“HEART DISEASE PREDICTION BASED ON MACHINE LEARNING”

A PROJECT REPORT SUBMITTED FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

In

“COMPUTER SCIENCE AND ENGINEERING”

SUBMITTED BY:

AMAN MATHUR (2020021023)
KISHAN KUMAR JAISWAL (2020021071)
ARNISH VERMA (2020021031)

UNDER THE SUPERVISION OF:

Assistant Prof. Dr. Sushil Kumar Saroj

Department of
Computer Science and Engineering,
Madan Mohan Malaviya University of Technology,
Gorakhpur, Uttar Pradesh (INDIA) – 273010

<u>CANDIDATE'S DECLARATION</u>

With the exception of instances where proper attribution has been made within the text, we hereby affirm that this submission is our genuine work and efforts and that, to the best of our belief, it does not contain any content that has already been published or written by another person. It also does not contain any material that has been approved in large part for any degree program from a university or other institution of higher learning.

Aman Mathur
Roll No. 2020021023
Sign :

Kishan Kumar Jaiswal
Roll No. 2020021071
Sign :

Arnish Verma
Roll No. 2020021031
Sign :

Bachelor of Technology
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project work entitled designing “Heart Disease Prediction Based On Machine Learning” is the bonafide work carried out by Aman Mathur (2020021023), Kishan Kumar Jaiswal (2020021071), Arnish Verma (2020021031) students of Bachelor of Technology in the Department of Computer Science and Engineering, Madan Mohan Malaviya University of Technology, Gorakhpur, during the academic session 2023-24, in partial fulfilment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering is an authentic work carried out by them under my supervision and guidance. To the best of my knowledge, the matter embodied in the project report has not been submitted to any other University/Institute for the award of any degree.

Supervisor
Assistant Prof. Dr. Sushil Kumar Saroj
Associate Professor
Department of CSE
M.M.M.U.T. Gorakhpur

Date:
Place:

APPROVAL SHEET

The report entitled "**HEART DISEASE PREDICTION BASED ON MACHINE LEARNING**" is approved for the degree of Bachelor of Technology.

Examiner

____

____

____

<u>Supervisor</u>

Associate Prof. Dr. Sushil Kumar Saroj

<u>Head of Department (CSED)</u>

Dr. Udai Shankar

Dean, Research & Development
or Other Dean/Professor to be
nominated by the Vice Chancellor
in his absence

Date:

Place: Gorakhpur

ACKNOWLEDGEMENT

It is our great fortune that we have got an opportunity to carry out this project work under the supervision of Assistant Prof. Dr. Sushil Kumar Saroj, Department of Computer Science and Engineering, Madan Mohan Malaviya University of Technology, Gorakhpur. We express our sincere thanks and deepest sense of gratitude to our guide for his constant support, unparalleled guidance, and illimitable encouragement. We opted to convey our gratitude to Prof. Dr. Udai Shankar, HOD, Department of Computer Science and Engineering, MMMUT, and the ascendancy of MMMUT for providing all kinds of infrastructural facilities for the research work. We would be conveying our gratitude to all the faculty members and staff of the Department of Computer Science for their wholehearted cooperation to make this work turn into authenticity.

Aman Mathur
Roll No. 2020021023
Sign :

Kishan Kumar Jaiswal
Roll No. 2020021071
Sign :

Arnish Verma
Roll No. 2020021031
Sign :

LIST OF FIGURES

TABLE OF CONTENTS

ABSTRACT

Major and avoidable biggest causes of death in the globe are heart disease. Although traditional techniques of testing can be intrusive and expensive, early detection and identification of cardiac disease can significantly improve patient outcomes. In this research, our goal is to employ machine learning techniques to predict the existence of heart disease in patients based on non-invasive medical parameters. We gathered and cleaned a dataset of patient data, which included demographic data, vital signs, and test findings. The dataset was then used to train a number of machine learning models. On the test set, our final model performed with an accuracy of more than 85%.

Additionally, we assessed the effectiveness of our model using a numberof parameters. We discovered that our model had good accuracy, which suggested that it had few false positives. However, we also discovered that recall, which gauges the proportion of true positives that the model properly identifies, might be improved.

To enhance the accuracy of the predictions, we looked at more sophisticated algorithms like Random Forest in addition to the models already described. To enhance the performance of the model, we also carried out feature selection and engineering. This entails assessing each feature's significance and deleting any redundant or pointless characteristics from the dataset. Randon Forest Algorithm is one the most used algorithm in the field of Machine Learning which help provide the outcome of the final live result on the the basis of the trained model and and test data it is also help to make the model more productive with the the help of multiple existing researches and libraries done by the communities to enhanced the working od Machine learning. It also create and improve the medical fields and much many more do for the patients and also it is cost effective and easily available for all to reach and use with easy implementation for common users.

INTRODUCTION

Heart disease is a critical health issue worldwide, responsible for a significant number of deaths each year. Early detection and accurate prediction of heart disease can greatly improve patient outcomes and reduce mortality rates. In recent years, the integration of machine learning algorithms in healthcare systems has shown tremendous potential in predicting and diagnosing various diseases, including heart disease. Machine learning algorithms have the capability to analyzelarge amounts of patient data and identify complex patterns that may be indicative of heart disease. By leveraging this technology, healthcare professionals can make more informed decisions and provide personalized treatment plans to individuals at risk of developing heart disease. The goal of this project is to develop a heart disease prediction system using machine learning algorithms. By utilizing a diverse set of patient data, including demographic information, medical history, and diagnostic test results, we aim to create a robust and accurate model capable of identifying individuals who are at a high risk of developing heart disease. The system will employ a range of machine learning algorithms, such as logistic regression, support vector machines, random forests, or neural networks, to analyze the data and generate predictions. These algorithms will be trained on a carefully curated dataset consisting of historical patient records, where the presence or absence of heart disease is known. The model will learn from this dataset and extract valuable insights to make predictions on new, unseen patient data. The potential impact of this heart disease prediction system is significant. It can assist healthcare professionals in identifying patients who require further examination and intervention, allowing for early detection and preventive measures to be implemented. Moreover, by providing personalized risk assessments, the system can empower individuals to take proactive steps towards maintaining a heart-healthy lifestyle and seek appropriate medical care. In conclusion, the integration of machine learning algorithms in heart disease prediction systems has the potential to revolutionize the field of cardiovascular medicine. By harnessing the power of data analysis and predictive modeling, we can enhance the accuracy and efficiency of heart disease diagnosis, leading to improved patient outcomes and a reduction in the global burden of this devastating condition.

PROBLEM STATEMENT

The extremely crucial and critical internal part in human anatomy is the circulatory system organ that is heart. It influences other internal organs if they are not functioning properly. According to statistics, heart attacks affect 7,000,000 lives annually. According to a WHO assessment, 17.9 million deaths were reported in 2018 because of CVDS. Heart disease accounts for 31% of all fatalities worldwide each year. Blood shortage in the body of a person makes the heart dysfunctional and subsequently ceases functioning, causing life threats.

Heart related Alignment is of two types:

1. Heart Attack: When the blood veins in the heart are abruptly clogged, a heart attack happens.
2. Heart Fail: When Body Stops Pumping through Articulatios

INPORTANT PARAMETER FOR CARDIOVASCULAR ISSUES [4]

1. Smoking
2. Elevated BP
3. Bad Insulin level
4. Diabetics patient
5. obesity
6. Lack Of Exercise
7. Dysmetabolic Defects

