

HEART DISEASE DIAGNOSIS AND PREDICTION USING WAVELET FEATURES AND DEEP CNN

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Technology degree in Information Technology

By

**JESTIN JOSE (Reg.No - 39120047)
JOSHVA J (Reg.No - 39120048)**



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119**

NOVEMBER - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with —All grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **JESTINJOSE(39120047)** and **JOSHVA J(39120048)** who carried out the Project Phase-II entitled “**HEART DISEASE DIAGNOSIS AND PREDICTION USING WAVELET FEATURE AND DEEP CNN**” under my supervision from January 2023 to May 2023.

Dr. G. Mathivanan
26/4/2023
Internal Guide

Dr. G. MATHIVANAN, M.E., Ph.D.,

Dr. R. Subhashini
20/5/23

Head of the Department

Dr. R.SUBHASHINI, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Dr. G. Mathivanan

Internal Examiner

Dr. R. Subhashini

External Examiner

DECLARATION

I, **JESTIN JOSE(Reg.No- 39120047)** and **JOSHVA J(Reg.No-39120048)** hereby declare that the Project Phase-II Report entitled “**HEART DISEASE DIAGNOSIS AND PREDICTION USING WAVELET FEATURE AND DEEP CNN**” done by us under the guidance of **Dr. G. Mathivanan, M.E.,Ph.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in **INFORMATION TECHNOLOGY**.

DATE:

Place :Chennai

A handwritten signature in black ink, appearing to be 'Jestin Jose', with a long horizontal stroke extending to the right.

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. R.SUBHASHINI M.E., Ph.D.**, Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.G.MATHIVANAN M.E.,Ph.D**,for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-II project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project

ABSTRACT

One of the most prominent tools for detecting cardiovascular problems is the electrocardiogram (ECG). The electrocardiogram (ECG or EKG) is a diagnostic tool that is used to routinely assess the electrical and muscular functions of the heart. Even though it is a comparatively simple test to perform, the interpretation of the ECG charts requires considerable amounts of training. Till recently, the majority of ECG records were kept on paper. Thus, manually examining and reexamining the ECG paper records often can be a time-consuming and daunting process. If we digitize such paper ECG records, we can perform automated diagnosis and analysis. The main goal of this project is to use machine learning to convert ECG paper records into a 1-D signal. This can be achieved by extracting the P, QRS, and T waves that exist in ECG signals to demonstrate the electrical activity of the heart using various techniques. The techniques include splitting the original ECG report into 13 Leads, extracting and converting into the signal, smoothing, converting them to binary images using threshold and scaling. Post-feature extraction, dimension reduction techniques like Principal Component Analysis are applied to understand the data. Multiple classifiers like k-nearest neighbors (KNN), Logistic Regression, Support Vector Machine (SVM), and Voting Based Ensemble Classifier are implemented, and based on the acceptable criteria on the accuracy, precision, recall, f1-score, and support, the model will be finalized. This final model will aid in the diagnosing of cardiac diseases, to detect whether a patient has/had Myocardial Infarction, Abnormal Heartbeat, or the patient is hale and healthy by inferring the ECG reports.

Keywords: Wavelet features, deep convolutional neural networks (deep CNNs), Electrocardiogram (ECG).

Chapter No	TITLE		Page No.
	ABSTRACT		5
	LIST OF FIGURES		7
1	INTRODUCTION		11
	1.1	Motivation For The Work	12
	1.2	Problem Statement	13
2	LITERATURE SURVEY		14
3	METHODOLOGY		16
	3.1	Existing System	16
	3.2	Proposed System	16
		3.21 Collection of Datasets	17
		3.22 Selection of Attributes	18
		3.23 Preprocessing of Data	19
		3.24 ECG Image Processing	19
		3.25 Prediction of Disease	21
4	WORKING OF SYSTEM		22
	4.1	System Architecture	22
	4.2	Machine Learning	23
	4.3	Algorithms	25
		4.31 Support Vector Machine	25
		4.32 Naive's Bayes	29
		4.33 Decision Tree	31

		4.34 Random Forest	35
		4.35 Logistic Regression	38
		4.36 Adaboost	41
		4.37 XgBoost	42
	4.4	Performance Analysis	45
5	RESULTS AND DISCUSSION		
6	CONCLUSION & FUTUREWORK		47
	REFERENCES		48
	APPENDIX A. Screenshots B. Source code		50

LIST OF FIGURES

S.NO	FIGURE DESCRIPTION	PAGE NO
3.1	Collection of Dataset	16
3.2	Correlation Matrix	17
3.3	Preprocessing Of Data	18
3.4	Data Balancing	19
3.5	Prediction Of Disease	20
4.1	System Architecture	21
4.2	Logistic Regression	39
4.3	Xg-boost	43
4.4	Confusion Matrix	44
4.5	Correlation Matrix	45
A.1	Pairplots	51
A.2	Colation HeatMap	51
A.3	Accuracy of Algorithms	52

LIST OF SYMBOLS

TP	Number of people with heart diseases.
TN	Number of people with heart diseases and no heart diseases.
FP	Number of people with no heart diseases.
FN	Number of people with no heart diseases and with heart diseases.
$f(x)$	Output between the 0 and 1 value.
e	Base of the natural logarithm.
x	Input to the function.

LIST OF ABBREVIATIONS

CNN	Convolution Neural Network
AI	Artificial Intelligence
NN	Neural Networks
SVM	Support Vector Machine
XG	Extreme Gradient
ECG	Electro Cardiogram

CHAPTER 1

INTRODUCTION

According to the World Health Organization, every year 12 million deaths occur worldwide due to Heart Disease. Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of data analysis. The load of cardiovascular disease is rapidly increasing all over the world from the past few years. Many researches have been conducted in attempt to pinpoint the most influential factors of heart disease as well as accurately predict the overall risk. Heart Disease is even highlighted as a silent killer which leads to the death of the person without obvious symptoms. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes in high-risk patients and in turn reduces the complications.

Machine learning proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the health care industry. This project aims to predict future Heart Disease by analyzing data of patients which classifies whether they have heart disease or not using machine-learning algorithm. Machine Learning techniques can be a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analyzing to extract the desired data we can say that this technique can be very well adapted to do the prediction of heart disease.

1.1 MOTIVATION FOR THE WORK

Heart disease is one of the leading causes of death worldwide, and early detection of the disease is critical for effective treatment and prevention. Predictive modeling based on medical imaging is an emerging area that holds great promise for accurate diagnosis and disease prognosis. Wavelet transform has proven to be an effective signal processing technique for analyzing and extracting features from biomedical signals. Combined with a deep convolutional neural network (CNN), wavelet features can be used to construct a powerful predictive model for heart disease diagnosis. The proposed model aims to leverage the advantages of wavelet analysis and deep CNNs to improve the accuracy and efficiency of heart disease prediction. The wavelet transform will be used to extract relevant features from electrocardiogram (ECG) signals, which will then be fed into a deep CNN for classification. The model will be trained on a large dataset of ECG recordings, and its performance will be evaluated using various metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic (ROC) curve. The outcome of this work has significant implications for improving the diagnosis and management of heart disease. The proposed model has the potential to identify patients at high risk of developing heart disease, allowing for early intervention and treatment. Additionally, the model can be used to monitor disease progression and treatment response, leading to improved patient outcomes and quality of life.

1.2 PROBLEM STATEMENT

Heart disease is a serious problem that affects many people around the world. Current diagnostic methods are not always accurate, particularly for patients with no symptoms. Deep learning algorithms have shown promise for heart disease prediction using ECG signals, but they can be improved by incorporating wavelet-based features. However, the potential of this approach has not been extensively studied.

The goal of this research is to develop a better predictive model for heart disease using wavelet features and deep learning. The research will answer questions such as how wavelet-based features improve accuracy and how the new model compares to existing methods. The results can help improve early detection and prevention of heart disease, and guide the development of better diagnostic tools.

CHAPTER 2

LITERATURE SURVEY

With growing development in the field of medical science alongside machine learning various experiments and researches has been carried out in these recent years releasing the relevant significant papers.

