute shows the gender of the person using the below given format :

  1 = male

  0 = female

- **Chest-pain type**: This attribute displays the type of pain in the chest experienced by the person using the defined format i.e, 1,2,3,4 meaning typical angina,atypical angina,non — anginal pain and asymptotic respectively.

- **Resting Blood Pressure**: displays the "resting blood pressure value of an individual in mmHg (unit)"
- **Serum Cholesterol**: displays the "*serum cholesterol in mg/dl (unit)*"
- **Fasting Blood Sugar**: compares the "*fasting blood sugar*" value of an individual with "*120 mg/dl*".

If fasting blood sugar > 120mg/dl then : 1 (true)

else : 0 (false)

- **Resting ECG** : displays resting "*electrocardiographic*" results

  0 = normal

  1 = having ST-T wave abnormality

  2 = left ventricular hypertrophy

- **Max heart rate achieved** : displays the "*max heart rate achieved*" by the person.

- **Exercise induced angina** :

  1 = yes

  0 = no

- **ST depression induced by exercise relative to rest**: displays the value which is an integer or float.

- **Peak exercise ST segment** :

  1 = upsloping

  2 = flat

  3 = downsloping

- **Number of major vessels (0–3) colored by fluoroscopy** : displays the value as integer or float.

- **Thal** : displays the "*thalassemia*" :

  3 = normal

  6 = fixed defect 7 = reversible defect

- **Diagnosis of heart disease:** Displays whether the person is having heart disease or not :

0 = absence

1 = present.

Random Forest

Data classification and prediction can be accomplished using the supervised machine learning method known as Random Forest. Its primary application is to tackle categorization difficulties." As we know, a forest is primarily made up of trees, thus more trees implies a more robust forest. A random forest method, on the other hand, constructs decision trees from data sets, obtains predictions from each, and then votes on the best solution. It is an ensemble strategy that differs from a single decision tree in that the outcomes are averaged to reduce overfitting.

Training Model(Without tuning of parameters)

We will first train the model using the training set to determine the score of the model. This is done by using RandomForestClassifier() from Python Sklearn.ensemble.

Hyperparameter Tuning of Random Forest.

The model's hyperparameters are tuned to extract the optimal parameter for training RandomForestClassifier (). To find the optimal parameter, we use a range of n estimators = [10, 100, 1000], max features = ['sqrt', 'log2'], criterion=gini', and n jobs = -1.

We discovered that the optimum settings are max features= sqrt and n estimator = 100 after adjusting the values.

Training the Model with tuned parameters.

Now we train the model using the best parameter values we got from the hypertuning of the model before i.e max_features= sqrt and n_estimator = 100 using the training dataset and testing them on the basis of their scores using the testing data set

K-Nearest Neighbour

It is a kind of supervised machine learning algorithm that can tackle classification and regression predicting problems. The KNN algorithm predicts the values of fresh data points based on 'feature similarities,' making sure that the fresh data point is assigned a value depending on how closely it resembles the fresh data points in the training collection.

Training Model(Without tuning of parameters)

We will first train the model using the training set to determine the score of the model.This is done by using KNeighborsClassifier() from Python Sklearn.ensemble.

Hyperparameter Tuning of K-Nearest Neighbours

The model's hyperparameters are tuned to extract the optimal parameter for training KNeighborClassifier (). To get the optimal parameter, we use a range of n neighbors = range(1, 21, 2), weights = ['uniform', 'distance'], metric = ['euclidean','manhattan','minkowski'] and n jobs = -1.

Training the Model with tuned parameters.

Now we train the model using the best parameter values we got from the hypertuning of the model before i.e metric='manhattan',n_neighbors=l1,weights="distance' using the training dataset and testing them on the basis of their scores using the testing data set.

Naive Bayes

"The Bayes Theorem is used to construct the Naive Bayes classifiers, a set of classification algorithms." It is a group of algorithms that all adhere to the same maxim: each pair of features being categorised is independent of the others. In Gaussian Naive Bayes, continuous values associated with each

function are assumed to have a Gaussian distribution.

Training Model(Without tuning of parameters)

We will first train the model using the training set to determine the score of the model. This is done by using GaussianNB() from Python Sklearn.naive_bayes.

Hyperparameter Tuning of Naive Bayes

The hyperparameter tuning of the model is done to extract the best parameter to train GaussianNB(). Here we are taking a range of 'var_smoothing': np.logspace(0,-9, num=100) to get the best parameter.

Decision Tree

It is a kind of supervised learning technique that can be applied to classification and regression issues, however it is most frequently used to address classification issues.In a tree-structured classifier, each node conveys the results while internal nodes provide dataset attributes and branches indicate decision laws.

Training Model(Without tuning of parameters)

We will first train the model using the training set to determine the score of the model. This is done by using DecisionTreeClassifier() from Python Sklearn.tree.

Hyperparameter Tuning of Decision Tree

The hyperparameter tuning of the model is done to extract the best parameter to train DecisionTreeClassifier(). Here we are taking a range of 'criterion': ['gini', 'entropy'], 'max_depth': [1, 2,3, 4, 5, 6, 7, 8], 'min_samples_split': [2, 3,4,5,6] to get the best parameter.

Training the Model with tuned parameters.

Now we train the model using the best parameter values we got from the hypertuning of the model before i.e criterion='entropy',max_depth=8,min_samples_split=2 using the training dataset and testing them on the basis of their scores using the testing dataset.

Logistic Regression

To calculate the likelihood of a reference variable, logistic regression, a supervised learning classification algorithm, is utilised. There are just two groups because the existence of the goal or dependent variable is dichotomous.

Training Model(Without tuning of parameters)

We will first train the model using the training set to determine the score of the model. This is done by using LogisticRegression() from Python Sklearn.linear_model.

Hyperparameter Tuning of Logistic Regression

The hyperparameter tuning of the model is done to extract the best parameter to train LogisticRegression() . Here we are taking a range of solvers = ['newton-cg', 'lbfgs', 'liblinear'],

Training the Model with tuned parameters.

Now we train the model using the best parameter values we got from the hypertuning of the model before i.e C=0.1,penalty='l2',solver='liblinear' using the training dataset and testing them on the basis of their scores using the testing data set.

Dataset used

This dataset was originally used in a Machine Learning challenge for the classification of heartbeat sounds by Mr Peter Bentley .

The dataset is divided into 2 sets depending on the sources from where it was collected:

- Set A (set_a.csv) data was collected from the general public via the iStethoscope Pro iPhone app.
    1. Normal
    2. Murmur
    3. Extra Heart Sound
    4. Artifact

- Set B (set_b.csv) from a clinical trial in hospitals using the digital stethoscope DigiScope.
    1. Normal
    2. Murmur
    3. Extrasystole