RISK FACTOR FOR HEART DISEASE

1. Genes Background
2. More Age (>55)
3. preeclampsia criteria on body (medical history)

MANIFESTATION OF CARDIOVASCULAR ALIGNMENTS

1. Unstable Angina
2. Arrhythmia
3. Elevated BP
4. Back pain

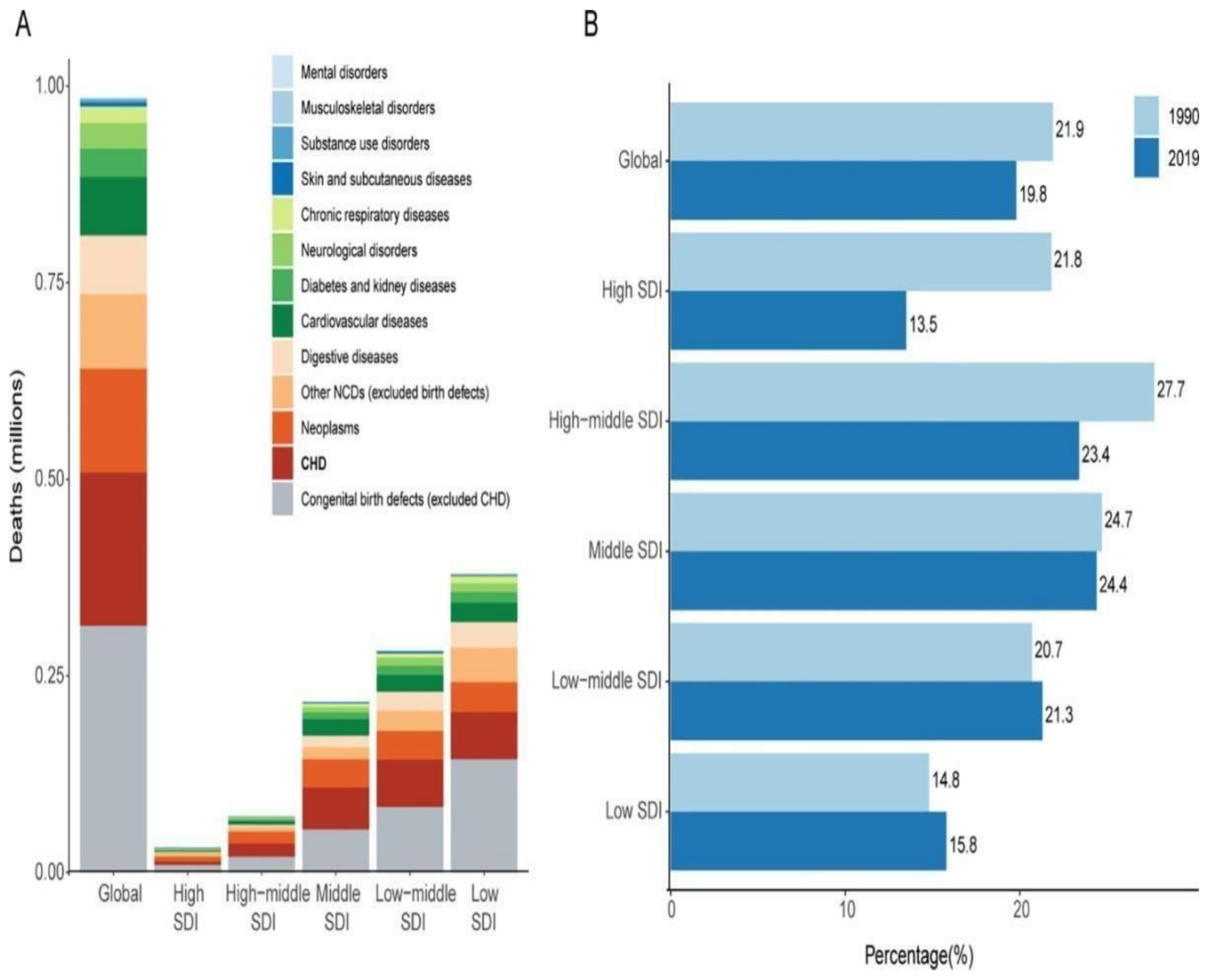
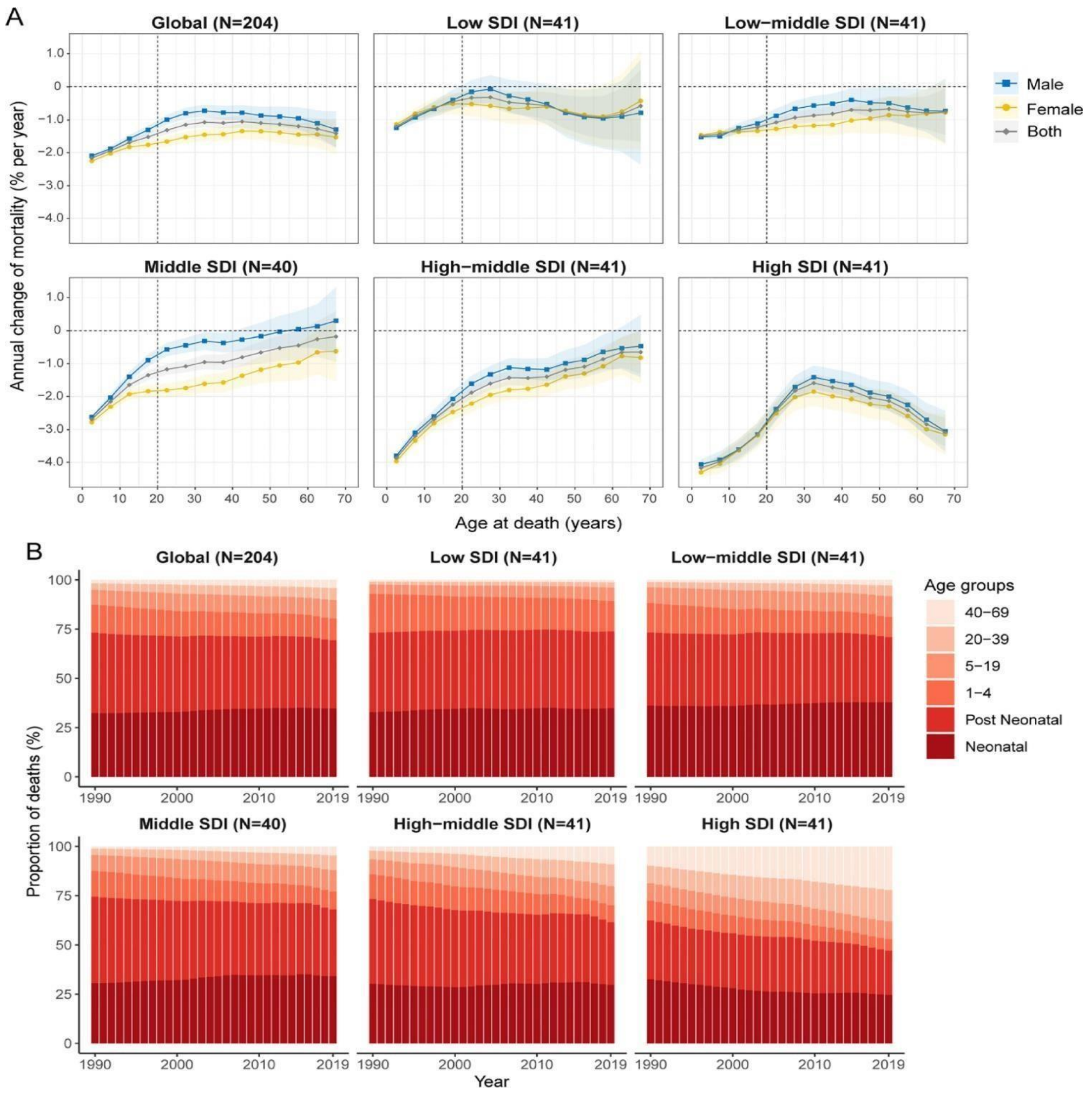


Fig..1



A
Global (N=204)
Low SDI (N=41)
Low−middle SDI (N=41)
Middle SDI (N=40)
High−middle SDI (N=41)
High SDI (N=41)
Annual change of mortality (% per year)
Age at death (years)
Male
Female
Both
B
Proportion of deaths (%)
Year
Age groups
40−69
20−39
5−19
1−4
Post Neonatal
Neonatal

Fig.2. .

LITERATURE REVIEW

Machine learning-based techniques and algorithms have been proposed and published by researchers to diagnose heart-related medical disorders, specifically in recognizing heart illness or alignment. This paper aims to highlight the growing importance of these methodologies. These algorithms play a crucial role in improving accuracy and efficiency in identifying and diagnosing heart illnesses. The research highlights the significance of incorporating machine learning in clinical settings and its potential impact on enhancing diagnostic capabilities. By exploring various algorithms, the paper aims to showcase the advancements in this field and promote their adoption in healthcare practices. Ultimately, these algorithms contribute to better patient outcomes and more effective screening for heart-related disorders. The paper aims to shed light on the implications and benefits of integrating machine learning into clinical decision-making processes.

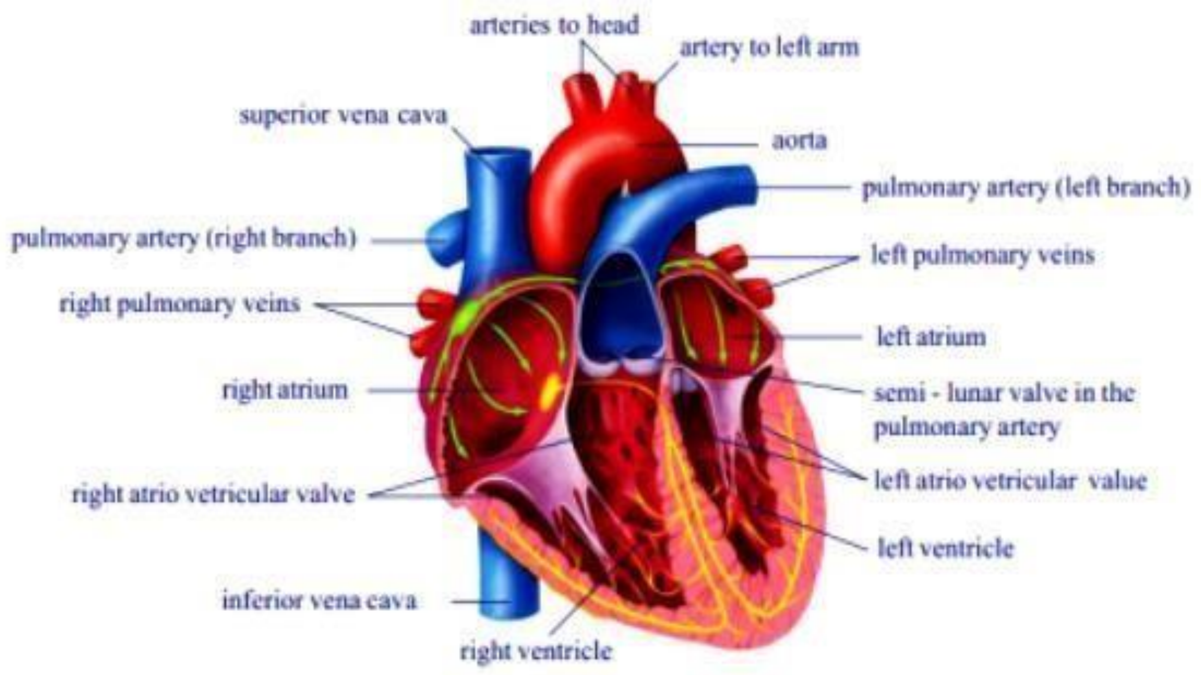


Fig.3

In ref [1]. The authors proposed a straightforward approach for predicting coronary heart disease through a heart disease prediction system. They selected thirteen clinical criteria to categorize the cardiac disease. The system demonstrated a prediction accuracy ranging from 75% to 80%. The components of the Heart Disease Prediction System include clinical data, ROC curves, and computational displays, which are presented in this study.