Purushottam proposed a paper “Efficient Heart Disease Prediction System” using hill climbing and decision tree algorithms .They used Cleveland Dataset and pre-processing of data is performed before using classification algorithms. The Knowledge Extraction is done based on Evolutionary Learning (KEEL), an open-source data mining tool that fills the missing values in the data set. A decision tree follows top-down order. For each actual node selected by hill-climbing algorithm a node is selected by a test at each level. The parameters and their values used are confidence. Its minimum confidence value is 0.25. The accuracy of the system is about 86.7%.

Santhana Krishnan. J ,et ,al proposed a paper “Prediction of Heart Disease Using Machine Learning Algorithms” using decision tree and Naive Bayes algorithm for prediction of heart disease. In decision tree algorithm the tree is built using certain conditions which gives True or False decisions. The algorithms like SVM, KNN are results based on vertical or horizontal split conditions depends on dependent variables. But decision tree for a tree like structure having root node, leaves and branches base on the decision made in each of tree Decision tree also help in the understating the importance of the attributes in the data set . They have also used Cleveland data set. Dataset splits in 70% training and 30% testing by using some methods. This algorithm gives 91% accuracy. The second algorithm is Naive Bayes, which is used for classification. It can handle complicated, nonlinear, dependent data so it is found

suitable for heart disease dataset as this dataset is also complicated, dependent and nonlinear in nature. This algorithm gives an 87% accuracy.

Sonam Nikhar et al proposed paper “ Prediction of Heart Disease Using Machine Learning Algorithms” their research gives point to point explanation of Naïve Bayes and decision tree classifier that are used especially in the prediction of Heart Disease Some analysis has been led to think about the execution of prescient data mining strategy on the same dataset, and the result decided that Decision Tree

Avinash Golande et al, proposed “Heart Disease Prediction Using Effective Machine Learning Techniques” in which few data mining techniques are used that support the doctors to differentiate the heart disease. Usually utilized methodologies are k-nearest neighbour, Decision tree and Naive Bayes. Other unique characterization-based strategies utilized are packing calculation, Part thickness, consecutive negligible streamlining and neural systems, straight Kernel self arranging guide and SVM (Bolster Vector Machine).

Lakshmana Rao et al, proposed “Machine Learning Techniques for Heart Disease Prediction” in which the contributing elements for heart disease are more. So, it is difficult to distinguish heart disease. To find the seriousness of the heart disease among people different neural systems and data mining techniques are used

CHAPTER 3

METHODOLOGY

3.1 EXISTING SYSTEM

Heart disease is even being highlighted as a silent killer which leads to the death of a person without obvious symptoms. The nature of the disease is the cause of growing anxiety about the disease & its consequences. Hence continued efforts are being done to predict the possibility of this deadly disease in prior. So that various tools & techniques are regularly being experimented with to suit the present-day health needs. Machine Learning techniques can be a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analyzing to extract the desired data we can conclude. This technique can be very well adapted to the do the prediction of heart disease. As the well-known quote says “Prevention is better than cure”, early prediction & its control can be helpful to prevent & decrease the death rates due to heart disease.

3.2 PROPOSED SYSTEM

The working of the system starts with the collection of data and selecting the important attributes. Then the required data is Preprocessing into the required format. The data is then divided into two parts training and testing data. The algorithms are applied and the model is trained using the training data. The accuracy of the system is obtained by testing the system using the testing data. This system is implemented using the following modules.

3.21 Collection of Dataset

Initially, we collect a data set for our heart disease prediction system. After the collection of the datasets, we split the datasets into training data and testing data. The training datasets is used for prediction model learning and testing data is used for evaluating the prediction model. For this project, 70% of training data is used and 30% of data is used for testing. The datasets used for this project is Heart Disease UCI. The datasets consists of 76 attributes; out of which, 14 attributes are used for the system.

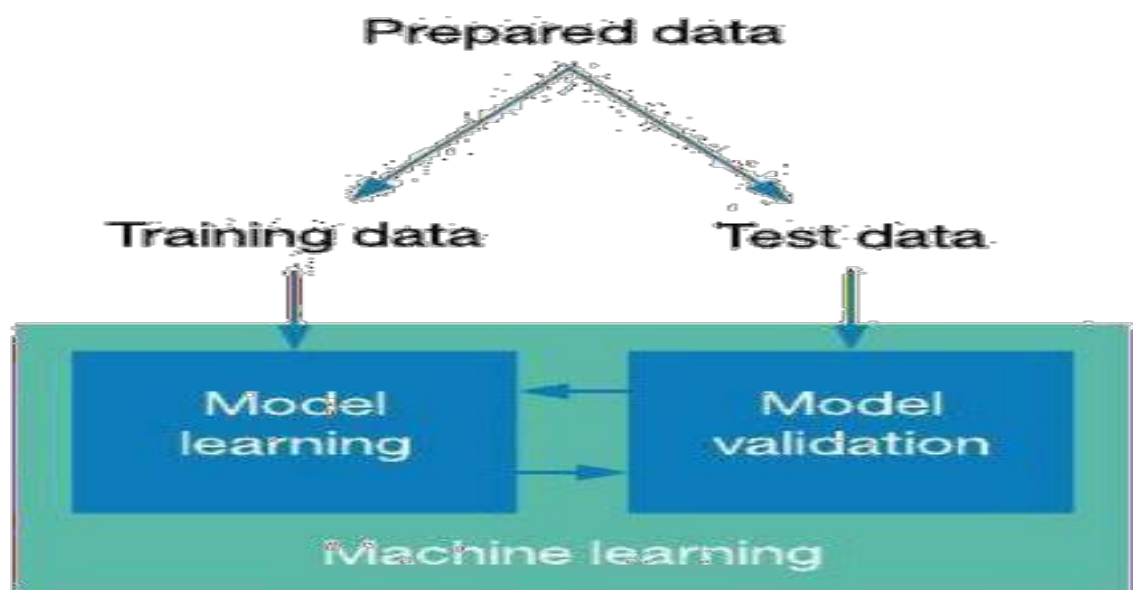


Fig 3.1: Collection of Data

3.22 Selection of attributes

Attribute or Feature selection includes the selection of appropriate attributes for the prediction system. This is used to increase the efficiency of the system. Various attributes of the patient like gender, chest pain type, fasting blood pressure, serumcholesterol, exang, etc are selected for the prediction. The Correlation matrix is used for attribute selection for this model.



Fig 3.2: Correlation matrix

3.23 Pre-processing of Data

Data pre-processing is an important step for the creation of a machine learning model. Initially, data may not be clean or in the required format for the model which can cause misleading outcomes. In pre-processing of data, we transform data into our required format. It is used to deal with noises, duplicates, and missing values of the datasets. Data pre-processing has the activities like importing datasets, splitting datasets, attribute scaling, etc. Preprocessing of data is required for improving the accuracy of the model.



Fig 3.3: Data Pre-processing

3.24 ECG Image processing

ECG image processing involves the extraction of relevant information from electrocardiogram (ECG) signals using digital image processing techniques. This process includes filtering, noise reduction, feature extraction, and classification of ECG signals. ECG image processing plays an important role in diagnosing and monitoring heart diseases, as it allows clinicians to extract critical information from ECG signals in a fast and efficient manner. With advances in technology, ECG image processing is becoming increasingly important in the development of intelligent ECG monitoring systems, which can improve the accuracy and effectiveness of

ECG signal analysis.