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1281 entries, 1241 to 1126
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   filename  1281 non-null   object
 1   label     1281 non-null   object
 2   offset    1281 non-null   float64
dtypes: float64(1), object(2)
memory usage: 40.0+ KB
```

Figure 5. Dataset Type

Requirements

We use Google colab for the implementation of the code. here are the hardware specifications of colab-

- 2vCPU @ 2.2GHz
- 13GB is the RAM
- 100GB Free Space
- idle cut-off is 90 minutes
- maximum takes 12 hours
- from 2020 onwards Updation:
- GPU instance downgraded to 64GB disk space.

Despite the fact that GPUs are capable of carrying out several calculations at once. This makes it possible to distribute training processes and considerably accelerate machine learning operations.

Pre-Processing

Basic shock is often present in heart sounds obtained by an advanced stethoscope and cellular phone receiver. Preprocessing of heart sounds is an important step in programmed study of heart rates. Since there weren't enough data points in audio beads shorter than three seconds for beat ordering to be reliable, we removed all of them. We split the heart sounds into 3-second chunks since the counts must first be condensed to a labeled word. We slice colossal records into a few more modest documents while protecting your distinguished name to expand the size of the record (i.e typical or weird) were also removed about a second earlier. We slice large documents into several more modest datasets, keeping their names unique (e.g. typical or uncommon). We also shave off a tiny bit of time at the beginning and conclusion of every sound recording because the amplifier's interaction with the body causes noise.

Feature extraction

The raw audio data was in the time domain and was in the amplitude vs time format (.wav type).We transformed the time-frequency distribution of this 1D time-series signal into a 2D heat map. The frequency spectrum of a signal is depicted in a spectrogram as it varies over time.

Since the amplitude ranges of the data were different due to the use of various instruments for transferring the data to the frequency domain produced more accurate results. For feature extraction in the frequency domain, we employed the following-

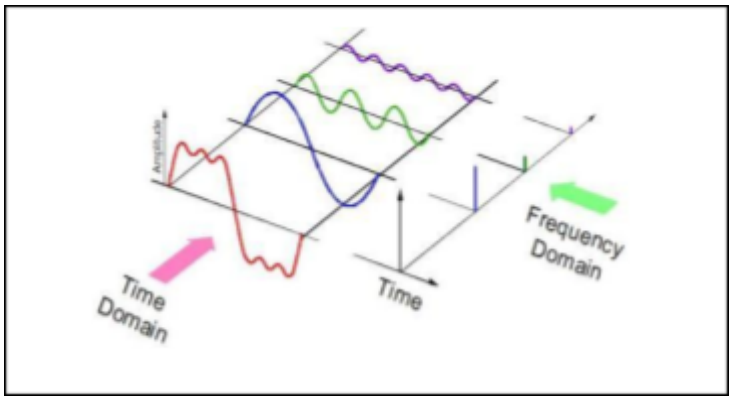


Figure 6. Frequency Domain

Mel Scale: It is the logarithmic transformation of signal's frequency.It mimics just like our own perception of sound.The formula to convert hertz scale to mel scale is-

$$m = 1127 \cdot \log\left(1 + \frac{f}{700}\right)$$

Melspectrogram: It represents an acoustic time-frequency representation of a sound. It is basically a spectrogram with the Mel Scale as its y-axis.

This Mel Scale is constructed such that sounds of equal distance from each other on the Mel Scale, also “sound” to humans as they are equal in distance from one another.

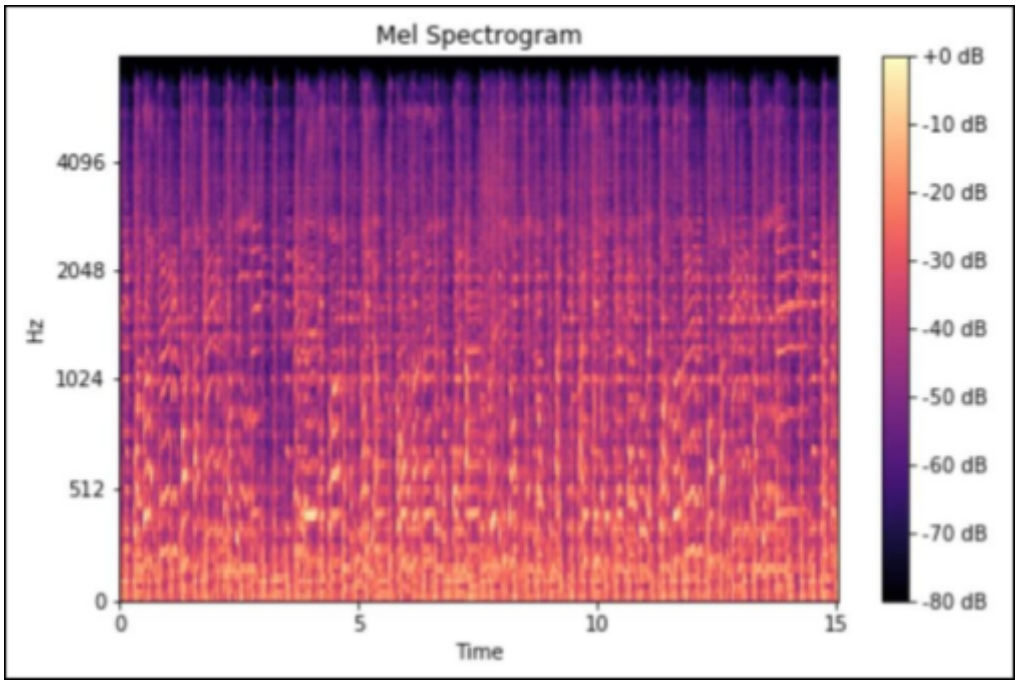


Figure 7. Mel Spectrogram

MFCC (Mel Frequency Cepstral Components) : We chose to use mel-frequency cepstrum coefficients to perform this transform because MFCC captures features from audio data that more closely resemble the human perception of loudness and pitch. MFCC is often used as a feature type for automatic speech recognition.**[4]** The roadmap for MFCC is as follows-

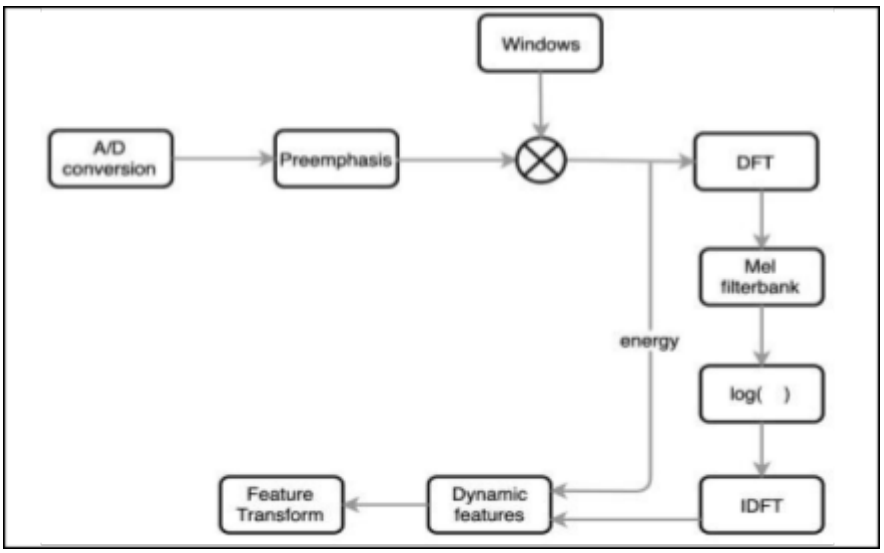


Figure 8. MFCC

Algorithm 1: MFCC feature extraction algorithm pseudocode
input : signal (Phonocardiogram signal)
output : MFCC (MFCC of phonocardiogram signal)
function MFCC (parameters)
Initialize parameters;
Split into frames phonocardiogram signals;
Apply **Hamming windowing** to frames;
Get spectrum by applying **Fast fourier transform** to all frames;
Determine matrix for a mel-spaced filterbank;
Transform spectrum to **mel spectrum**;
Obtain **MFCC vector** for each frame by applying **discrete cosine transform**;
end function

Figure 9. MFCC algo

Mathematical Analysis of MFCC

- Frame blocking :provides that sound signals are divided into short time intervals and framed
- Windowing is applied to minimize discontinuities located at the start and end points of each frame. In this,Hamming windowing function is one of the most suited windowing functions:

$$Y(m)=X(m)\, W_n(m), \qquad 0 \leq m \leq N_m - 1$$

Here Y(m) is the output signal, X(m) is the input signal, Nm is the number of samples within each window.

$W_n(m)$ represents the hamming window applied to the input signal:

$$W_n(m) = 0.54 - 0.46 \cos(2\pi m/(N_m - 1)) \quad , \quad 0 \leq m \leq N_m - 1$$

Now,FFT is used to transform the sample's time domain into frequency domain.

$$D_m = \sum_{m=0}^{N_m-1} \left(e^{-\frac{j2\pi km}{Nm}} D_k \right) \qquad k= 0,1,2,\ldots, N_m - 1$$

- M= 2595 log (1 + f / 700) —> This equation is used to convert the frequency to mel-frequency where f is frequency in hertz, M represents the mel frequency.
- Finally, work space is transformed from frequency domain to time domain by performing the inverse of Fourier transformation on all frames, and now the MFCC values are obtained as-

$$\text{MFCC} = \sum_{k=1}^{k} (\log D_k) \cos[\text{m}(\text{k} - 1/2)\, \pi/k\,] \qquad \text{m=0,1,…,k-1,}$$

Here m shows the number of coefficients, (log D_k) shows the log-energy output of the kth filter.

Now,MFCC feature vectors are obtained for each frame in the selected MFCC size.

*The output from MFCC is a matrix having feature vectors extracted from the frames.Here the rows→ the corresponding frame numbers, columns→corresponding feature vector coefficients *

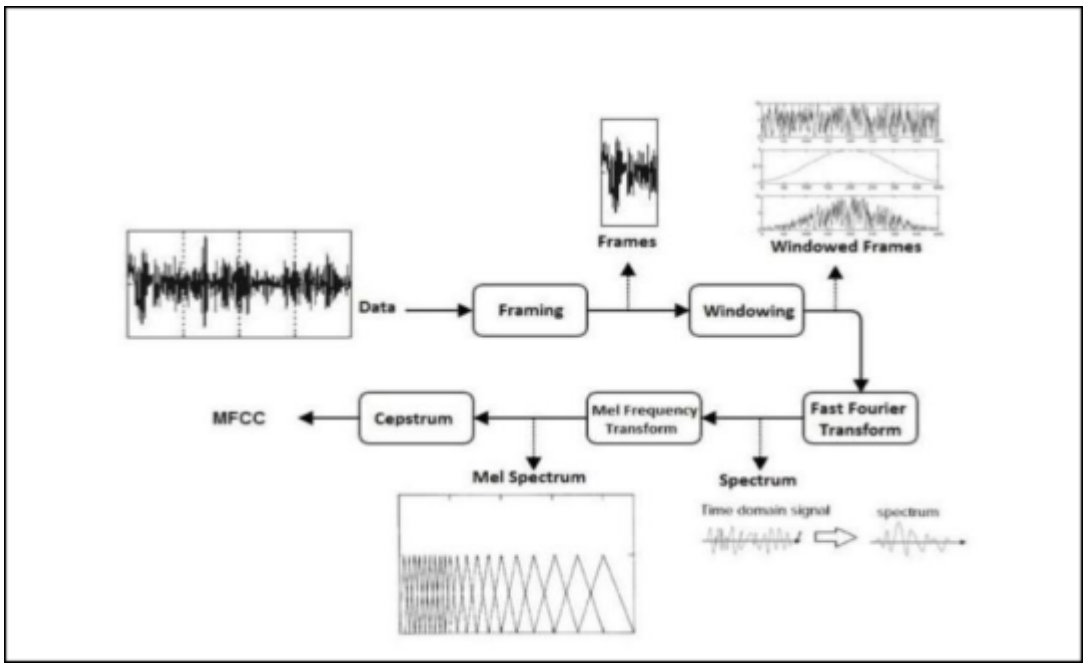


Figure 10. Process Flow to get the information in Frequency domain from time domain

Splitting of training set and testing set

Our Test data and the training data are from Feature Vectors which are obtained from MFCC, or we can say that these are cepstral vectors. This is the result obtained from implementing MFCC.

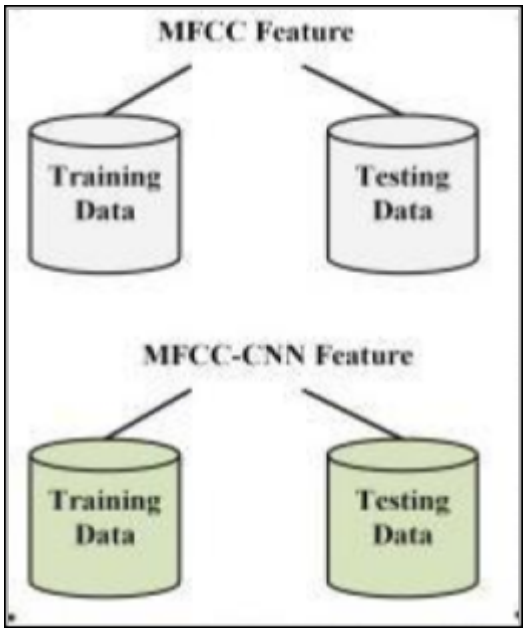
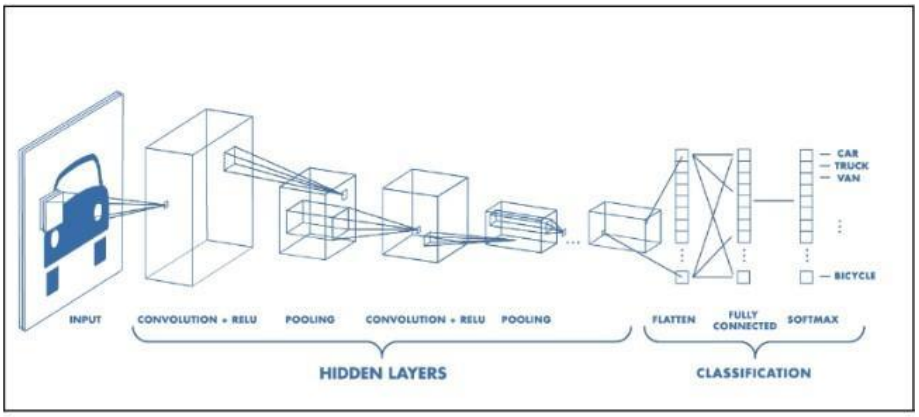


Figure 11.MFCC-CNN

CNN (Convolutional Neural Networks)

- CNN is a class of neural networks that specialises in processing data .
- Has a grid like topology , such as image.
- CNN has three layers—>convolutional layer,pooling layer,and a fully connected layer.



- Convolutional Layer is used to extract the various features from the input images.
- Pooling Layer decreases the size of the convolved feature map to reduce the computational costs
- Fully Connected Layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers.
- The Dropout layer which helps in preventing overfitting and sets input units to 0 with a frequency of rate at each step during training time.
- Hamming window function for signal or image filtering using a fast fourier transform.
- DCT cepstrum-DCT is simpler and faster than DFT and also FFT

- Mel energy-The energy features from an audio signal.
- FFT-an important measurement method for audio and acoustics measurement.
- Window size- It is the amount of time over which a wave form is sampled.

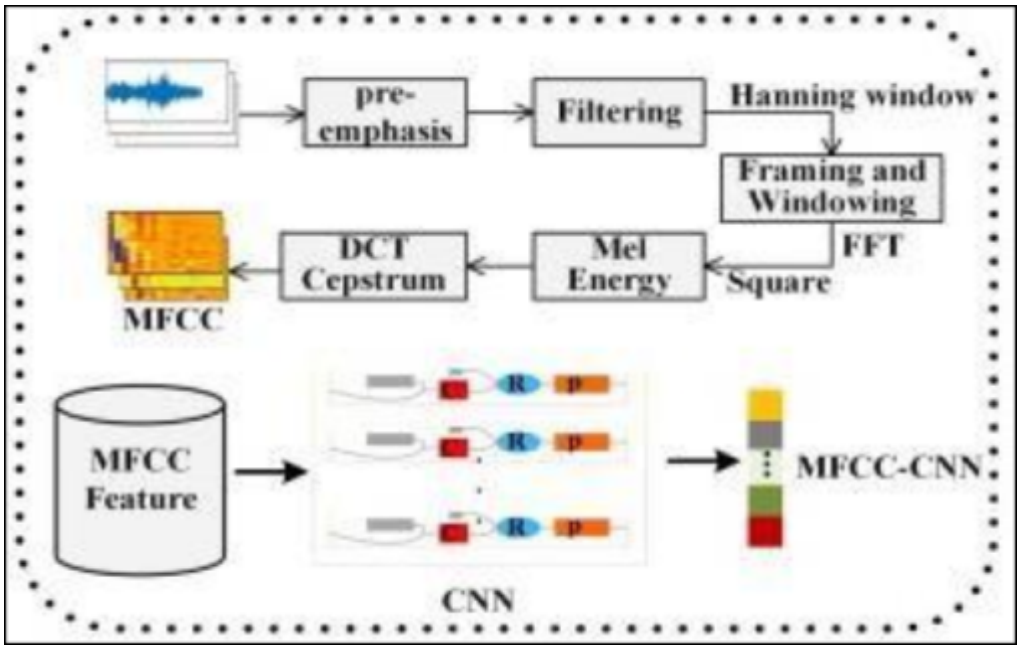


Figure 12. Model development

CHAPTER 4

PERFORMANCE ANALYSIS

The following libraries have been used for the implementation of the model and for training the data.