In ref [2] In this study, various approaches, including KNN (k-nearest neighbors), decision trees, linear regression, and SVM (support vector machines), were applied to predict cardiovascular illness. The accuracy of each algorithm was evaluated. All datasets used for prediction were obtained from the UCI repository site. The implementation of the techniques was done using the Python programming language, utilizinga Jupyter notebook for each algorithm's processing. Experimental resultsrevealed that the k-nearest neighbor algorithm exhibited the highestaccuracy of 80% in predicting heart disease. It was followed by support vector machines with an accuracy of 78%, decision trees with 79%, and linear regression with 78%.

In ref [3] The detection of heart problems involves the utilization of the popular Flask framework and the Random Forest (R.F.) machine learning approaches. Cardiovascular disorders or abnormalities become evident when there is arterial obstruction. The progression of heart disorders corresponds to the worsening condition of the blockage. Age, sex, blood pressure, and blood sugar levels are among the predicted values considered in the prediction process. Experimental results demonstrate that the predictions generated by the Flask Web system and the Random Forest machine learning algorithm consistently outperformed predictions made using alternative methods.

EXISTING SYSTEM

Heart disease is commonly referred to as a silent killer as it can lead to death without exhibiting obvious symptoms. The nature of this disease has created significant concerns and increased anxiety about its consequences. Therefore, ongoing efforts are being made to predict the likelihood of developing this life-threatening condition. Various tools and techniques are continually being explored to meet the current healthcare needs, and Machine Learning techniques have emerged as a valuable asset in this pursuit. Although heart disease can manifest in different forms, there are core risk factors that significantly influence an individual's susceptibility to the disease. These factors provide crucial insights into the risk assessment of body defect . By gathering data from diverse sources, organizing it into suitable categories, and conducting thorough analysis, meaningful information can be extracted. Through the utilization of comprehensive datasets and the application of advanced algorithms. Consequently, they empower healthcare professionals to make informed decisions and take proactive measures for early prediction and control of heart disease. Embracing the principle of "prevention is better than cure," early detection and management have the potential to reduce mortality rates Heart disease is a broad term that encompasses various conditions affecting the heart. One of the significant outcomes of heart disease can be death. The number of deaths caused by heart disease can vary depending on factors such as region, population, access to healthcare, lifestyle choices, and advancements in medical treatments. To provide specific information about deaths from heart disease,. Additionally, it's worth noting that my knowledge cutoff is in September 2021, so I may not have the most up-to-date statistics on this topic. For the most recent and accurate information, it's recommended to refer to reliable sources like health organizations, government agencies, or medical journals [3].

PROPOSED SYSTEM

The heart disease prediction system follows a step-by-step process, starting by gathering of information field selection of pertinent fields. The collected data is then prepared through preprocessing techniques to ensure it meets the necessary format. Subsequently, the data is deviate into distinct sets for training-testing purposes. Machine learning algorithms are applied to train the model using the training data.

Dataset Collection: Model begins by acquiring the relevant datasetcontaining the required information for heart disease prediction.

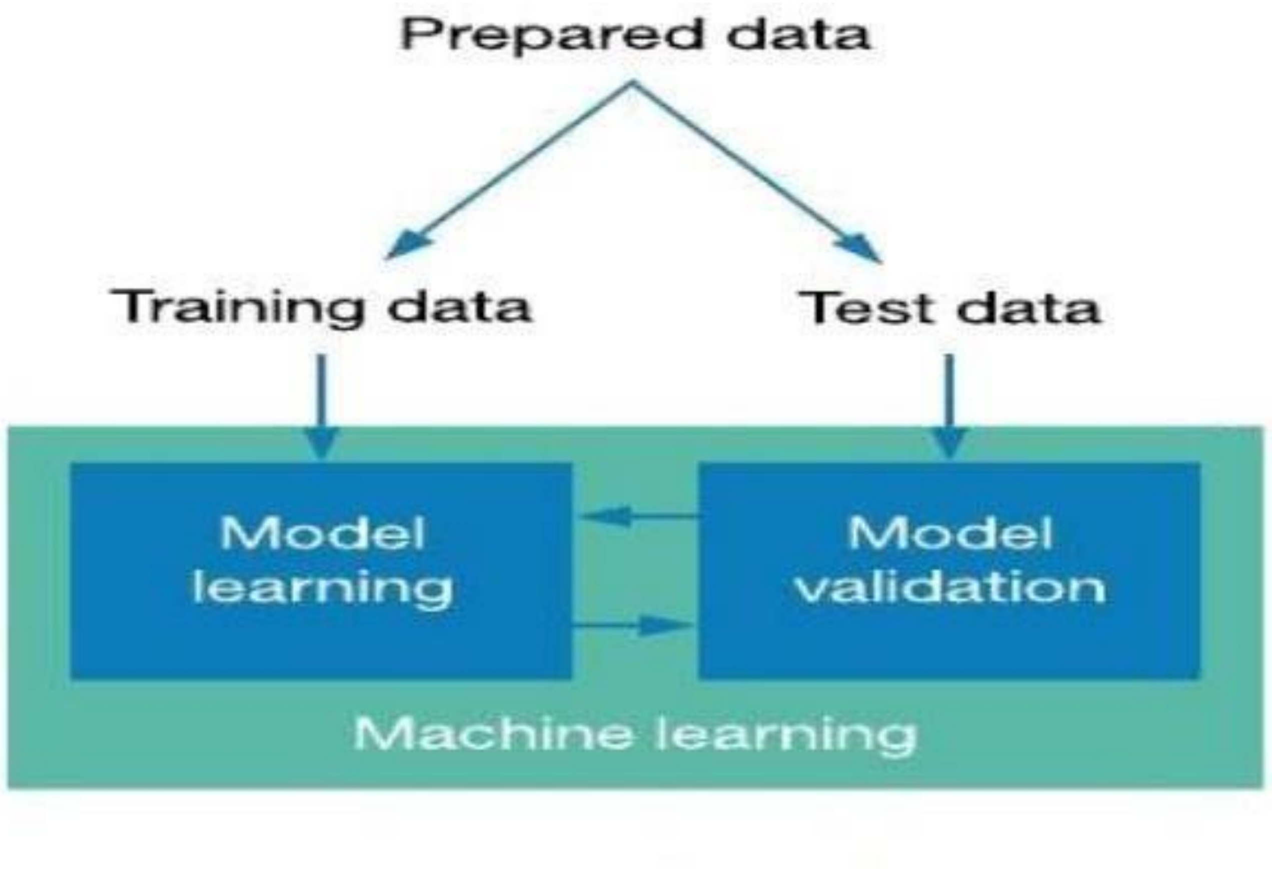
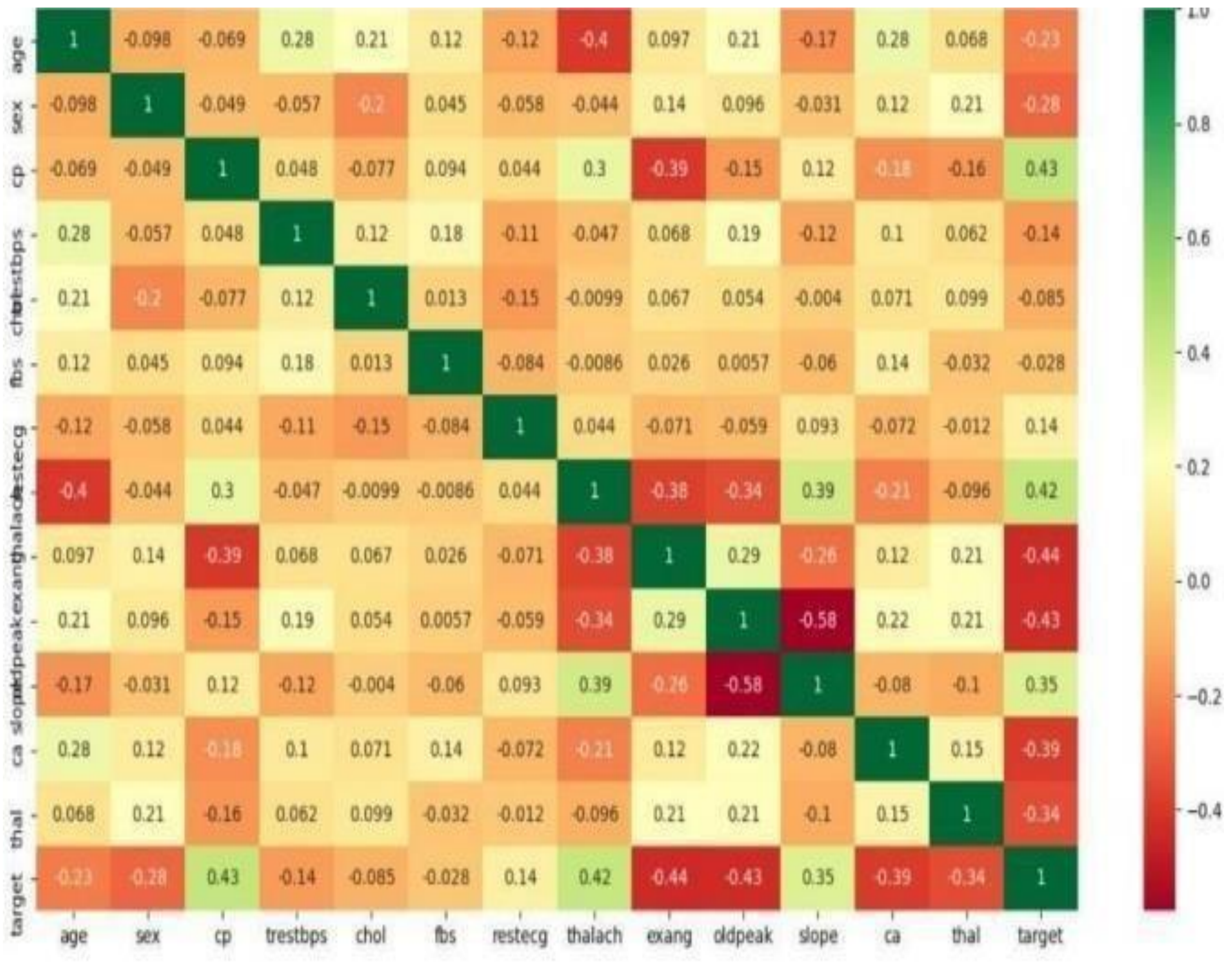