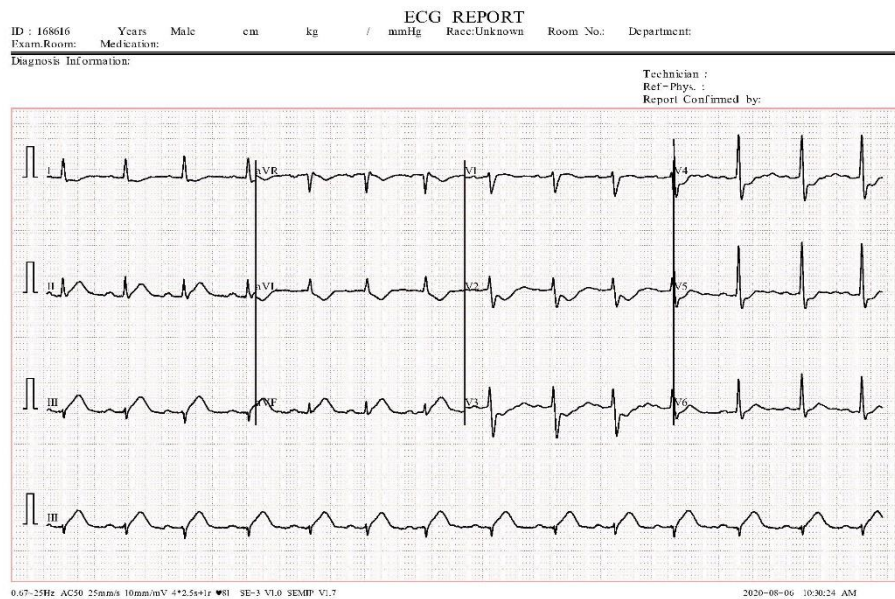


Fig 3.4: ECG Image processing

Data Cleaning & Feature Transformation: To prepare Leads (1-13) for further processing, each individual lead image is transformed by removing Gridlines, converting to Grayscale, applying Gaussian filtering, and performing Thresholding to convert to binary image.

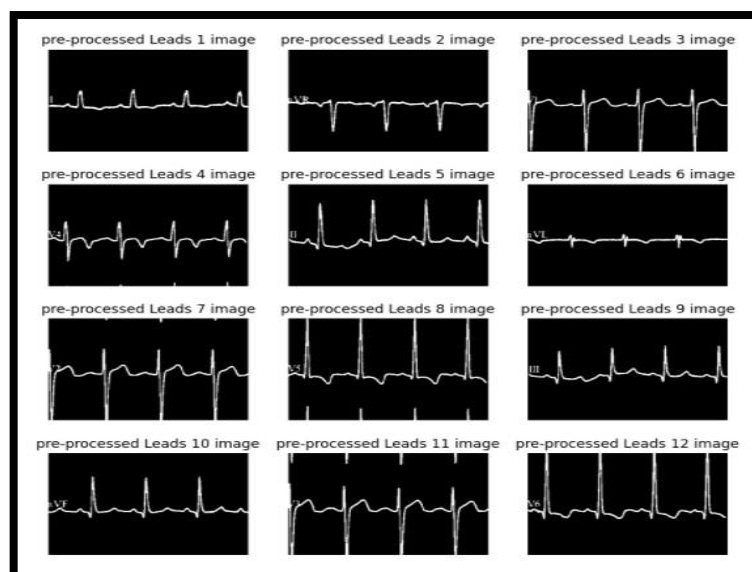


Fig 3.4.1 Data Cleaning and Feature transformation

The transformed image is **traced** to extract only the signals from the image using the **contour technique**, and the values are scaled using the Minmax Scalar. The normalized output is saved in CSV format as a 2D signal.

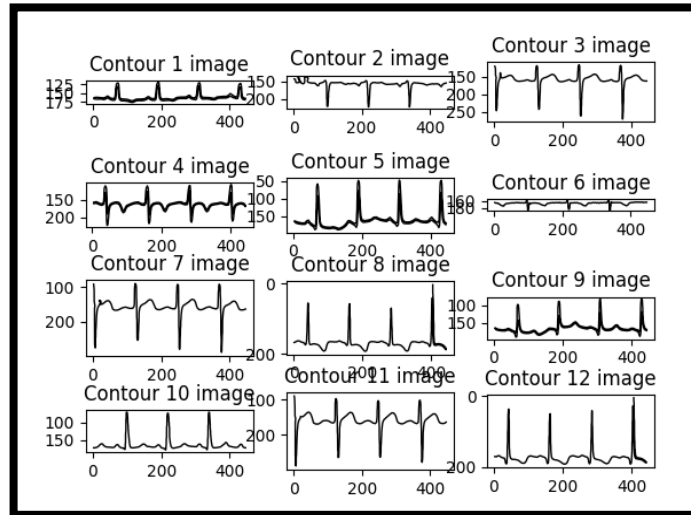


Fig 3.4.2 Contouring Image

3.25 Prediction of Disease

Various machine learning algorithms like SVM, Naive Bayes, Decision Tree, Random Tree, Logistic Regression, Ada-boost, Xg-boost are used for classification. Comparative analysis is performed among algorithms and the algorithm that gives the highest accuracy is used for heart disease prediction.

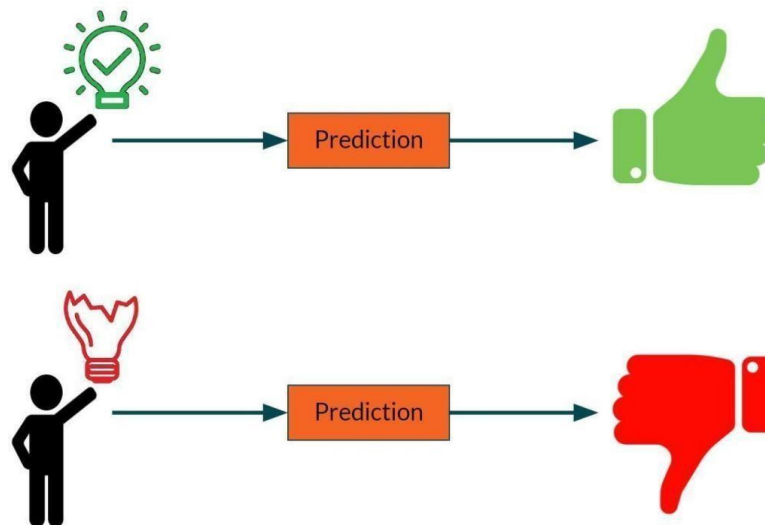


Fig 3.5: Prediction of Disease

CHAPTER-4

WORKING OF SYSTEM

4.1 SYSTEM ARCHITECTURE

The system architecture gives an overview of the working of the system.

The working of this system is described as follows:

Dataset collection is collecting data which contains patient details. Attributes selection process selects the useful attributes for the prediction of heart disease. After identifying the available data resources, they are further selected, cleaned, made into the desired form. Different classification techniques as stated will be applied on Preprocessing data to predict the accuracy of heart disease. Accuracy measure compares the accuracy of different classifiers.

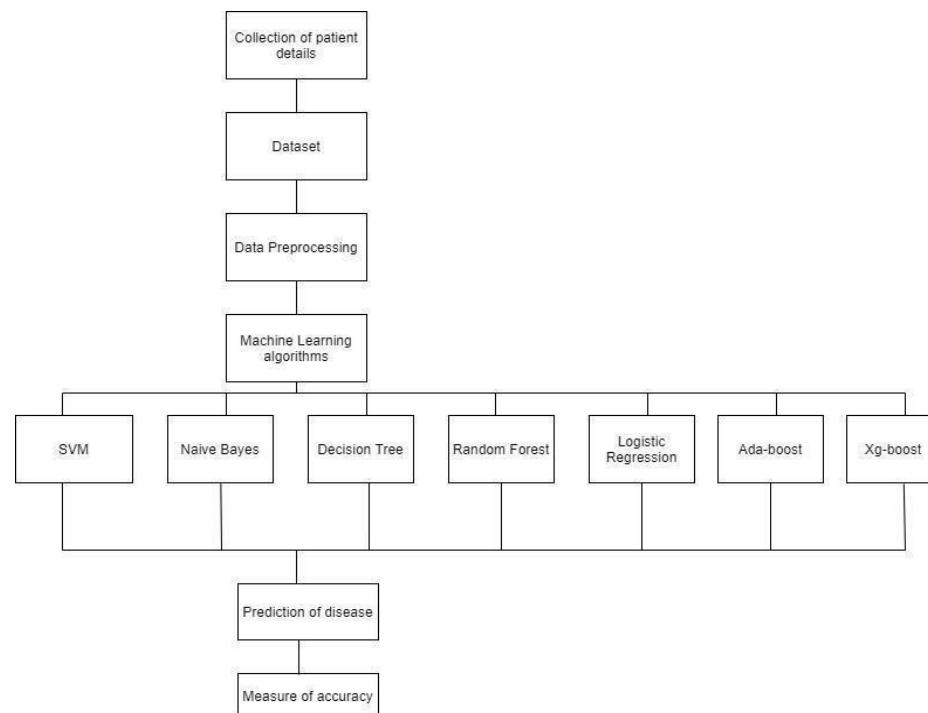


Fig 4.1: System Architecture

4.2 MACHINE LEARNING

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data

Supervised Learning

Supervised learning is the type of machine learning in which machines are trained using well "labeled" training data, and on the basis of that data, machines predict the output. The labeled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

Unsupervised learning

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of datasets, group that data according to similarities, and represent that datasets in a compressed

format.