```
import pandas as pd
import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import plot_confusion_matrix ,confusion_matrix, precision_score, recall_score, f1_score

from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
```

Figure 13. Libraries used

Loading the dataset

```
df = pd.read_csv("heart.csv")
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Figure 14. Loading the dataset

Data Types

```
df.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

Figure 15. Data type

Description of Dataset

```
print("Description of Dataset")
df.describe()
```

Description of Dataset

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	th
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	3
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3

Figure 16. Description

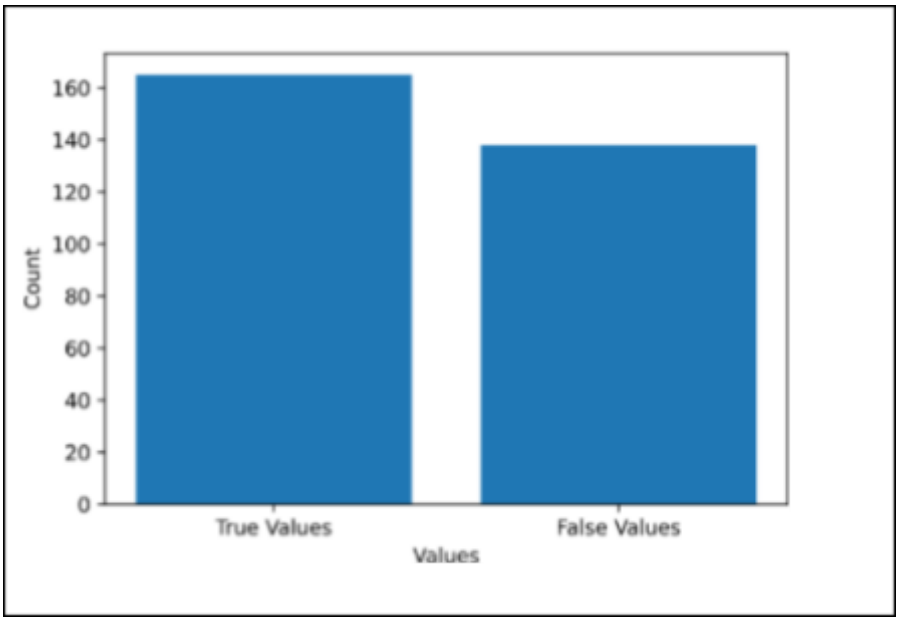


Figure 17. Visualisation of data

Output types