Figure: Collection of Data

Fig. 4.

Attribute Selection: From the collected dataset, the system identifies and chooses effective attributes (fields) that support to predicting heart disease. This step focuses on reducing data dimensionality andhighlighting key factors.



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.098	-0.069	0.28	0.21	0.12	-0.12	-0.4	0.097	0.21	-0.17	0.28	0.068	-0.23
sex	-0.098	1	-0.049	-0.057	-0.2	0.045	-0.058	-0.044	0.14	0.096	-0.031	0.12	0.21	-0.28
cp	-0.069	-0.049	1	0.048	-0.077	0.094	0.044	0.3	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trestbps	0.28	-0.057	0.048	1	0.12	0.18	-0.11	-0.047	0.068	0.19	-0.12	0.1	0.062	-0.14
chol	0.21	-0.2	-0.077	0.12	1	0.013	-0.15	-0.0099	0.067	0.054	-0.004	0.071	0.099	-0.085
fbs	0.12	0.045	0.094	0.18	0.013	1	-0.084	-0.0086	0.026	0.0057	-0.06	0.14	-0.032	-0.028
restecg	-0.12	-0.058	0.044	-0.11	-0.15	-0.084	1	0.044	-0.071	-0.059	0.093	-0.072	-0.012	0.14
thalach	-0.4	-0.044	0.3	-0.047	-0.0099	-0.0086	0.044	1	-0.38	-0.34	0.39	-0.21	-0.096	0.42
exang	0.097	0.14	-0.39	0.068	0.067	0.026	-0.071	-0.38	1	0.29	-0.26	0.12	0.21	-0.44
oldpeak	0.21	0.096	-0.15	0.19	0.054	0.0057	-0.059	-0.34	0.29	1	-0.58	0.22	0.21	-0.43
slope	-0.17	-0.031	0.12	-0.12	-0.004	-0.06	0.093	0.39	-0.26	-0.58	1	-0.08	-0.1	0.35
ca	0.28	0.12	-0.18	0.1	0.071	0.14	-0.072	-0.21	0.12	0.22	-0.08	1	0.15	-0.39
thal	0.068	0.21	-0.16	0.062	0.099	-0.032	-0.012	-0.096	0.21	0.21	-0.1	0.15	1	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.085	-0.028	0.14	0.42	-0.44	-0.43	0.35	-0.39	-0.34	1

Figure: Correlation matrix

Fig.5

Data Preprocessing: The selected data undergoes preprocessing techniques to handle missing values, outliers, and inconsistencies. Approaches such as data-cleansing, nor_malization, and transformation are applied, ensure data quality-consistency.

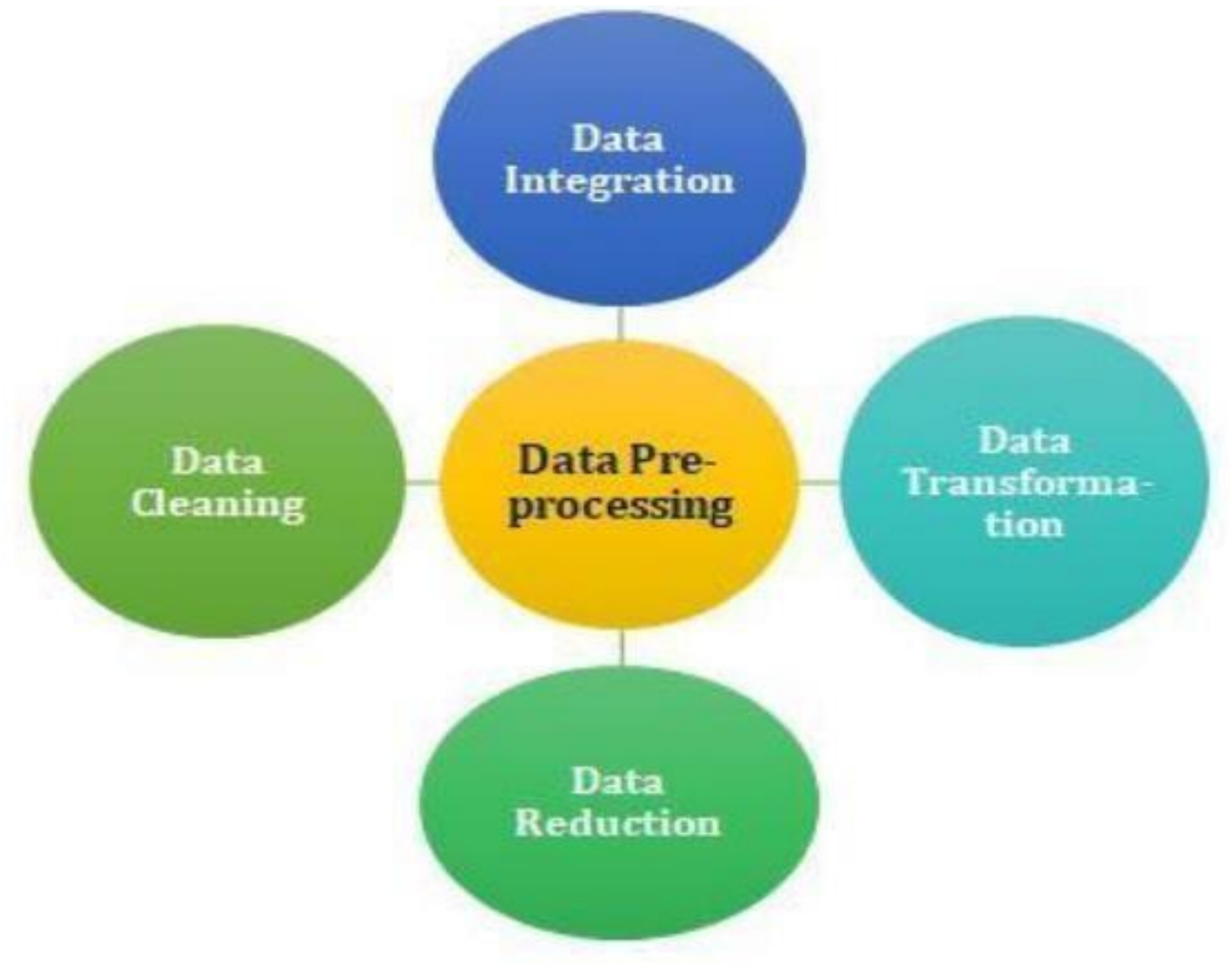


Figure: Data Pre-processing

Fig. 6

Data Balancing: In cases where the dataset exhibits imbalanced class distribution (e.g., presence or absence of heart disease), data balancing techniques like oversampling or undersampling are employed. This helps address biases and enhance the performance of the predictionmodel.