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar to how a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Reinforcement learning

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training datasets, it is bound to learn from its experience.

4.3 ALGORITHMS

4.31 SUPPORT VECTOR MACHINE (SVM):

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In the 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

The followings are important concepts in SVM -

Support Vectors - Data Points that are closest to the hyperplane are called support vectors. Separating line will be defined with the help of these data points.

Hyperplane - As we can see in the above diagram, it is a decision plane or

space which is divided between a set of objects having different classes.

Margin - It may be defined as the gap between two lines on the closest data points of different classes. It can be calculated as the perpendicular distance from the line to support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin

Types of SVM:

SVM can be of two types:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N - the number of features) that distinctly classifies the data points.

The advantages of support vector machines are:

Effective in high dimensional spaces.

Still effective in cases where the number of dimensions is greater than the number of samples.

Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Versatile: different kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.

SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

4.32 NAIVE BAYES ALGORITHM:

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset.

Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naive Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

The Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

The Naive Bayes algorithm is comprised of two words Naive and Bayes, Which can be described as:

Naive: It is called Naive because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the basis of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

Bayes's theorem:

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

The formula for Bayes' theorem is given as:

Where,

$P(A|B)$ is Posterior probability: Probability of hypothesis A on the observed event B.

$P(B|A)$ is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

$P(A)$ is Prior Probability: Probability of hypothesis before observing the evidence. $P(B)$ is Marginal Probability: Probability of Evidence.

Types of Naive Bayes model:

There are three types of Naive Bayes Model, which are given below:

Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.

Multinomial: The Multinomial Naive Bayes classifier is used when the data

is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc. The classifier uses the frequency of words for the predictors.

Bernoulli: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Boolean's variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

4.33 DECISION TREE ALGORITHM

Decision Tree is a Supervised learning technique that can be used for both classification and regression problems, but mostly it is preferred for solving classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision Tree, there are two nodes, which are the Decision Node and Leaf Node.

Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a Decision Tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A Decision Tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into sub trees.

The Decision Tree Algorithm belongs to the family of supervised machine learning algorithms. It can be used for both a classification problem as well as for a regression problem.

The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision Tree:

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

The logic behind the decision tree can be easily understood because it shows a tree-like structure.

In Decision Tree the major challenge is to identify the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

Information Gain:

When we use a node in a Decision Tree to partition the training instances into smaller subsets, the entropy changes. Information gain is a measure of this change in entropy.

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples.

The higher the entropy the more the information content.

Gini Index:

Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower Gini index should be preferred. Sklearn supports “Gini” criteria for Gini Index and by default, it takes “gini” value.

The most notable types of Decision Tree algorithms are:-

IDichotomiser 3 (ID3):

This algorithm uses Information Gain to decide which attribute is to be used to classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.

1. **C4.5:** This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.
2. **Classification and Regression Tree (CART):** It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent variable.

Working:

In a Decision Tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the completed dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure(ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the Decision Tree node, which contains the best attribute.

4.34 RANDOM FOREST ALGORITHM

Random Forest is a supervised learning algorithm. It is an extension of machine learning classifiers which include the bagging to improve the performance of Decision Tree. It combines tree predictors, and trees are dependent on a random vector which is independently sampled. The distribution of all trees are the same. Random Forests splits nodes using the best among of a predictor subset that are randomly chosen from the node itself, instead of splitting nodes based on the variables. The time complexity of the worst case of learning with Random Forests is $O(M(dn \log n))$, where M is the number of growing trees, n is the number of instances, and d is the data dimension.

It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest consists of trees. It is said that the more trees it has, the more robust a forest is. Random Forests create Decision Trees on randomly selected data samples, get predictions from each tree and select the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random Forests have a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

Assumptions:

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- | There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- | The predictions from each tree must have very low correlations.

Algorithm Steps:

It works in four steps:

- | Select random samples from a given dataset.
- | Construct a Decision Tree for each sample and get a prediction result from each Decision Tree.
- | Perform a vote for each predicted result.
- | Select the prediction result with the most votes as the final prediction.

Advantages:

- | Random Forest is capable of performing both Classification and Regression tasks.

4.35 LOGISTIC REGRESSION ALGORITHM

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Advantages:

Logistic Regression is one of the simplest machine learning algorithms and is easy to implement yet provides great training efficiency in some cases. Also due to these reasons, training a model with this algorithm doesn't require high computation power.

The predicted parameters (trained weights) give inference about the importance of each feature. The direction of association i.e. positive or negative is also given. So we can use Logistic Regression to find out the relationship between the features.

This algorithm allows models to be updated easily to reflect new data, unlike Decision Tree or Support Vector Machine. The update can be done using stochastic gradient descent.

Logistic Regression outputs well-calibrated probabilities along with classification results. This is an advantage over models that only give the final classification as results. If a training example has a 95% probability for a class, and another has a 55% probability for the same class, we get an inference about which training examples are more accurate for the formulated problem.

Disadvantages:

Logistic Regression is a statistical analysis model that attempts to predict precise probabilistic outcomes based on independent features. On high dimensional datasets, this may lead to the model being over-fit on the training set, which means overstating the accuracy of predictions on the training set and thus the model may not be able to predict accurate results on the test set. This usually happens in the case when the model

is trained on little training data with lots of features. So on high dimensional datasets, Regularization techniques should be considered to avoid over-fitting (but this makes the model complex). Very high regularization factors may even lead to the model being under-fit on the training data.

Non linear problems can't be solved with logistic regression since it has a linear decision surface. Linearly separable data is rarely found in real world scenarios. So the transformation of non linear features is required which can be done by increasing the number of features such that the data becomes linearly separable in higher dimensions.

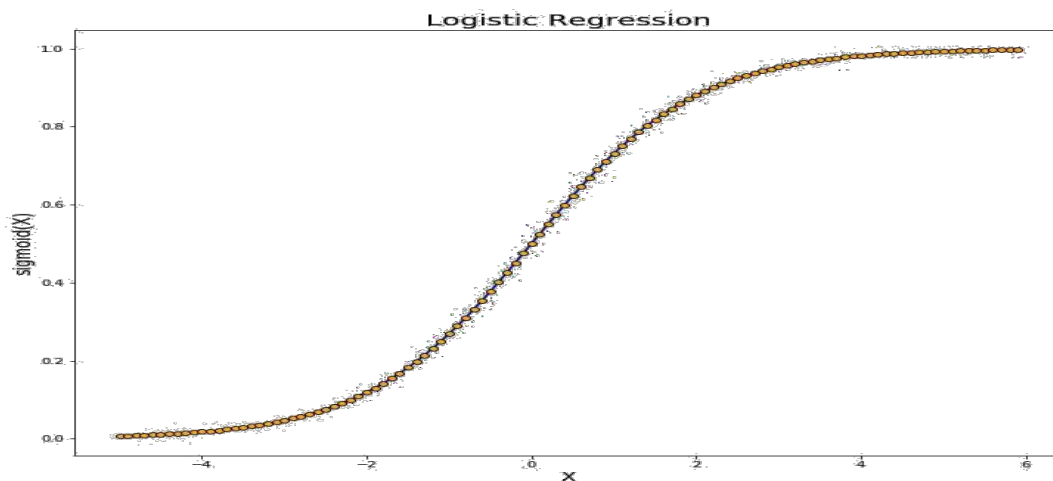


Fig 4.2: Logistic Regression

4.36 ADABOOST ALGORITHM

Adaboost was the first really successful boosting algorithm developed for the purpose of binary classification. Adaboost is short for Adaptive Boosting and is a very popular boosting technique which combines multiple “weak classifiers” into a single “strong classifier”

Algorithm:

Initially, Adaboost selects a training subset randomly.