```
No = Y.unique()
print("Unique",No)
Y_total = Y.count()
Y_true = Y.sum()

Y_false = Y_total - Y_true
print("Positive Value = ", Y_true,", i.e :",round((Y_true/Y_total)*100,2),"%")
print("False Value = ", Y_false,", i.e :",100-round((Y_true/Y_total)*100,2),"%")
```

```
Unique [1 0]
Positive Value =  165 , i.e : 54.46 %
False Value =  138 , i.e : 45.54 %
```

Figure 18.Output types

Figure 19. Hypertuning RFC

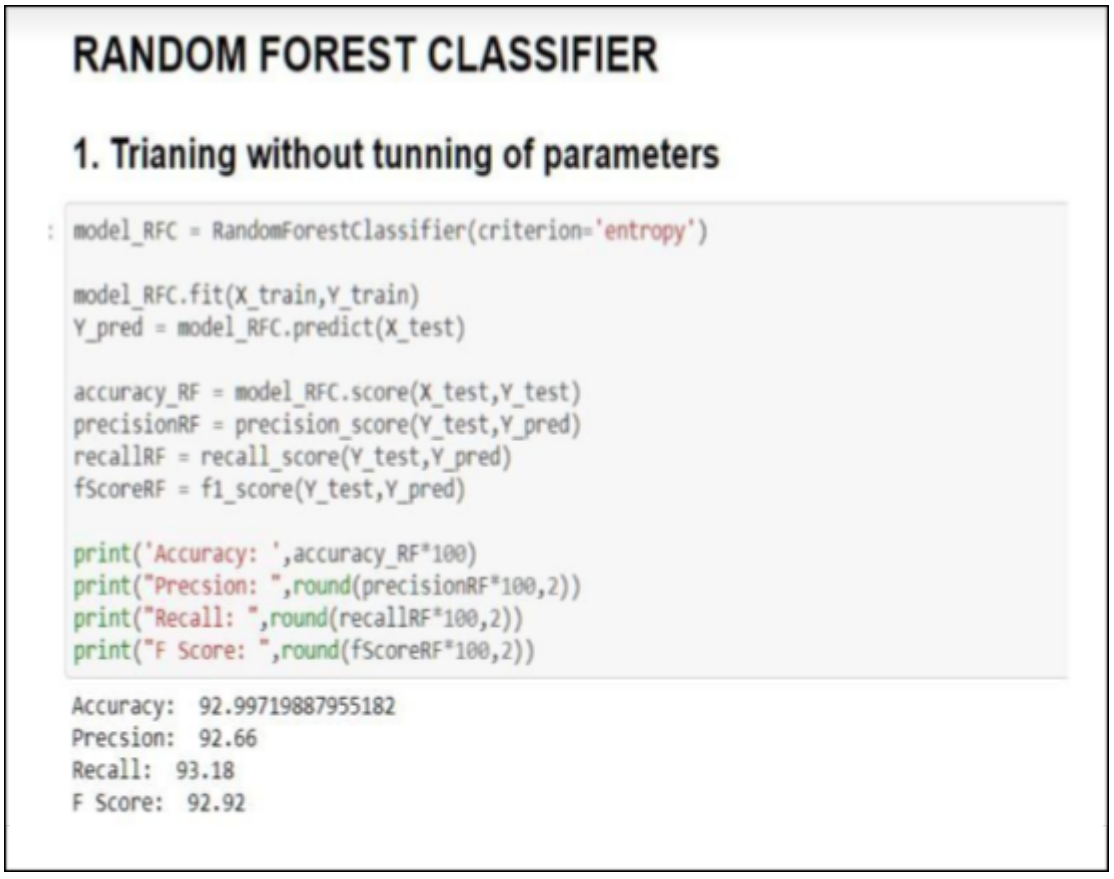


Figure 20. Training without tuning of parameters-RF

3. Training and Testing Model

```
model_RFC = RandomForestClassifier(n_estimators=100,max_features='sqrt',criterion='gini',n_jobs=-1)

model_RFC.fit(X_train,Y_train)
accuracy_RF = model_RFC.score(X_test,Y_test)

print('Accuracy: ',accuracy_RF*100)
Y_pred = model_RFC.predict(X_test)

conMatrix = confusion_matrix(Y_test,Y_pred)
precisionRF = precision_score(Y_test,Y_pred)
recallRF = recall_score(Y_test,Y_pred)
fScoreRF = f1_score(Y_test,Y_pred)

totalAccuracy.append(accuracy_RF)
totalPrecision.append(precisionRF)
totalRecall.append(recallRF)
totalFscore.append(fScoreRF)

print("Precsion: ",round(precisionRF*100,2))
print("Recall: ",round(recallRF*100,2))
print("F Score: ",round(fScoreRF*100,2))

print("Confusion Matrix: \n",conMatrix)
```

```
Accuracy:  93.83753501400056
Precsion:  93.75
Recall:  93.75
F Score:  93.75
Confusion Matrix:
 [[170  11]
 [ 11 165]]
```

Figure 21. Training and testing model

4. Confusion Matrix

```
plot_confusion_matrix(model_RFC, X_test, Y_test)
plt.title("Random Forest Confusion Matrix")
plt.show()
```

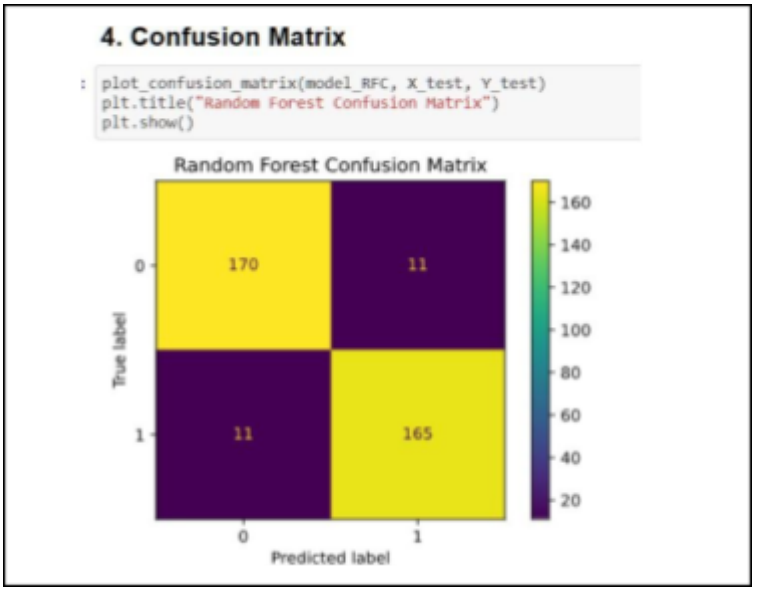


Figure 22. Confusion matrix

K-Nearest Neighbors

1. Trianing without tunning of parameters

```
model_KNN = KNeighborsClassifier()

model_KNN.fit(X_train,Y_train)
accuracy_KNN = model_KNN.score(X_test,Y_test)
print('Accuracy: ',accuracy_KNN*100,"\n")
Y_pred = model_KNN.predict(X_test)


precisionKNN = precision_score(Y_test,Y_pred)
recallKNN = recall_score(Y_test,Y_pred)
fScoreKNN = f1_score(Y_test,Y_pred)

print("Precsion: ",round(precisionKNN*100,2))
print("Recall: ",round(recallKNN*100,2))
print("F Score: ",round(fScoreKNN*100,2))
```

```
Accuracy:  85.43417366946778

Precsion:  82.63
Recall:  89.2
F Score:  85.79
```

Figure 23. Training without tuning -KNN

2. Hyperparameter tuning of K-Nearsest Neighbor

```
model = KNeighborsClassifier()

n_neighbors = range(1, 21, 2)
weights = ['uniform', 'distance']
metric = ['euclidean', 'manhattan', 'minkowski']
# define grid search
grid = dict(n_neighbors=n_neighbors,weights=weights,metric=metric)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv, scoring='accuracy',error_score=0)
grid_result = grid_search.fit(X_train,Y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Best: 0.914788 using {'metric': 'manhattan', 'n_neighbors': 11, 'weights': 'distance'}
```

Figure 24. Hypertuning with KNN

3. Training and Testing Model

```
model_KNN = KNeighborsClassifier(metric='manhattan',n_neighbors=11,weights='distance',n_jobs=-1)

model_KNN.fit(X_train,Y_train)
accuracy_KNN = model_KNN.score(X_test,Y_test)
print('Accuracy: ',accuracy_KNN*100,"\n")
Y_pred = model_KNN.predict(X_test)

confMatrix = confusion_matrix(Y_test,Y_pred)
precisionKNN = precision_score(Y_test,Y_pred)
recallKNN = recall_score(Y_test,Y_pred)
fScoreKNN = f1_score(Y_test,Y_pred)

totalAccuracy.append(accuracy_KNN)
totalPrecision.append(precisionKNN)
totalRecall.append(recallKNN)
totalFscore.append(fScoreKNN)

print("Precsion: ",round(precisionKNN*100,2))
print("Recall: ",round(recallKNN*100,2))
print("F Score: ",round(fScoreKNN*100,2))

print("Confusion Matrix: \n",confMatrix)
```

```
Accuracy:  92.43697478991596

Precsion:  91.16
Recall:  93.75
F Score:  92.44
Confusion Matrix:
 [[165  16]
 [ 11 165]]
```

Figure 25. Training and testing model-knn

4. Confusion Matrix