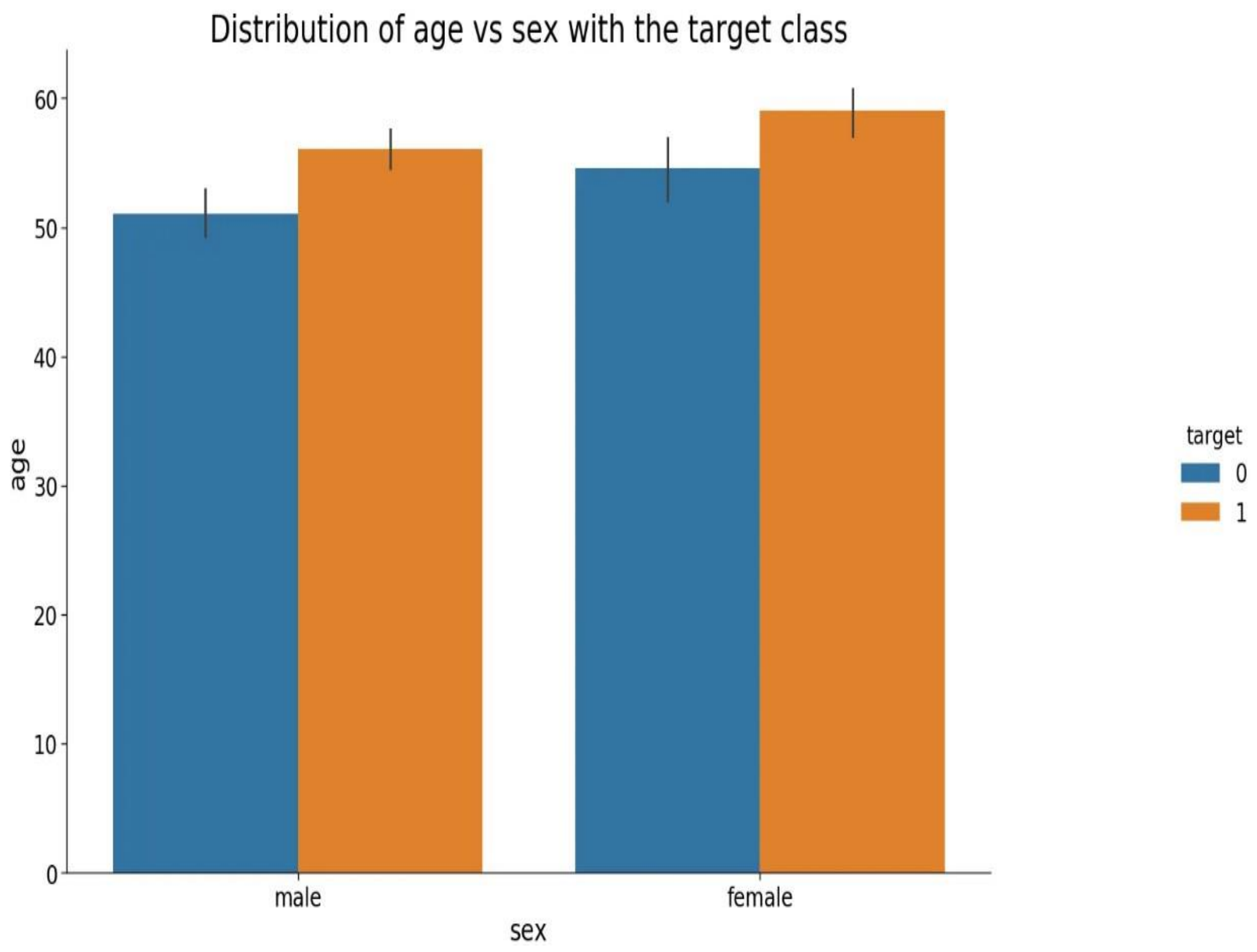


Fig. 7.

Disease Prediction: This module encompasses the model of machine learning algorithms to the preprocessed and balanced data. Thealgorithms analyze patterns and relationships within the data to developa predictive model. By incorporating these modules, the heart disease prediction system leverages machine learning techniques to facilitate accurate predictions and support healthcare professionals in making informed decisions [9].

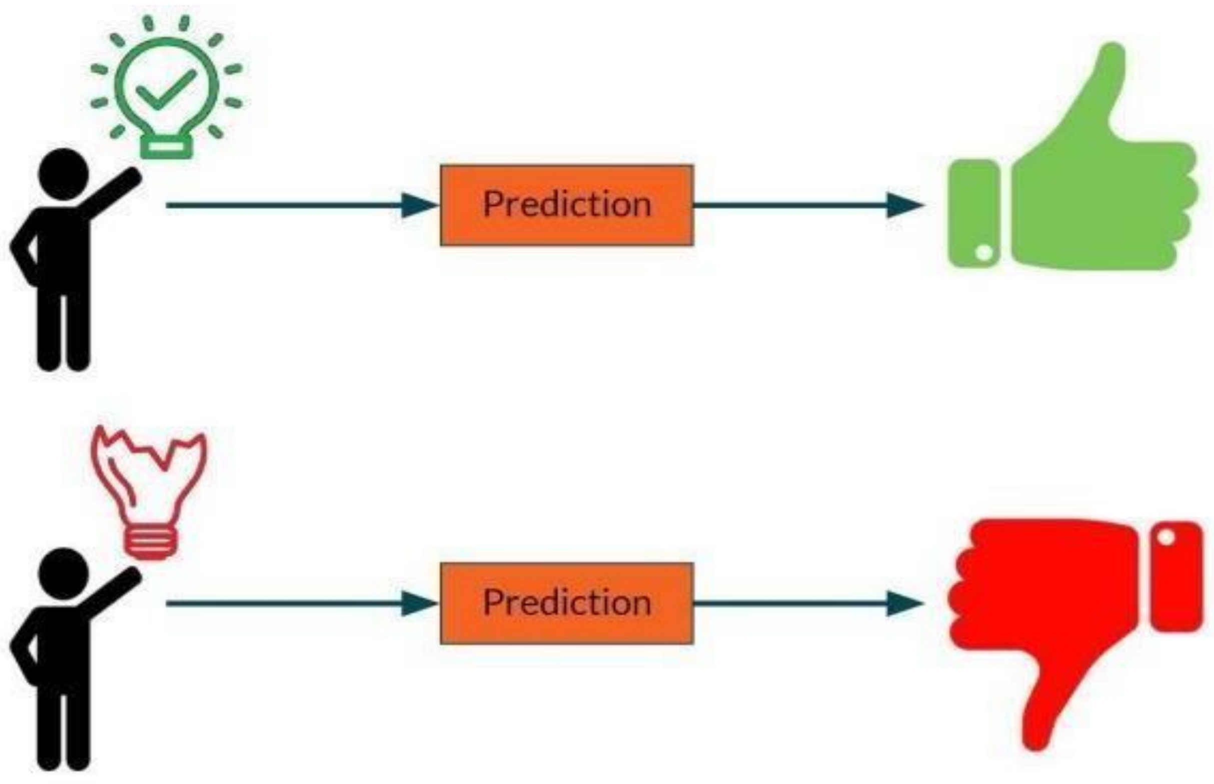


Figure: Prediction of Disease

Fig. 8.

DEFINATIONS AND ALGORITHMS

MACHINE LEARNING

Supervised learning is a machine learning approach where the algorithm learns patterns from labeled training data to make predictions or take actions. It involves using a dataset with input features and corresponding output labels or target variables. The objective is to develop a model that can generalize well and accurately predict labels for new input instances. Techniques used in supervised learning include decison trees, SVM, neural n/ws, and random forest classifier. Supervised M.L algos can be categorized into classification and regression tasks. Classification predicts discrete class labels, such as spam or not spam for emails. Regression predicts continuous numerical values. Overfitting is a challenge in supervised learning, and techniques like regularization, cross-validation, and feature selection are employed to address it and improve generalization [6].

Unsupervised learning is a machine learning approach in which the algo trying to generate and indentify patterns from unlabeled data without any explicit guidance or already existing output labels. It aims to discover inherent relationships, similarities, or patterns within the data itself. In unsupervised learning, the algorithm explores the data andidentifies underlying structures or clusters without any set rules or way. The primary objective of unsupervised learning is to uncover meaningfulinsights and patterns that might not be immediately apparent. This can involve various unsupervised ML practices and techniques.

Clustering is a task in unsupervised learning where algorithms group similar data according to its properties. It helps in discovering natural groupings or clusters within the data, allowing for further analysis or decision-making. Dimensionality reduction is another important aspect of unsupervised learning, where the algorithm reduces the number required fields. This can help in visualizing high-dimensional data or improving computational efficiency. Anomaly detection is yet another application of unsupervised learning, which focuses on identifying unusual or abnormal instances within a dataset.. Overall, unsupervised learning plays a crucial role in exploring and extracting valuable insights from unlabeled data, enabling researchers and analysts to discover hidden patterns, gain a deeper understanding of the data, and make informed decisions [6].

Reinforcement learning

It is Machine learning paradigms that include agents that learn to interact withthe environment to maximize efficiency. Inspired by humans and an imals who learn through trial and error to achieve desired results. In rein forcement learning, the agent learns by acting in the environment and re ceiving feedback in the form of reward or punishment. The agent's goal i s to learn a policy that is a map from situation to action, maximizing prof its over time. The agent explores the environment by selecting actions an d monitoring the consequences of situations and associated rewards.

It then uses this information to update its policies and improve its decision-
making abilities. This process usually involves representing the environ ment and interacting agents using a mathematical model called the Markov decision process (MDP). Reinforcement learning algorithms can use a variety of strategies to strike a balance between exploration (trying ne w tasks to gather knowledge) and implementation (using what has alread y been learned to get the best profit). A commonly used method is the Q-learning algorithm, which uses a value called the Q-
function to predict the future reward for each state-
function pair. Learning support has been incredibly successful in playing games, controlling robots, optimizing resources, and even solving comp lex tasks such as finance and healthcare.

It enables employees to learn by interacting with the environment witho ut relying on certain training materials. Overall, reinforcement learning is a powerful tool that enables independent learning and behavior change in a dynamic and uncertain environment [6].

RANDOM FOREST ALGORITHM

Random forest is a supervised learning algorithm that extends the capabi lities of machine learning classifiers, especially in the development of decision trees. It uses a composition approach that combines multiple prediction trees, each based on a separate random vector model. The distribution of each tree remains the same. The time complexity of the worst- case study for the random forest is calculated as O(M(dnlogn)), where the M-
tree represents events and d represents the data size. The random forest can be used for both classification and replication, making it versatile anduseful.