It iteratively trains the Adaboost machine learning model by selecting the training set based on the accurate prediction of the last training.

It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.

Also, it assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.

This process iterates until the complete training data fits without any error or until reached to the specified maximum number of estimators.

To classify, perform a "vote" across all of the learning algorithms you built

Advantages:

Adaboost has many advantages due to its ease of use and less parameter tweaking when compared with the SVM algorithms. Plus Adaboost can be used with SVM though theoretically, overfitting is not a feature of Adaboost applications, perhaps because the parameters are not optimized jointly and the learning process is slowed due to estimation stage-wise. This link is useful to understand mathematics. The flexible Adaboost can also be used for accuracy improvement of weak classifiers and cases in image/text class.

Disadvantages:

Adaboost uses a progressively learning boosting technique. Hence high-quality data is needed in examples of Adaboost vs Random Forest. It is also very sensitive to outliers and noise in data requiring the elimination of these factors before using the data. It is also much slower than the XG-boost algorithm.

4.37 XGBOOST ALGORITHM

XG-boost is an implementation of Gradient Boosted decision trees. It is a type of Software library that was designed basically to improve speed and model performance. In this algorithm, decision trees are created in sequential form. Weights play an important role in XG-boost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Weight of variables predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These individual classifiers/predictors then assemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined predict.

Regularization: XG-boost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, XG-boost is also called regularized form of GBM (Gradient Boosting Machine).

While using Sci kit Learn library, we pass two hyper-parameters (alpha and lambda) to XG-boost related to regularization. alpha is used for L1 regularization and lambda is used for L2 regularization.

Parallel Processing: XG-boost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model. While using Sci kit Learn library, nthread hyper-

parameter is used for parallel processing. nthread represents number of CPU cores to be used. If you want to use all the available cores, don't mention any value for nthread and the algorithm will detect automatically.

Handling Missing Values: XG-boost has an in-built capability to handle missing values. When XG-boost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

2. **Cross Validation:** XG-boost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we have to run a grid-search and only a limited values can be tested.

3. **Effective Tree Pruning:** A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a greedy algorithm. XG-boost on the other hand make splits upto the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

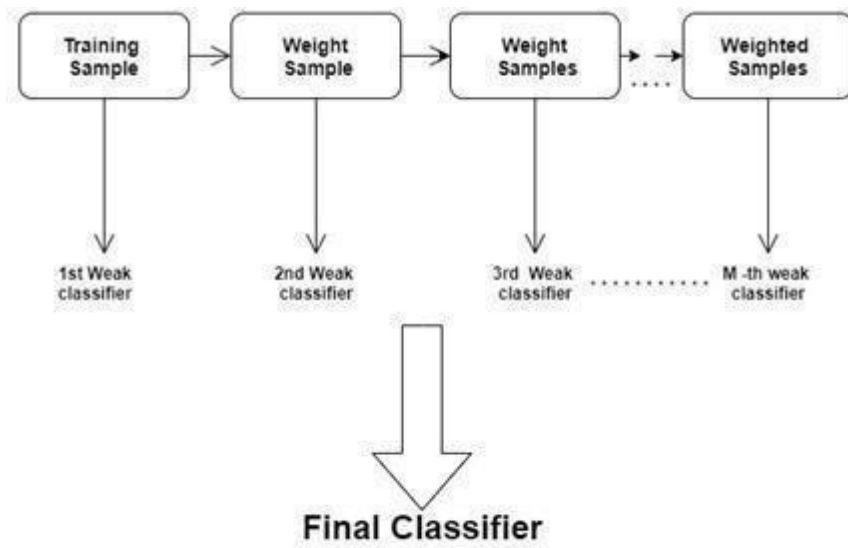


Figure : Xgboost

Fig 4.3 Xgboost

3. Cross Validation: XG-boost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we have to run a grid-search and only a limited values can be tested.

4. Effective Tree Pruning: A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a greedy algorithm. XG-boost on the other hand make splits up to the `max_depth` specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

4.4 PERFORMANCE ANALYSIS

In this project, various machine learning algorithms like SVM, Naive Bayes, Decision Tree, Random Forest, Logistic Regression, Adaboost, XG-boost are used to predict heart disease. Heart Disease UCI dataset, has a total of 76 attributes, out of those only 14 attributes are considered for the prediction of heart disease. Various attributes of the patient like gender, chest pain type, fasting blood pressure, serum cholesterol, exang, etc are considered for this project. The accuracy for individual algorithms has to be measured and whichever algorithm is giving the best accuracy, that is considered for the heart disease prediction. For evaluating the experiment, various evaluation metrics like accuracy, confusion matrix, precision, recall, and f1-score are considered.

Accuracy- Accuracy is the ratio of the number of correct predictions to the total number of inputs in the dataset. It is expressed as:

$$\text{Accuracy} = (TP + TN) / (TP + FP + FN + TN)$$

Confusion Matrix- It gives us a matrix as output and gives the total performance of the system.

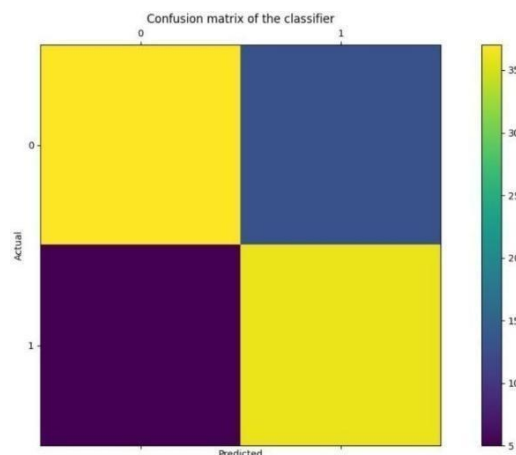


Fig 4.4: Confusion Matrix

Correlation Matrix: The correlation matrix in machine learning is used for feature selection. It represents dependency between various attributes.



Fig 4.5: Correlation matrix

Precision- It is the ratio of correct positive results to the total number of positive results predicted by the system.

It is expressed as:

Recall-It is the ratio of correct positive results to the total number of positive results predicted by the system.

It is expressed as:

F1 Score-It is the harmonic mean of Precision and Recall. It measures the test accuracy. The range of this metric is 0 to 1.

CHAPTER-5

RESULTS

ECG image processing is an important step in the analysis of ECG signals and can help improve the accuracy of diagnosis and monitoring of heart diseases. The use of web applications can make ECG image processing more accessible to clinicians and patients, as it eliminates the need for specialized hardware or software. RGB to grayscale conversion is a common technique used in image processing to simplify image analysis and reduce computational complexity. Denoising and Gaussian filtering are essential for removing noise from ECG signals and reducing the impact of artifacts, which can interfere with signal analysis. Thresholding and contouring techniques help to separate the ECG signal from the background and remove grid lines and other irrelevant information. Dimensionality reduction techniques are often used to reduce the computational cost of ECG signal analysis, as these signals can contain large amounts of data. Segmentation is a critical step in ECG signal processing, as it allows for the identification of specific waves (P, QRS, T) that can provide insights into heart function. Pre-trained models can be used to automate ECG signal analysis, saving time and reducing errors associated with manual analysis. By returning results based on the findings of the pre-trained model, the web application can provide clinicians with actionable insights into heart health and help improve patient outcomes.