```
plot_confusion_matrix(model_KNN, X_test, Y_test)
plt.title("K-Nearest Neighbor Matrix")
plt.show()
```

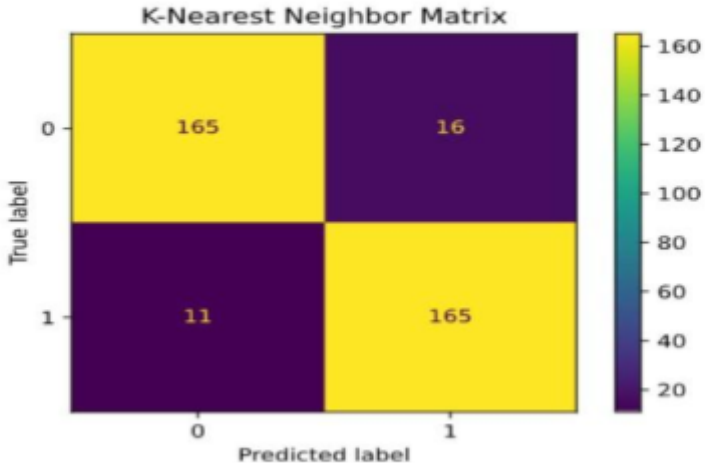


Figure 26. Confusion matrix

Naive Bayes

1. Training without tuning of parameters ¶

```
model_GNB = GaussianNB()
model_GNB.fit(X_train, Y_train)

Y_pred = model_GNB.predict(X_test)
accuracy_GNB = model_GNB.score(X_test,Y_test)
print('Accuracy: ',accuracy_GNB*100,"\n")

conMatrix = confusion_matrix(Y_test,Y_pred)
precisionGNB = precision_score(Y_test,Y_pred)
recallGNB = recall_score(Y_test,Y_pred)
fScoreGNB = f1_score(Y_test,Y_pred)


print("Precsion: ",round(precisionGNB*100,2))
print("Recall: ",round(recallGNB*100,2))
print("F Score: ",round(fScoreGNB*100,2))
```

```
Accuracy:  85.43417366946778

Precsion:  85.23
Recall:  85.23
F Score:  85.23
```

Figure 27. Training without tuning of parameters-NB

2. Hypertuning of Gaussian Naive Bayes

```
nb_classifier = GaussianNB()

params_NB = {'var_smoothing': np.logspace(0,-9, num=100)}
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=4, random_state=1)
gs_NB = GridSearchCV(estimator=nb_classifier,
                     param_grid=params_NB,
                     verbose=1,
                     cv=cv,
                     scoring='accuracy')


gs_NB.fit(X_train,Y_train)
print("Best: %f using %s" % (gs_NB.best_score_, gs_NB.best_params_))
```

```
Fitting 40 folds for each of 100 candidates, totalling 4000 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
Best: 0.831056 using {'var_smoothing': 0.2848035868435802}
[Parallel(n_jobs=1)]: Done 4000 out of 4000 | elapsed:   11.1s finished
```

Figure 28. Hypertuning with Naive bayes

3. Training and Testing Model

```
model_GNB = GaussianNB(var_smoothing=0.2848035868435802)
model_GNB.fit(X_train, Y_train)

Y_pred = model_GNB.predict(X_test)
accuracy_GNB = model_GNB.score(X_test,Y_test)
print('Accuracy: ',accuracy_GNB*100,"\n")

conMatrix = confusion_matrix(Y_test,Y_pred)
precisionGNB = precision_score(Y_test,Y_pred)
recallGNB = recall_score(Y_test,Y_pred)
fScoreGNB = f1_score(Y_test,Y_pred)

totalAccuracy.append(accuracy_GNB)
totalPrecision.append(precisionGNB)
totalRecall.append(recallGNB)
totalFscore.append(fScoreGNB)

print("Precsion: ",round(precisionGNB*100,2))
print("Recall: ",round(recallGNB*100,2))
print("F Score: ",round(fScoreGNB*100,2))

print("Confusion Matrix: \n",conMatrix)
```

```
Accuracy:  85.15406162464986

Precsion:  85.14
Recall:  84.66
F Score:  84.9
Confusion Matrix:
 [[155  26]
 [ 27 149]]
```

Figure 29. Training and testing model-NB

4. Confusion Matrix

```
plot_confusion_matrix(model_GNB, X_test, Y_test)
plt.title("Gausian Niave Bayes Confusion Matrix")
plt.show()
```

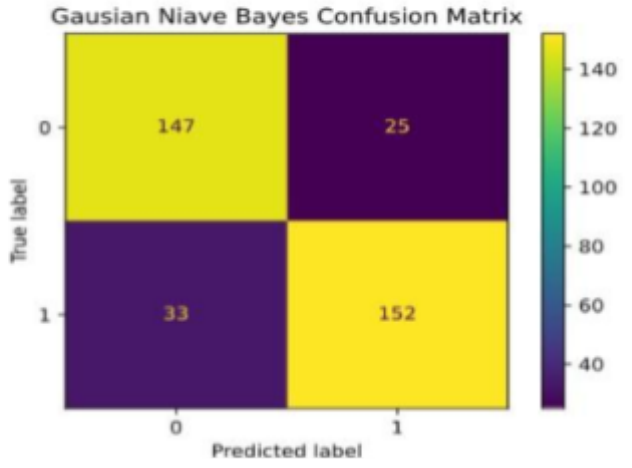


Figure 30. Confusion matrix-NB

Decision Tree

1. Training without tuning of parameters

```
model_DTC = DecisionTreeClassifier()
model_DTC.fit(X_train, Y_train)

Y_pred = model_DTC.predict(X_test)
accuracy_DTC = model_DTC.score(X_test,Y_test)
print('Accuracy: ',accuracy_DTC*100,"\n")

conMatrix = confusion_matrix(Y_test,Y_pred)
precisionDTC = precision_score(Y_test,Y_pred)
recallDTC = recall_score(Y_test,Y_pred)
fScoreDTC = f1_score(Y_test,Y_pred)

print("Precsion: ",round(precisionDTC*100,2))
print("Recall: ",round(recallDTC*100,2))
print("F Score: ",round(fScoreDTC*100,2))
```

```
Accuracy:  87.11484593837535

Precsion:  87.79
Recall:  85.8
F Score:  86.78
```

Figure 31. Training without tuning in DT

3. Training and Testing of Model

```
model_DTC = DecisionTreeClassifier(criterion='entropy',max_depth=8,min_samples_split=2)
model_DTC.fit(X_train, Y_train)

Y_pred = model_DTC.predict(X_test)
accuracy_DTC = model_DTC.score(X_test,Y_test)
print('Accuracy: ',accuracy_DTC*100,"\n")

conMatrix = confusion_matrix(Y_test,Y_pred)
precisionDTC = precision_score(Y_test,Y_pred)
recallDTC = recall_score(Y_test,Y_pred)
fScoreDTC = f1_score(Y_test,Y_pred)

print("Precsion: ",round(precisionDTC*100,2))
print("Recall: ",round(recallDTC*100,2))
print("F Score: ",round(fScoreDTC*100,2))

totalAccuracy.append(accuracy_DTC)
totalPrecision.append(precisionDTC)
totalRecall.append(recallDTC)
totalFscore.append(fScoreDTC)

print("Confusion Matrix: \n",conMatrix)
```

```
Accuracy:  89.35574229691878

Precsion:  86.32
Recall:  93.18
F Score:  89.62
Confusion Matrix:
 [[155  26]
 [ 12 164]]
```

Figure 32. Training and testing in DT

2. Hyperparameter tuning of Decision Tree

```
df_classifier = DecisionTreeClassifier()

params_DT = {'criterion': ['gini', 'entropy'],
             'max_depth': [1, 2, 3, 4, 5, 6, 7, 8],
             'min_samples_split': [2, 3,4,5,6]}

gs_DT = GridSearchCV(estimator=df_classifier,
                     param_grid=params_DT,
                     cv=10,
                     verbose=1,
                     scoring='accuracy')

gs_DT.fit(X_train,Y_train)
# df_classifier.score(X_test,Y_test)
print("Best: %f using %s" % (gs_DT.best_score_, gs_DT.best_params_))
```

```
Fitting 10 folds for each of 80 candidates, totalling 800 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
Best: 0.857157 using {'criterion': 'entropy', 'max_depth': 8, 'min_samples_split': 2}
[Parallel(n_jobs=1)]: Done 800 out of 800 | elapsed:    3.2s finished
```

Figure 33. Hypertuning in DT

4. Confusion Matrix