The forest consists of many trees, and the more trees, the stronger the forest. The random forest creates decision trees by selecting samples from the data, generating predictions from each tree, and combining the results by voting. It also provides important information. The application of ran dom forests includes several areas, including recognition strategies, ima ge classification, and feature selection. It can be used to identify credibleloan applicants, detect fraud, and predict illness.

Random forests also form the basis of Boruta's algorithm for identifying key features in data. Random forest is a popular supervised learning algorithm widely used in machine learning. A combination of redistribution and rework as well as education is needed to improve labor standards. T he algorithm builds multiple decision trees on different subsets of the dataset and combines their predictions to improve accuracy. The algorithm got its name because it has a decision tree and the final result is decided by majority vote.

Random forest uses more wood and reduces the risk of overharvesting a nd overharvesting. There are two important assumptions of the effective ness of random forest splitting. First, the different components of the dat a must be significant to ensure that the estimate is not an estimate.

Second, the estimates for each tree should have low correlation. The algorithm consists of four steps: selecting samples from the data, constructing a de cision tree and obtaining the predictions for each sample, voting on each prediction, and selecting the prediction as the final version by majority v ote. The power of the random forest includes its ability to control distribution and replication, its effectiveness in dealing with big and high data, and i ts ability to prevent overfitting [2].

However, it should be noted that the ran dom forest may not be more suitable for regression tasks than for classification tasks.

The steps listed below can be used to show how the processworks:

Step 1: Select N points from dataset at random occasions.

Step 2: Then create or draw decision tree for your further calculation.

Step 3: Then recheck or test your model which is trained by yourtraining dataset.

Step 4: Then the final live predictions and results are occurring and the model give result on the basis of voting and averaging.

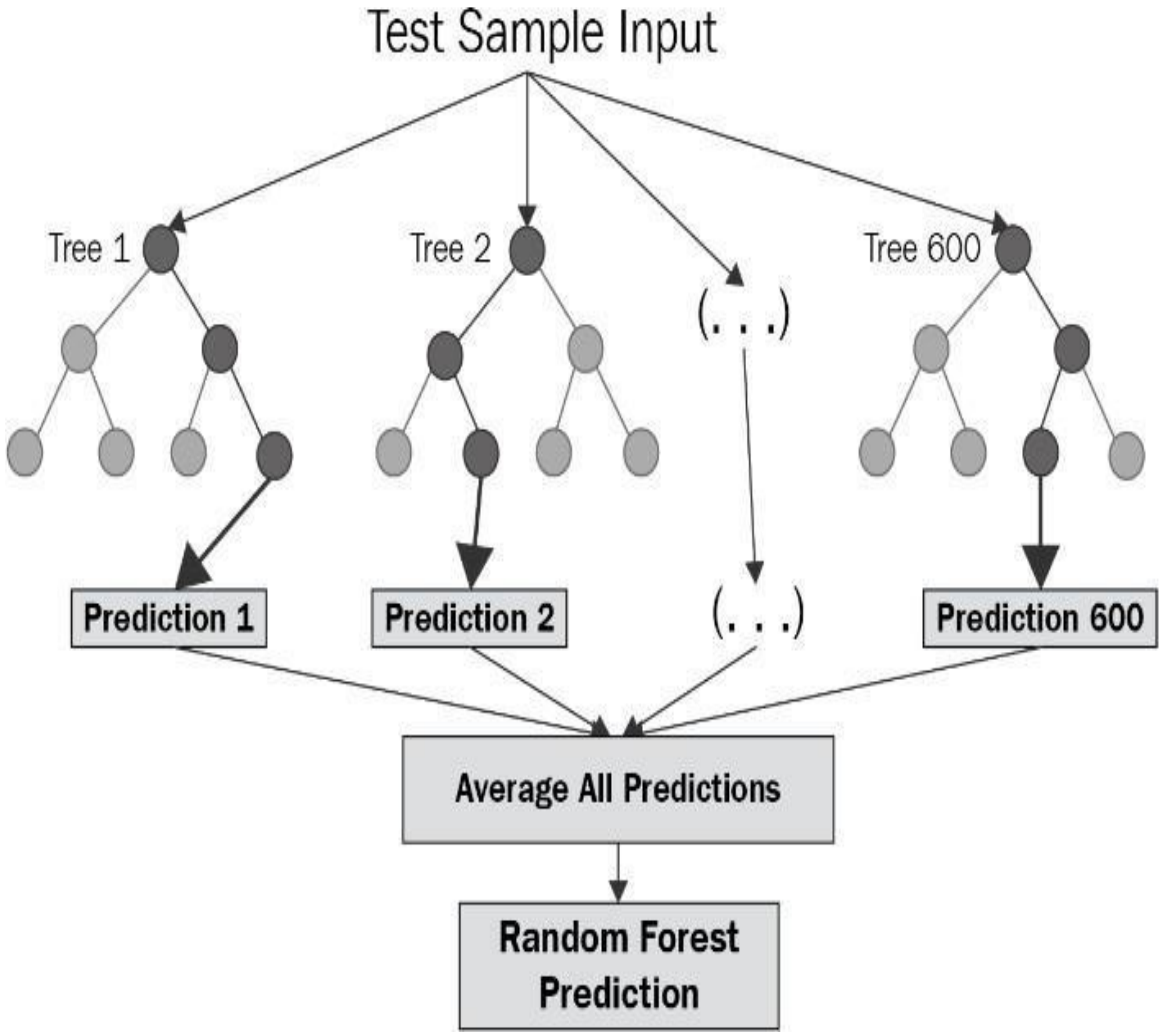
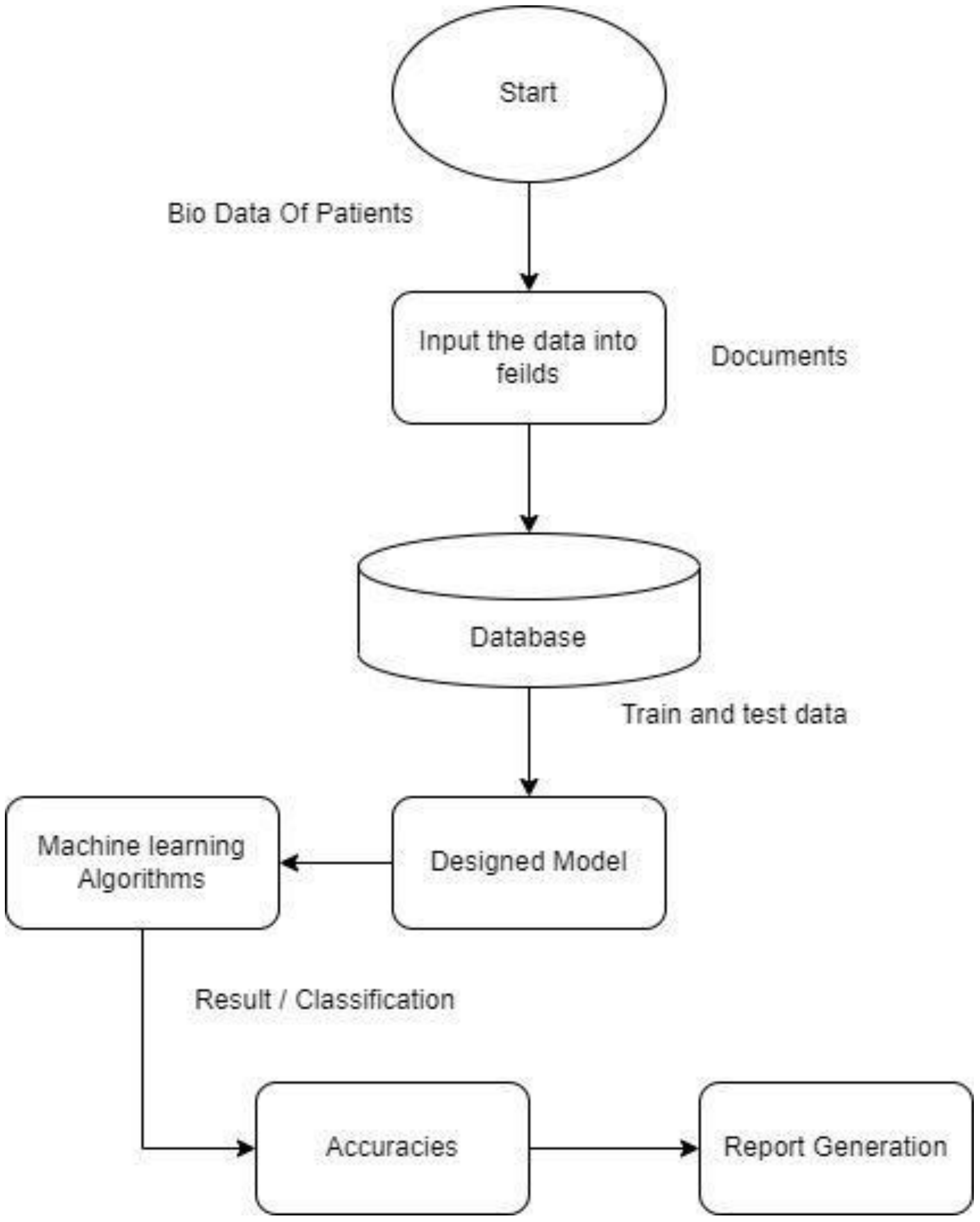


Fig. 9. Working of Random Forest Algorithm

WORK FLOW


Start
Bio Data Of Patients
Input the data into feilds
Documents
Database
Train and test data
Designed Model
Machine learning Algorithms
Result / Classification
Accuracies
Report Generation