CHAPTER 6

CONCLUSION

In conclusion, ECG image processing is an essential technique that can aid in the accurate diagnosis and monitoring of heart diseases. In this work, we have presented a web application that utilizes various image processing techniques to extract relevant information from ECG signals. By implementing techniques such as RGB to grayscale conversion, denoising, Gaussian filtering, thresholding, and contouring, we were able to extract signals without the grid lines and reduce the dimensionality of the signal. We then utilized segmentation to extract specific waves (P, QRS, T) and feed them into a pre-trained model for analysis. Our approach shows promising results in terms of accuracy and efficiency in analyzing ECG signals.

REFERENCES

- [1]Abdar M Kalhori SRN Sutikno T Subroto IMI Arji G 2015 Comparing Performance of Data Mining Algorithms in Prediction Heart Diseases International Journal of Electrical and Computer Engineering (IJECE) **5** 1569.
- [2] Mastery What is Deep Learning *Machine Learning* 2020 [Internet] Retrieved from <https://machinelearningmastery.com/what-is-deep-learning> on July 15th, 2021.
- [3] Maind SB Wankar 2014 P Research Paper on Basic of Artificial Neural Network *International Journal on Recent and Innovation Trends in Computing and Communication* **2** 96-100
- [4] Ranjana Raut Reader Dr. S. V. Dudul Professor & Head," Intelligent Diagnosis of Heart Diseases using Neural Network Approach" ,International Journal of Computer Applications (0975 – 8887) Volume 1– No. 2
- [5] Awan, S.M.; Riaz, M.U.; Khan, A.G.: Prediction of heart disease using artificial neural network. VFAST Trans. Software Eng. 13,102–112, 2018.
- [6] A. H. Chen, S.-Y. Huang, P.-S. Hong, C.-H. Cheng, and E.-J. Lin, "HDPS: Heart disease prediction system," in Computing in Cardiology, 2011, 2011, pp. 557-560.
- [7] Recognition (ICPR), 2019 21st International Conference on, 2019, pp. 3288-3291.
- [8] Karpagachelvi, S., Arthanari, M., & Sivakumar, M. (2018). ECG feature extraction techniques-a survey approach. arXiv preprint arXiv:1005.0957.
- [9] Vaneghi, F. M., Oladazimi, M., Shiman, F., Kordi, A., Safari, M. J., & Ibrahim, F. (2012, February). A comparative approach to ECG feature extraction methods. In Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on (pp. 252-256). IEEE.
- [10] Mazomenos, E. B., Chen, T., Acharyya, A., Bhattacharya, A., Rosengarten, J., & Maharatna, K. (2012, March). A timedomain morphology and gradient based algorithm for ECG feature extraction. In Industrial Technology (ICIT), 2012 IEEE International Conference on (pp. 117-122). IEEE.

- [11] Li, C., Zheng, C., & Tai, C. (1995). Detection of ECG characteristic points using wavelet transforms. *IEEE Transactions on biomedical Engineering*, 42(1), 21-28.
- [12] Saxena, S. C., Kumar, V., & Hamde, S. T. (2019). Feature extraction from ECG signals using wavelet transforms for disease diagnostics. *International Journal of Systems Science*, 33(13), 1073-1085.
- [13] Martínez, J. P., Almeida, R., Olmos, S., Rocha, A. P., & Laguna, P. (2020). A wavelet-based ECG delineator: evaluation on standard databases. *IEEE Transactions on biomedical engineering*, 51(4), 570-581.
- [14] Martis, R. J., Chakraborty, C., & Ray, A. K. (2020, December). using PCA and wavelet transform. In *India Conference(INDICON), 2020 Annual IEEE* (pp. 1-4). IEEE.