```
plot_confusion_matrix(model_DTC, X_test, Y_test)
plt.title("Decision Tree Confusion Matrix")
plt.show()
```

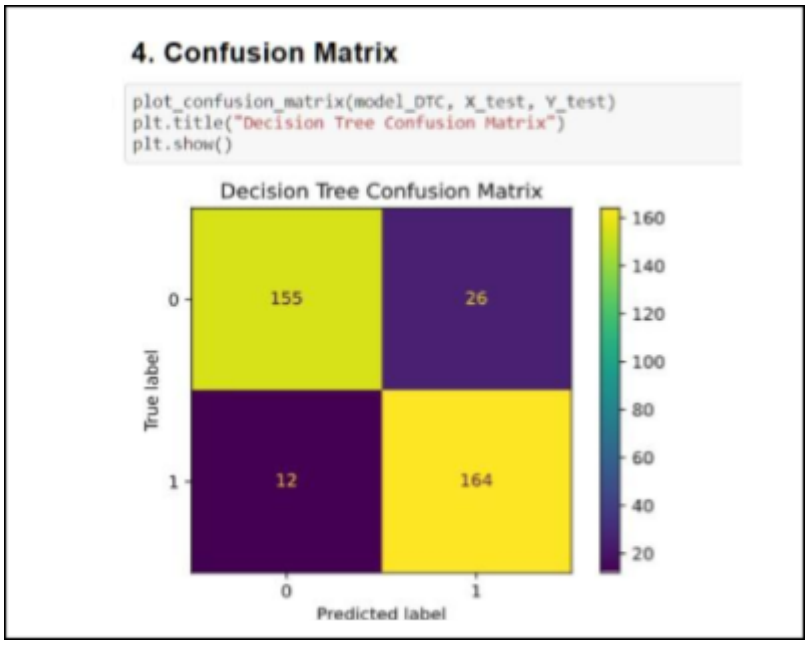


Figure 34.Confusion matrix-DT

Logistic Regression

1. Training wihtout tuning of parameters

```
model_LR = LogisticRegression()
model_LR.fit(X_train, Y_train)

Y_pred = model_LR.predict(X_test)
accuracy_LR = model_LR.score(X_test,Y_test)
print('Accuracy: ',accuracy_LR*100,"\n")

conMatrix = confusion_matrix(Y_test,Y_pred)
precisionLR = precision_score(Y_test,Y_pred)
recallLR = recall_score(Y_test,Y_pred)
fScoreLR = f1_score(Y_test,Y_pred)

print("Precsion: ",round(precisionLR*100,2))
print("Recall: ",round(recallLR*100,2))
print("F Score: ",round(fScoreLR*100,2))
```

```
Accuracy:  83.19327731092437

Precsion:  81.18
Recall:  85.8
F Score:  83.43
```

Figure 35.Training without tuning in logistic regression

2. Hyperparameter tuning of the model

```
model = LogisticRegression()
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
c_values = [100, 10, 1.0, 0.1, 0.01]

grid = dict(solver=solvers,penalty=penalty,C=c_values)
cv = RepeatedStratifiedKFold(n_splits=20, n_repeats=4, random_state=1)
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv, scoring='accuracy',error_score=0)
grid_result = grid_search.fit(X_train,Y_train)

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Best: 0.826604 using {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
```

penalty = ['l2'], c_values = [100, 10, 1.0, 0.1, 0.01] to get the best parameter.

Figure 36. Hyperparameter tuning of model

3. Training and Testing of the Model

```
model_LR = LogisticRegression(C=0.1,penalty='l2',solver='liblinear')
model_LR.fit(X_train, Y_train)

Y_pred = model_LR.predict(X_test)
accuracy_LR = model_LR.score(X_test,Y_test)
print('Accuracy: ',accuracy_LR*100,"\n")

conMatrix = confusion_matrix(Y_test,Y_pred)
precisionLR = precision_score(Y_test,Y_pred)
recallLR = recall_score(Y_test,Y_pred)
fScoreLR = f1_score(Y_test,Y_pred)

print("Precsion: ",round(precisionLR*100,2))
print("Recall: ",round(recallLR*100,2))
print("F Score: ",round(fScoreLR*100,2))


totalAccuracy.append(accuracy_LR)
totalPrecision.append(precisionLR)
totalRecall.append(recallLR)
totalFscore.append(fScoreLR)

print("Confusion Matrix: \n",conMatrix)
```

```
Accuracy:  82.6330532212885

Precsion:  80.98
Recall:  84.66
F Score:  82.78
Confusion Matrix:
 [[146  35]
 [ 27 149]]
```

Figure 37. Training and testing model in logistic regression

4. Confusion Matrix

```
plot_confusion_matrix(model_LR, X_test, Y_test)
plt.title("Logistic Regression Confusion Matrix")
plt.show()
```

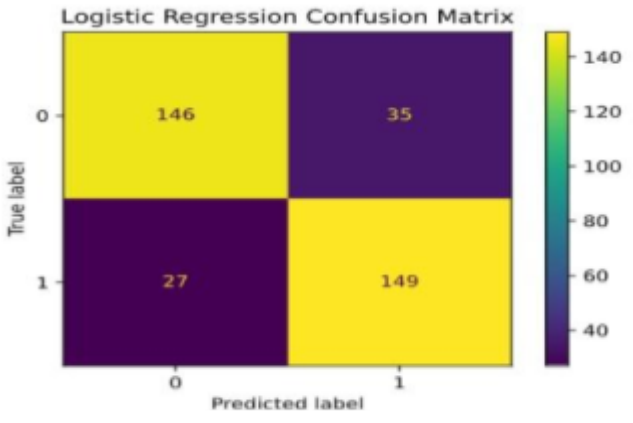
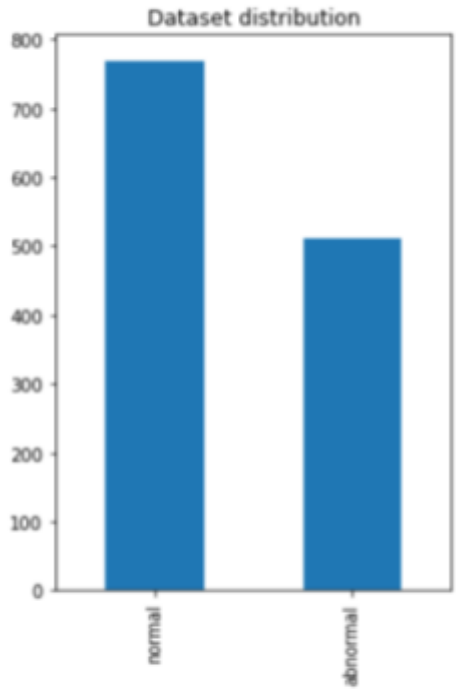


Figure 38. Confusion matrix-LR

```
plt.figure(figsize=(4,6))
dataset.label.value_counts().plot(kind='bar', title="Dataset distribution")
plt.show()
```




```
train, test = train_test_split(dataset, test_size=0.2, random_state=42)