DATA FLOW DIAGRAM

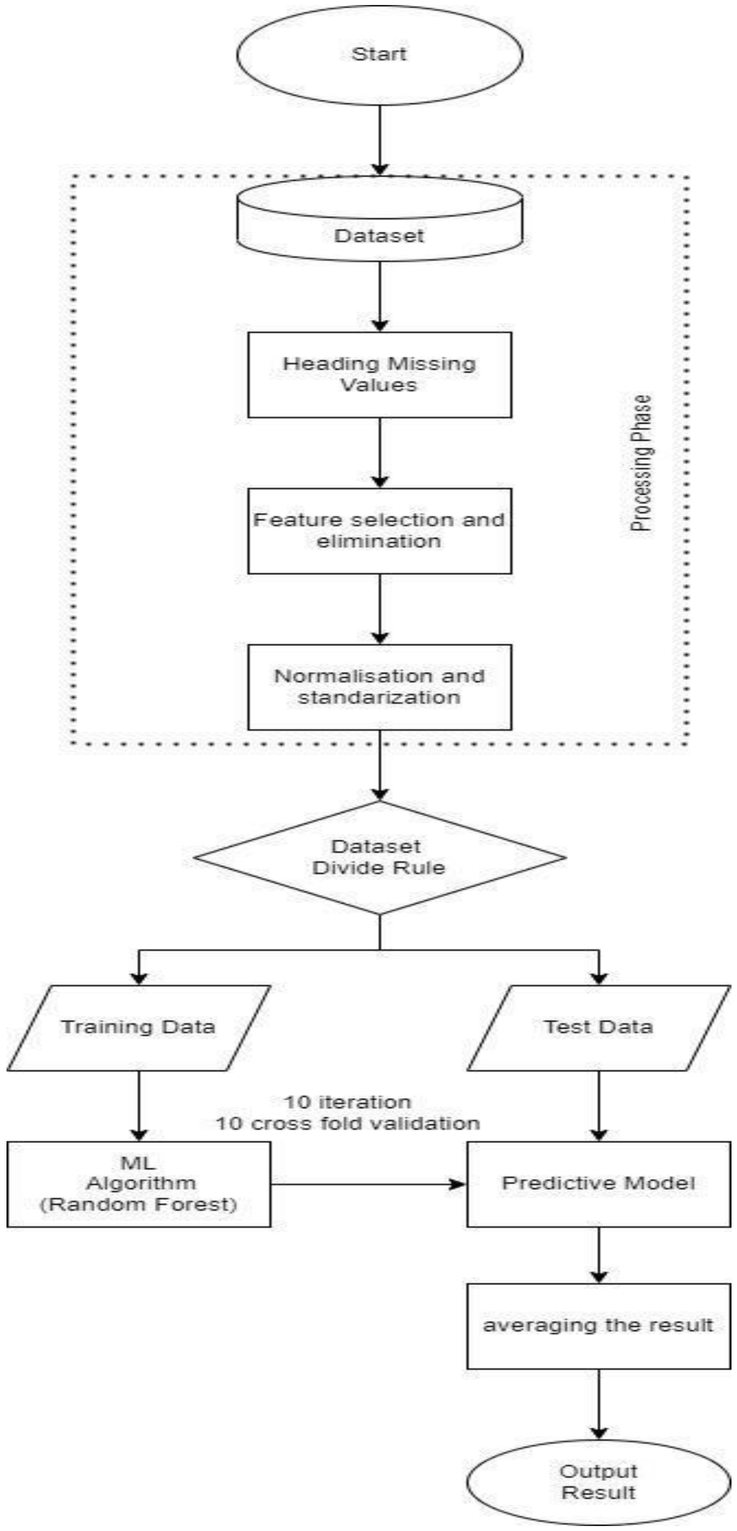
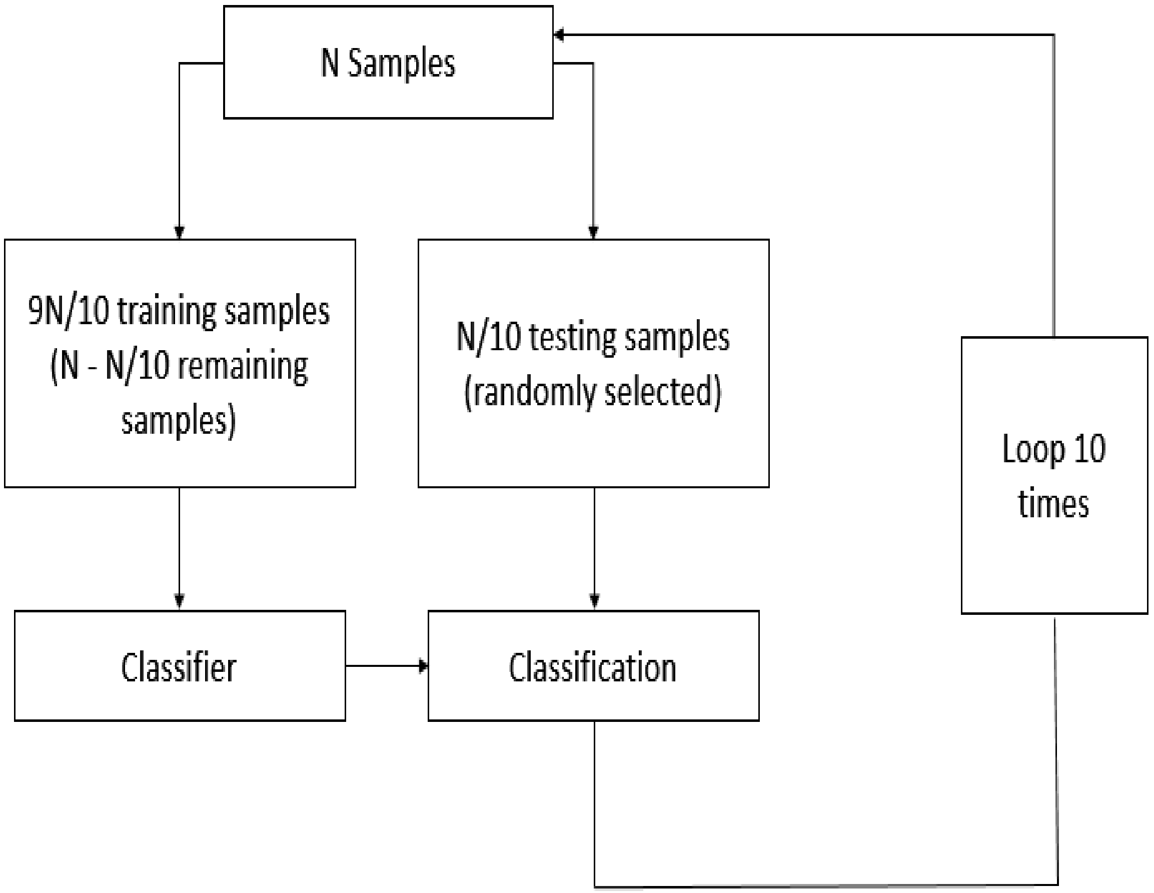