APPENDIX

A. SCREENSHOTS

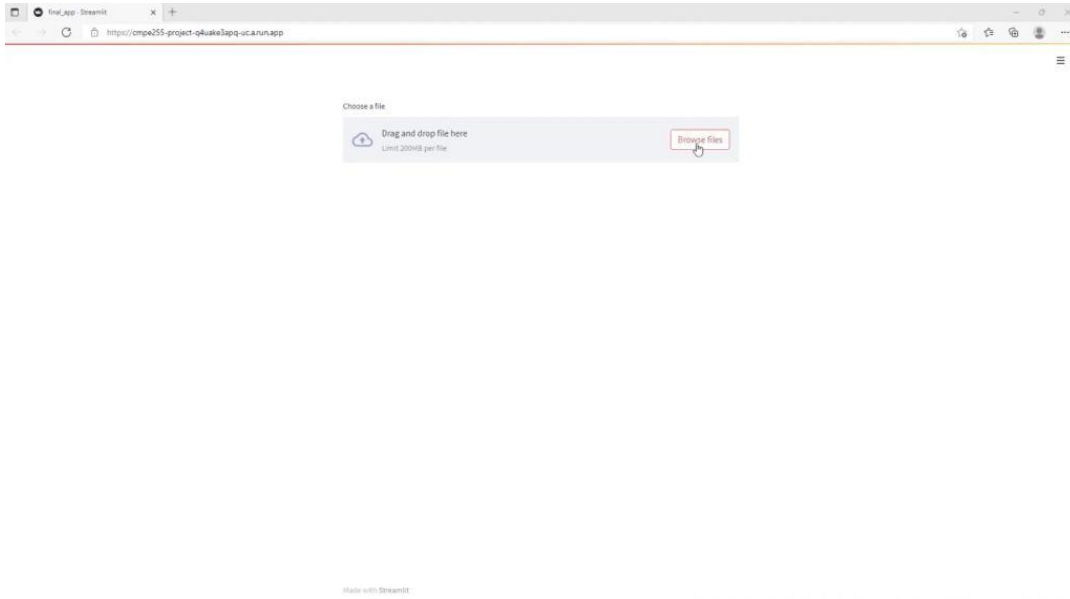


FIG: A.1 A WEBPAGE FOR SELECTING THE ECG IMAGE

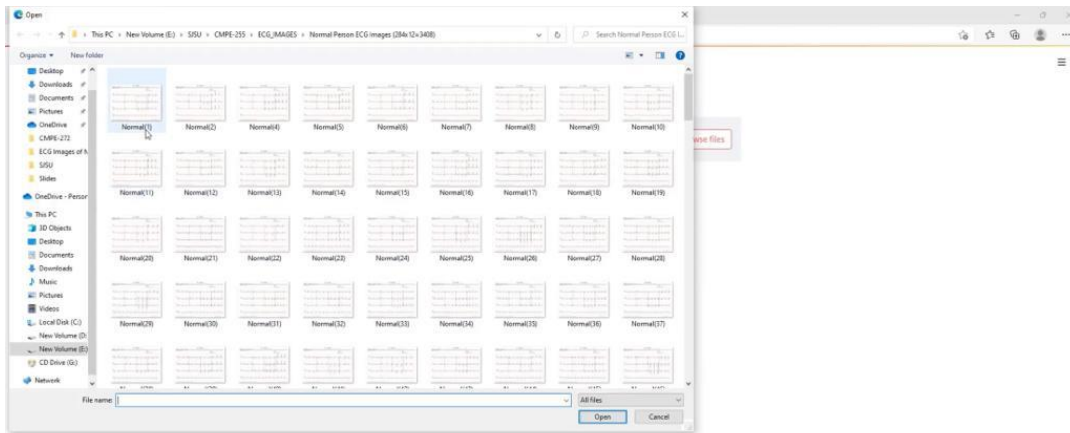


FIG: A.2 SELECTING THE ECG IMAGE

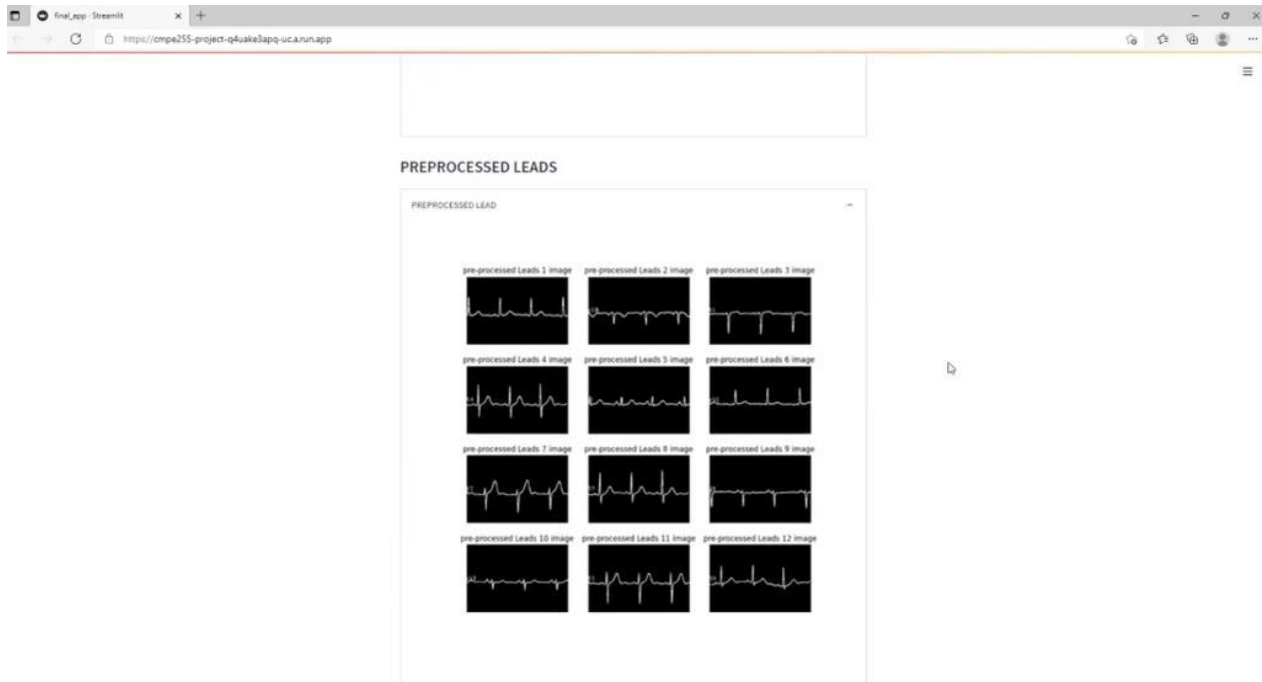


FIG: A.3 PREPROCESSED ECG IMAGE

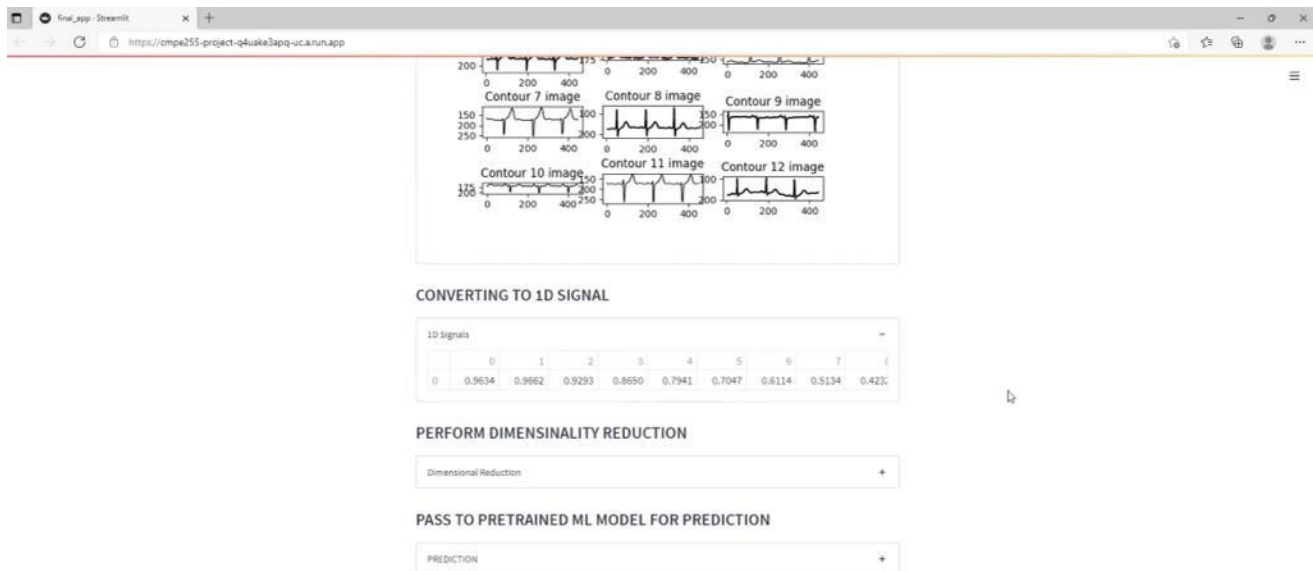


FIG: A.4 CONTOURED ECG IMAGE

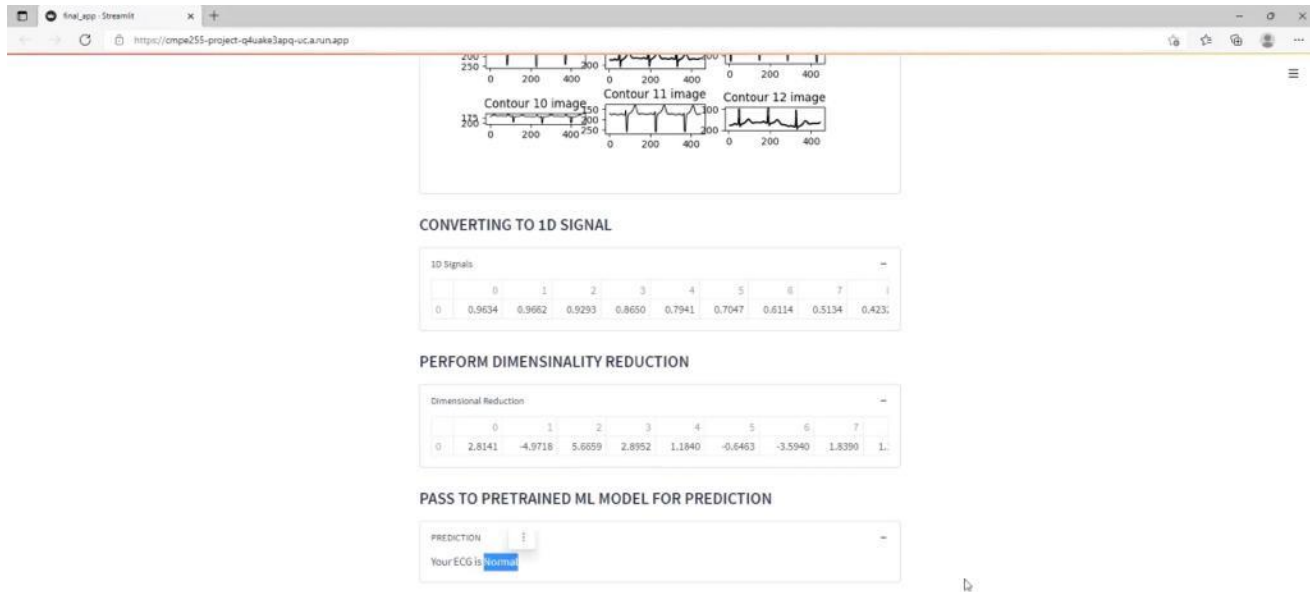


FIG: A.5 PREDICTION OF THE ECG IMAGE

B. SOURCE CODE:

```
!pip install streamlit

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

uploaded_file = st.file_uploader("Choose a file")

if uploaded_file is not None:
    image=imread(uploaded_file)
    image_gray = color.rgb2gray(image)
    image_gray=resize(image_gray, (1572, 2213))
```

```

#checkign if we parse the user image and similar to our format
image1=imread('/content/PMI(1).jpg')
image1 = color.rgb2gray(image1)
image1=resize(image1,(1572,2213))

image2=imread('/content/HB(6).jpg')
image2 = color.rgb2gray(image2)
image2=resize(image2,(1572,2213))

image3=imread('/content/Normal(1).jpg')
image3 = color.rgb2gray(image3)
image3=resize(image2,(1572,2213))

image4=imread('/content/MI(1).jpg')
image4 = color.rgb2gray(image4)
image4=resize(image2,(1572,2213))

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

if similarity_score > 0.70:
    st.image(image)
    """#### **GRAY SCALE IMAGE**"""
    my_expander = st.expander(label='Gray SCALE IMAGE')
    with my_expander:
        st.image(image_gray)
        """#### **DIVIDING LEADS**"""
        #dividing the ECG leads from 1-13 from the above image
        Lead_1 = image[300:600, 150:643]
        Lead_2 = image[300:600, 646:1135]
        Lead_3 = image[300:600, 1140:1625]
        Lead_4 = image[300:600, 1630:2125]
        Lead_5 = image[600:900, 150:643]
        Lead_6 = image[600:900, 646:1135]
        Lead_7 = image[600:900, 1140:1625]
        Lead_8 = image[600:900, 1630:2125]
        Lead_9 = image[900:1200, 150:643]
        Lead_10 = image[900:1200, 646:1135]
        Lead_11 = image[900:1200, 1140:1625]
        Lead_12 = image[900:1200, 1630:2125]
        Lead_13 = image[1250:1480, 150:2125]

        Leads=[Lead_1,Lead_2,Lead_3,Lead_4,Lead_5,Lead_6,Lead_7,Lead_8,Lead_9,Lead_10,Lead_11,Lead_12,Lead_13]
        #plotting lead 1-12
        fig , ax = plt.subplots(4,3)
        fig.set_size_inches(10, 10)
        x_counter=0
        y_counter=0

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

```

```

    ax[x_counter][y_counter].axis('off')
    ax[x_counter][y_counter].set_title("Leads {}".format(x+1))
    y_counter+=1

fig.savefig('Leads_1-12_figure.png')
fig1 , ax1 = plt.subplots()
fig1.set_size_inches(10, 10)
ax1.imshow(Lead_13)
ax1.set_title("Leads 13")
ax1.axis('off')
fig1.savefig('Long_Lead_13_figure.png')
my_expander1 = st.expander(label='DIVIDING LEAD')
with my_expander1:
    st.image('Leads_1-12_figure.png')
    st.image('Long_Lead_13_figure.png')

"""#### **PREPROCESSED LEADS**"""
fig2 , ax2 = plt.subplots(4,3)
fig2.set_size_inches(10, 10)
#setting counter for plotting based on value
x_counter=0
y_counter=0

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

    #creating binary image based on threshold
    binary_global = blurred_image < global_thresh
    #resize image
    binary_global = resize(binary_global, (300, 450))
    if (x+1)%3==0:
        ax2[x_counter][y_counter].imshow(binary_global,cmap="gray")
        ax2[x_counter][y_counter].axis('off')
        ax2[x_counter][y_counter].set_title("pre-processed Leads {}
image".format(x+1))
        x_counter+=1
        y_counter=0
    else:
        ax2[x_counter][y_counter].imshow(binary_global,cmap="gray")
        ax2[x_counter][y_counter].axis('off')
        ax2[x_counter][y_counter].set_title("pre-processed Leads {}
image".format(x+1))
        y_counter+=1
fig2.savefig('Preprocessed_Leads_1-12_figure.png')

#plotting lead 13
fig3 , ax3 = plt.subplots()
fig3.set_size_inches(10, 10)
#converting to gray scale
grayscale = color.rgb2gray(Lead_13)
#smoothing image
blurred_image = gaussian(grayscale, sigma=0.7)