print("Train: %i" % len(train))
print("Test: %i" % len(test))
```

```
Train: 1024
Test: 257
```

Figure 39. Splitting

From this we can see that we split the dataset into 80-20 ratio where in training set we have 1024 samples of audio files while in testing dataset we have 257 samples. We can also test this by changing the ratio to 70-30,80-20.

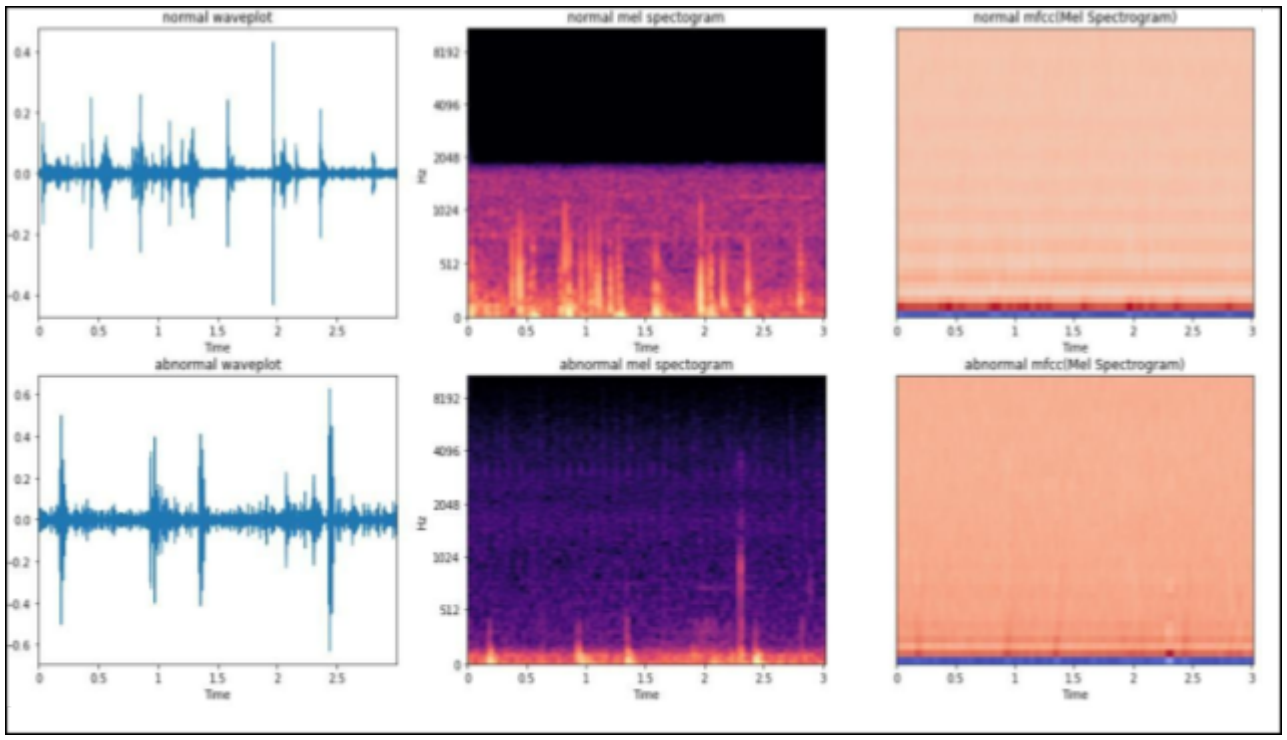


Figure 40. Spectrogram generation

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 39, 129, 16)       80

 max_pooling2d_8 (MaxPooling  (None, 19, 64, 16)       0
 2D)

 dropout_8 (Dropout)         (None, 19, 64, 16)        0

 conv2d_9 (Conv2D)           (None, 18, 63, 32)        2080

 max_pooling2d_9 (MaxPooling  (None, 9, 31, 32)        0
 2D)

 dropout_9 (Dropout)         (None, 9, 31, 32)         0

 conv2d_10 (Conv2D)          (None, 8, 30, 64)         8256

 max_pooling2d_10 (MaxPoolin  (None, 4, 15, 64)        0
 g2D)

 dropout_10 (Dropout)        (None, 4, 15, 64)         0

 conv2d_11 (Conv2D)          (None, 3, 14, 128)        32896

 max_pooling2d_11 (MaxPoolin  (None, 1, 7, 128)        0
 g2D)

 dropout_11 (Dropout)        (None, 1, 7, 128)         0

 global_average_pooling2d_2   (None, 128)              0
 (GlobalAveragePooling2D)

 dense_2 (Dense)             (None, 2)                 258

=================================================================
Total params: 43,570
Trainable params: 43,570
Non-trainable params: 0
```

Figure 41.CNN layers

```
Epoch 285/300
8/8 [==============================] - 4s 458ms/step - loss: 0.2111 - accuracy: 0.9082 - val_loss: 0.3311 - val_accuracy: 0.8482
Epoch 286/300
8/8 [==============================] - 4s 463ms/step - loss: 0.1989 - accuracy: 0.9170 - val_loss: 0.3352 - val_accuracy: 0.8521
Epoch 287/300
8/8 [==============================] - 4s 459ms/step - loss: 0.2110 - accuracy: 0.9062 - val_loss: 0.3440 - val_accuracy: 0.8366
Epoch 288/300
8/8 [==============================] - 4s 456ms/step - loss: 0.2283 - accuracy: 0.8994 - val_loss: 0.3792 - val_accuracy: 0.8171
Epoch 289/300
8/8 [==============================] - 4s 464ms/step - loss: 0.2113 - accuracy: 0.9131 - val_loss: 0.3903 - val_accuracy: 0.8054
Epoch 290/300
8/8 [==============================] - 4s 456ms/step - loss: 0.2218 - accuracy: 0.8936 - val_loss: 0.3755 - val_accuracy: 0.8132
Epoch 291/300
8/8 [==============================] - 6s 736ms/step - loss: 0.1946 - accuracy: 0.9150 - val_loss: 0.3474 - val_accuracy: 0.8405
Epoch 292/300
8/8 [==============================] - 4s 462ms/step - loss: 0.1953 - accuracy: 0.9238 - val_loss: 0.3398 - val_accuracy: 0.8288
Epoch 293/300
8/8 [==============================] - 4s 457ms/step - loss: 0.1988 - accuracy: 0.9189 - val_loss: 0.3115 - val_accuracy: 0.8521
Epoch 294/300
8/8 [==============================] - 4s 478ms/step - loss: 0.1954 - accuracy: 0.9287 - val_loss: 0.3577 - val_accuracy: 0.8444
Epoch 295/300
8/8 [==============================] - 4s 460ms/step - loss: 0.2039 - accuracy: 0.9209 - val_loss: 0.3127 - val_accuracy: 0.8482
Epoch 296/300
8/8 [==============================] - 4s 463ms/step - loss: 0.1737 - accuracy: 0.9326 - val_loss: 0.3664 - val_accuracy: 0.8093
Epoch 297/300
8/8 [==============================] - 4s 461ms/step - loss: 0.2190 - accuracy: 0.9033 - val_loss: 0.2983 - val_accuracy: 0.8638
Epoch 298/300
8/8 [==============================] - 4s 456ms/step - loss: 0.1926 - accuracy: 0.9053 - val_loss: 0.3338 - val_accuracy: 0.8482
Epoch 299/300
8/8 [==============================] - 4s 456ms/step - loss: 0.1835 - accuracy: 0.9258 - val_loss: 0.3145 - val_accuracy: 0.8560
Epoch 300/300
8/8 [==============================] - 4s 460ms/step - loss: 0.1856 - accuracy: 0.9229 - val_loss: 0.3439 - val_accuracy: 0.8405
```

Figure 42. Epoch stage

Chapter-5

CONCLUSIONS

Ist approach

The aim of this project was to achieve the most significant results in determining whether a patient had heart disease or not. Various machine learning algorithms, such as logistic regression, naive bayes, decision trees, KNN, and Random Forest, were used to make these predictions. An organised dataset from the UCI repository was utilised for this. Python 3.8 and its required libraries were used for modelling. All team members ran the code on their various systems, although the major findings were produced on an Intel(R) Corel(TM) i5-9Gl with a CPU clock speed of 1.00GHz 1.19GHz and 8GB of RAM.Visual and Tabular representation of the results are mentioned below.

Table3:-Tabular representation of results

AI	Random Forest	KNN	Naïve Bayes	Decision Tree	Logistic Regression
Accuracy	93.84%	92.44%	85.15%	89.36%	82.63%
Precision	93.75%	91.16%	85.14%	86.32%	80.98%
Recall	93.75%	93.75%	84.66%	93.18%	84.66%
F Score	93.75%	92.44%	84.90%	89.62%	82.78%

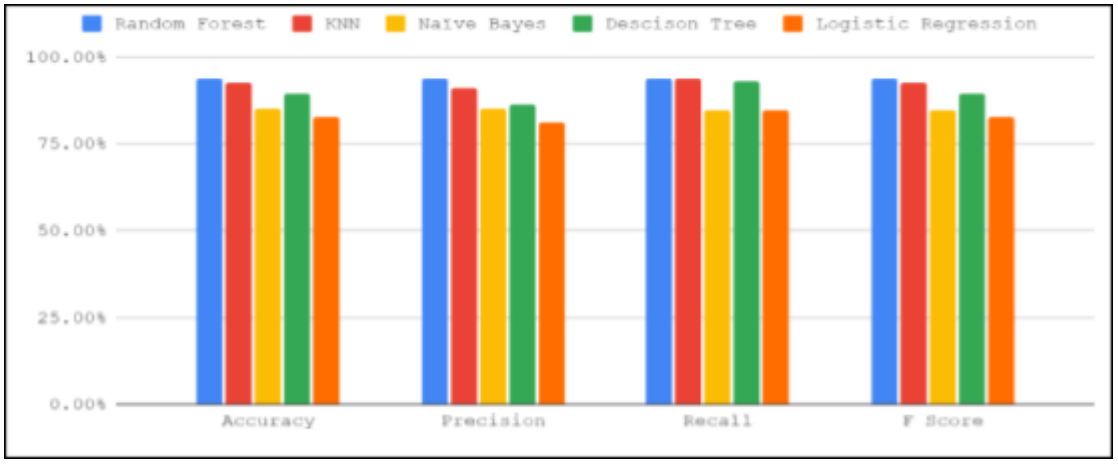


Figure 43. Histograph

Splitting Cases

1)90-10

Table 4:90-10

Parameters	RF	KNN	NB	DT
Accuracy	96.63%	96.63%	89.07%	88.23%
Precision	98.57%	98.57%	92.75%	96.77%
Recall	95.83%	95.83%	88.89%	83.33%
F1Score	97.18%	97.18%	90.78%	89.55%

2)70-30

Table 5: 70-30

Parameters	RF	KNN	NB	DT
Accuracy	93.55%	92.71%	85.71%	87.11%
Precision	94.12%	92.75%	87.62%	87.2%
Recall	94.58%	94.58%	87.19%	90.64%
F1Score	94.35%	93.66%	87.41%	88.89%

3)80-20

Table 6: 80-20

Parameters	RF	KNN	NB	DT
Accuracy	94.95%	94.11%	85.29%	87.81%
Precision	94.74%	93.33%	86.36%	88.64%
Recall	96.18%	96.18%	87.02%	89.31%
F1Score	95.45%	96.74%	86.69%	88.97%

We computed the results for the machine learning algorithms for three sets of training -testing data, where we split our data into 70-30, 90-10, 80-20 .We

observed that value for precision,recall,accuracy,f1score is best in case of 90-10.But if we talk about the algorithms we found that Random Forest is performing better than rest of the algorithms. As we know that Random Forest avoids overfitting and takes less computational time.