#thresholding to distinguish foreground and background
#using otsu thresholding for getting threshold value
global_thresh = threshold_otsu(blurred_image)
print(global_thresh)

```

```

#creating binary image based on threshold
binary_global = blurred_image < global_thresh
ax3.imshow(binary_global, cmap='gray')
ax3.set_title("Leads 13")
ax3.axis('off')
fig3.savefig('Preprocessed_Leads_13_figure.png')

my_expander2 = st.expander(label='PREPROCESSED LEAD')
with my_expander2:
    st.image('Preprocessed_Leads_1-12_figure.png')
    st.image('Preprocessed_Leads_13_figure.png')

"""#### **EXTRACTING SIGNALS (1-13) **"""
fig4 , ax4 = plt.subplots(4,3)
fig4.set_size_inches(10, 10)
x_counter=0
y_counter=0
for x,y in enumerate(Leads[:len(Leads)-1]):
    #converting to gray scale
    grayscale = color.rgb2gray(y)
    #smoothing image
    blurred_image = gaussian(grayscale, sigma=0.9)
    #thresholding to distinguish foreground and background
    #using otsu thresholding for getting threshold value
    global_thresh = threshold_otsu(blurred_image)

    #creating binary image based on threshold
    binary_global = blurred_image < global_thresh
    #resize image
    binary_global = resize(binary_global, (300, 450))
    #finding contours
    contours = measure.find_contours(binary_global,0.8)
    contours_shape = sorted([x.shape for x in contours])[:,::-1][0:1]
    for contour in contours:
        if contour.shape in contours_shape:
            test = resize(contour, (255, 2))
            if (x+1)%3==0:
                ax4[x_counter][y_counter].plot(test[:, 1], test[:,
0],linewidth=1,color='black')
                ax4[x_counter][y_counter].axis('image')
                ax4[x_counter][y_counter].set_title("Contour {} image".format(x+1))
                x_counter+=1
                y_counter=0
            else:
                ax4[x_counter][y_counter].plot(test[:, 1], test[:,
0],linewidth=1,color='black')
                ax4[x_counter][y_counter].axis('image')
                ax4[x_counter][y_counter].set_title("Contour {} image".format(x+1))
                y_counter+=1

    #scaling the data and testing
    lead_no=x
    scaler = MinMaxScaler()
    fit_transform_data = scaler.fit_transform(test)
    Normalized_Scaled=pd.DataFrame(fit_transform_data[:,0], columns = ['X'])
    Normalized_Scaled=Normalized_Scaled.T
    #scaled_data to CSV
    if
(os.path.isfile('scaled_data_1D_{lead_no}.csv'.format(lead_no=lead_no+1))):

Normalized_Scaled.to_csv('Scaled_1DLead_{lead_no}.csv'.format(lead_no=lead_no+
1), mode='a', index=False)

```

```

else:

Normalized_Scaled.to_csv('Scaled_1DLead_{lead_no}.csv'.format(lead_no=lead_no+
1),index=False)

fig4.savefig('Contour_Leads_1-12_figure.png')
my_expander3 = st.expander(label='CONOTUR LEADS')
with my_expander3:
    st.image('Contour_Leads_1-12_figure.png')

    """#### **CONVERTING TO 1D SIGNAL**"""
    #lets try combining all 12 leads
    test_final=pd.read_csv('/content/Scaled_1DLead_1.csv')
    location= '/content/'
    for files in natsorted(os.listdir(location)):
        if files.endswith(".csv"):
            if files!='Scaled_1DLead_1.csv':
                df=pd.read_csv('/content/{0}'.format(files))
                test_final=pd.concat([test_final,df],axis=1,ignore_index=True)

    st.write(test_final)
    """#### **PASS TO ML MODEL FOR PREDICTION**"""
    loaded_model = joblib.load('/content/model_test.pkl')
    result = loaded_model.predict(test_final)
    if result[0] == 0:
        st.write("You ECG corresponds to Myocardial Infarction")

    if result[0] == 1:
        st.write("You ECG corresponds to Abnormal Heartbeat")

    if result[0] == 2:
        st.write("Your ECG is Normal")

    if result[0] == 3:
        st.write("You ECG corresponds to History of Myocardial Infarction")

    else:
        st.write("Sorry Our App won't be able to parse this image format right
now!!!. Pls check the image input sample section for supported images")

```

Overwriting final_app.py

```
! streamlit run final_app.py & npx localtunnel --port 8501
```

[REDACTED]