IInd approach

This study was meant to detect an aberrant heartbeat sound utilising stethoscope sounds and heartbeats collected using a mobile phone's microphone. Convolutional Neural Networks were used to classify heartbeat sounds. Since the temporal behavior of the heartbeat repeated itself within the observation window, we did not use any other time sequence-based neural networks, such as RNNs, because there was no need to learn any sequential patterns. One of the first studies to use deep convolutional neural networks for the automatic classification of heart sounds from stethoscope-captured heartbeat sounds is the one shown here.Through the development of a new algorithm, we were able to create a two-dimensional time-frequency Mel spectrogram from a one-dimensional time series. The MFCC of the Mel Spectrogram is then trained using a 4-layer CNN architecture. The trained network recognises normal and pathological heartbeat sound inputs automatically. The following epoch values were used: 100, 150, 200, 250, and 300. 300 epoch with a learning rate of 0.001 produced the best results on a batch size of 128. The training accuracy is 89,45 percent, whereas the testing accuracy is 82.10 percent.

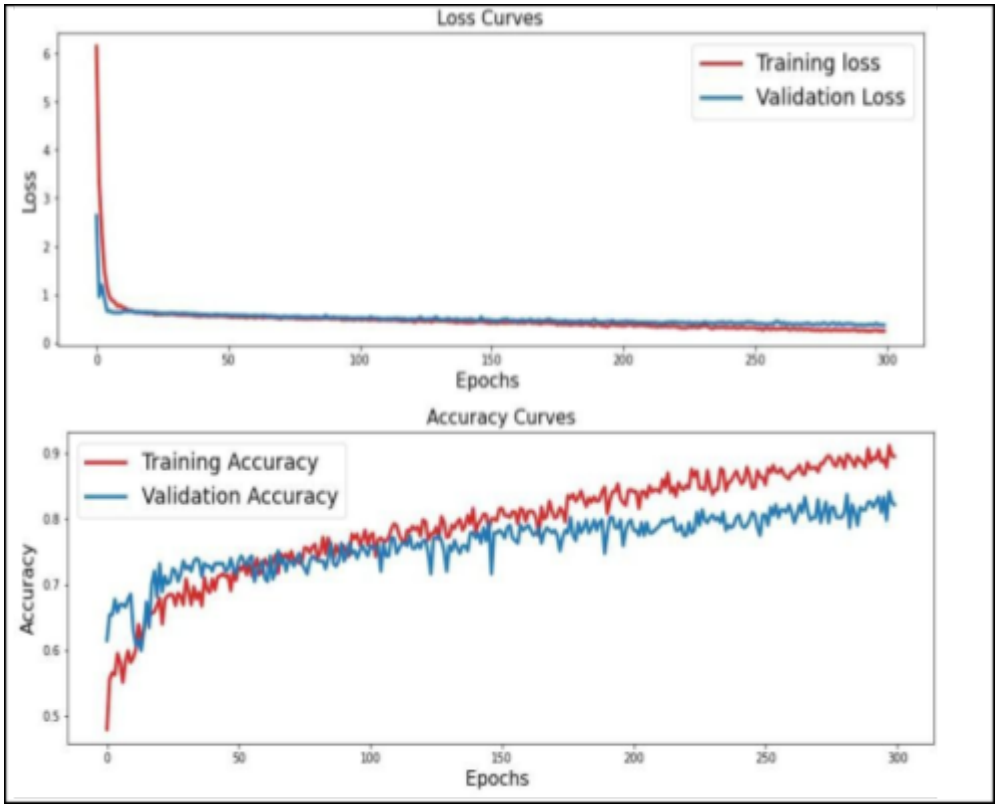


Figure 44. Model Evaluation

Classification report of Heartbeat classifier

	precision	recall	f1-score	support
abnormal	0.74	0.79	0.77	95
normal	0.87	0.84	0.86	162
accuracy			0.82	257
macro avg	0.81	0.81	0.81	257
weighted avg	0.82	0.82	0.82	257

Figure 45. Classification report

Model Test on my heartbeat.wav file

```
# load and evaluate a saved model
from keras.models import load_model

# load model
model = load_model("heartbeat_classifier.h5")

# File to be classified
classify_file = "/content/drive/MyDrive/heartbeat-dataset/my heartbeat/my_heartbeat.wav"
x_test = []
x_test.append(extract_features(classify_file,0.5))
x_test = np.asarray(x_test)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)
pred = model.predict(x_test,verbose=1)

print(pred)
```

```
1/1 [==============================] - 0s 100ms/step
[[4.5213135e-04 9.9954790e-01]]
```

```
pred_class = np.argmax(model.predict(x_test),axis=-1)
if pred_class[0]:
    print("Normal heartbeat")
    print("confidence:",pred[0][1])
else:
    print("Abnormal heartbeat")
    print("confidence:",pred[0][0])
```

```
1/1 [==============================] - 0s 26ms/step
Normal heartbeat
confidence: 0.9995479
```

Figure 46. Model test

References

A) Datasets

https://archive.ics.uci.edu/ml/datasets/heart+disease
http://www.peterjbentley.com/heartchallenge/index.html

B) Journals/periodicals

[1]Cleveland_clinic:https://my.clevelandclinic.org/health/body/21704-heart [Access Date: 08-11-22]

[2]mayo_clinic: https://www.mayoclinic.org/diseases-conditions/heart-failure/symptoms-causes/s
[Access Date: 09-11-22]

[3]WHO: https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1 [Access Date: 09-11-22]

[4]NIH:https://www.ncbi.nlm.nih.gov/books/NBK541010/#:~:text=Heart%20sounds%20are%20%20%20created%20from,more%20vibrations%20that%20get%20created. [Access Date: 14-11-22]

[5]PNSN:https://pnsn.org/spectrograms/what-is-a-spectrogram#:~:text=A%20spectrogram%20is%20%20%20a%20visual,energy%20levels%20vary%20over%20time.
[Access Date: 15-11-22]

[6]D. Shah, S. Patel, and S. K. Bharti, ‘‘Heart disease prediction using machine learning techniques,’’ Social Netw. Computer. Sci., vol. 1, no. 6, pp. 1–6, Nov. 2020

[7]V. Sharma, S. Yadav, and M. Gupta, “Heart disease prediction using machine learning techniques,” in Proceedings of the 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), pp. 177–181, December 2020.

[8]Demir F, Şengür A, Polat VBK “Towards the classification of heart sounds based on Convolutional deep neural network”, Health Inf Sci Syst 7, 16 2019.

[9]M.S. Wibawa, I.M.D. Maysanjaya, N.K.D.P. Novianti, and P.N. Crisnapati, “Abnormal Heart Rhythm Detection Based on Spectrogram of Heart Sound using Convolutional Neural Network”, 2018 6th International Conference on Cyber and IT Service Management (CITSM), 2019, pp. 1–4.

Riya_Report_Second

ORIGINALITY REPORT

16%	13%	10%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.hindawi.com Internet Source	1%
2	arxiv.org Internet Source	1%
3	Vijeta Sharma, Shrinkhala Yadav, Manjari Gupta. "Heart Disease Prediction using Machine Learning Techniques", 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2020 Publication	1%
4	www.researchgate.net Internet Source	1%
5	iopscience.iop.org Internet Source	1%
6	Hiteshwar Singh, Tushar Gupta, Jagpreet Sidhu. "Prediction of Heart Disease using Machine Learning Techniques", 2021 Sixth International Conference on Image Information Processing (ICIIP), 2021 Publication	<1%

7	hdl.handle.net Internet Source	<1 %
8	research-repository.griffith.edu.au Internet Source	<1 %
9	escholarship.org Internet Source	<1 %
10	www.nature.com Internet Source	<1 %
11	rstudio-pubs-static.s3.amazonaws.com Internet Source	<1 %
12	quieora.ink Internet Source	<1 %
13	stackoverflow.com Internet Source	<1 %
14	N. S. A. Azhar, N. M. Z. Hashim, A. I. Kamaruddin, M. Kamal, M. D. Sulistiyo. "A deep learning-based smart application for malay vowel recognition toward rehabilitation treatment", Engineering Technology International Conference (ETIC 2022), 2022 Publication	<1 %
15	thesai.org Internet Source	<1 %
16	file.techscience.com Internet Source	<1 %