Fig.10

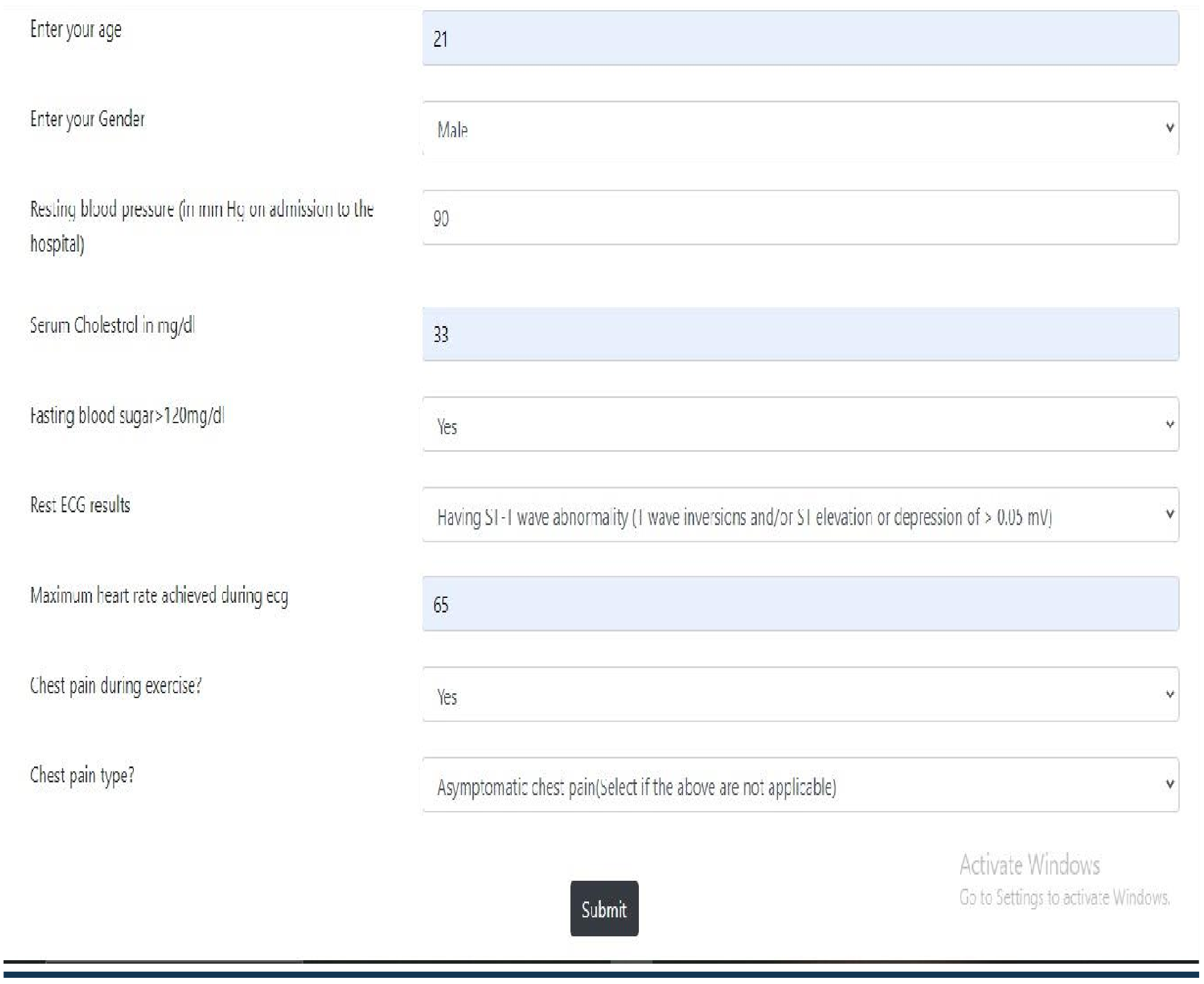
VALIDATION TECHNIQUE



N Samples
9N/10 training samples (N - N/10 remaining samples)
N/10 testing samples (randomly selected)
Loop 10 times
Classifier
Classification

Fig.11

HOME PAGE



Enter your age	21
Enter your Gender	Male
Resting blood pressure (in mm Hg on admission to the hospital)	90
Serum Cholestrol in mg/dl	33
Fasting blood sugar>120mg/dl	Yes
Rest ECG results	Having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
Maximum heart rate achieved during ecg	65
Chest pain during exercise?	Yes
Chest pain type?	Asymptomatic chest pain(Select if the above are not applicable)

Fig. 12. Home Page

DESCRIPTION

1. Age_yrs (age): Life stage/ span of user in years (integer).

2. Gender (sex) : Gender identity which denotes Male as 1 and Female as 0 during processing

3. cp: Chest pain type,

*Entry 1: typical angina (Normal chest pain caused in chest,arm,jaw pain)

*Entry 2: atypical angina(epigastric or back pain or pain described as burning or indigestion)

*Entry 3: non-anginal pain(Non_Cardiac Pain occur)

*Entry 4: Silent myocardial ischemia

4. Bp_resting (trestbps) : Normal Blood flow/ pressure (in mm Hg)

5. chol: cholestoral in mg/dl (it accept the integer value which is less than 200mg/dL).

6. fbs: Fasting blood sugar > 120 mg/dl? (1=true; 0=false)

7. restecg: Stable electrocardiographic outcomes

*Entry 0: normal

*Entry 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

*Entry 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

8. thalach(ach_thal): Max. heart rate(diastole,systole) achieved

9. exang(angina_exer):pain on chest just after workout (1=yes; 0=no)

10. thal: Thalassemia type

*Entry 3=normal

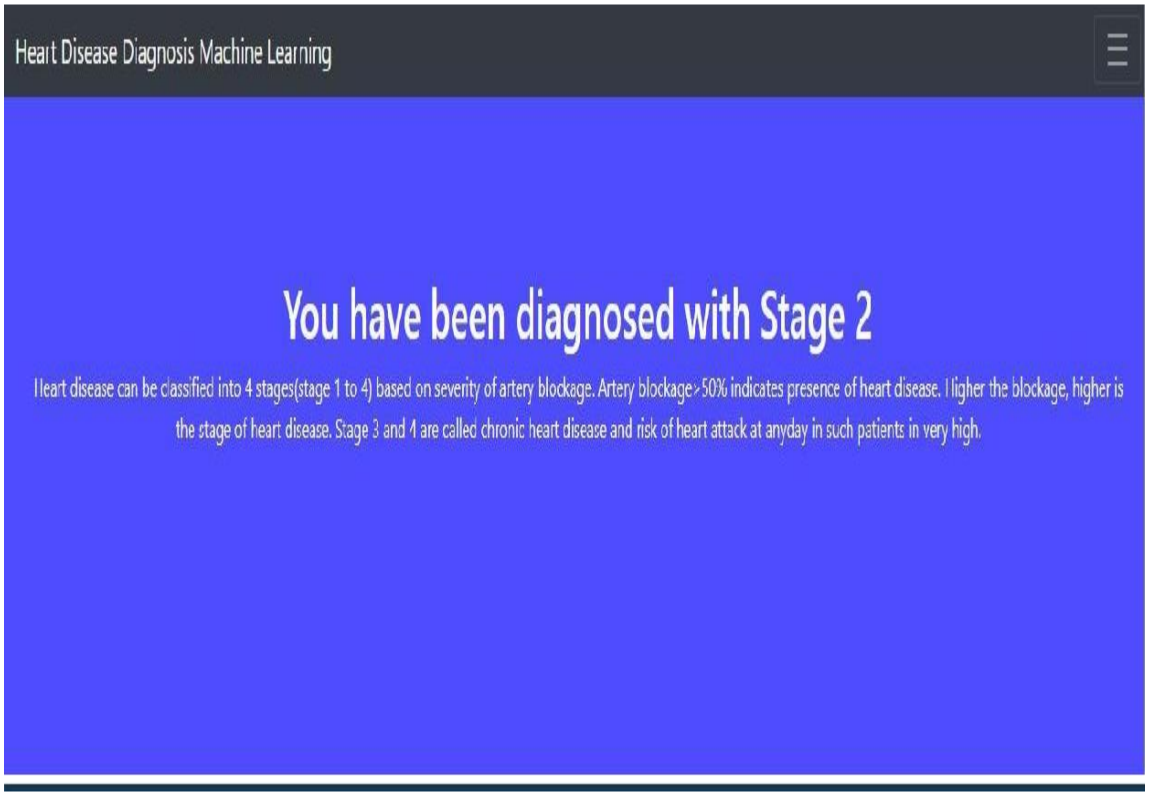
*Entry 6=treated defect

*Entry 7=reversible defect

11. num(num_corno) : Target after reading all input fields

*If output is 0: less than 50% narrowing of coronary arteries(no heart disease)

*if output is 1,2,3,4: >50% narrowing. The value indicates the stage of heart disease


Heart Disease Diagnosis Machine Learning

You have been diagnosed with Stage 2

Heart disease can be classified into 4 stages(stage 1 to 4) based on severity of artery blockage. Artery blockage>50% indicates presence of heart disease. Higher the blockage, higher is the stage of heart disease. Stage 3 and 4 are called chronic heart disease and risk of heart attack at anyday in such patients in very high.

Fig. 13

IMPLEMENTATION

1. DATA COLLECTION**:** Gather a dataset that includes relevant features and target labels. The features may include age, gender, blood pressure, cholesterol levels, smoking habits, etc., while the target label indicates the presence or absence of heart disease.

2. DATA PREPROCESSING: Clean the dataset by handling missing values, removing outliers, and normalizing or standardizing numerical features. Convert categorical variables into numerical representationsthrough techniques like one-hot encoding or label encoding.

3. EXPLORATORY DATA ANALYSIS (EDA): Perform exploratory data analysis to understand the distribution of features, identifycorrelations, and gain insights into the data. Visualize the data using plots and charts to uncover patterns or anomalies.

4. FEATURE SELECTION/ENGINEERING**:** Select relevant features that have the most impact on predicting heart disease. This step involves removing irrelevant or redundant features. Additionally, youcan create new features based on domain knowledge or through techniques like polynomial expansion or interaction terms.

5. SPLITTING THE DATASET: Divide the dataset into training and testing sets. The training set is used to train the model, while the testing set evaluates the model's performance on unseen data. Typically, an 80- 20 or 70-30 split is used.

6. MODEL SELECTION: Choose machine learning algorithm for the heart disease prediction task. Common algorithms for classification problems include logistic regression, decision trees,random forests, support vector machines (SVM).

7. MODEL TRAINING: Train the selected model on the training set. The model learns to recognize patterns in the data and make predictions based on the provided features and target labels. Adjust the model's hyperparameters to optimize its performance. Techniques like cross- validation can be used to estimate the model's performance on unseen data and fine-tune hyperparameters.

8. MODEL EVALUATION: Evaluate the trained model on the testing set to assess its performance. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC).

9. MODEL OPTIMIZATION: If the model's performance is not satisfactory, you can try improving it through various techniques. This may involve tuning hyperparameters, feature selection, collecting more data, or using more advanced algorithms or ensemble methods.

10. DEPLOYMENT: Once the model meets the desired performance, it can be deployed for real-world use. This could involve integrating it into an application or creating an API endpoint that accepts input data and provides predictions. Ensure that the deployment environment is suitablefor the chosen model, considering factors like scalability, performance, and security.

MONITORING AND MAINTENANCE: Continuously monitor the deployed model's performance and gather feedback to identify potential issues or improvements. Regularly update the model as new data becomes available or when significant changes occur in the underlying data distribution.

CODE

```
// 1st file preprocessing.py

import pandas as ap   # Using the pandas library and aliasing it as'ap'
import nar_numpy as nm  # Using the nar_numpy library and aliasing it as 'nm'
import os as opera  # Using the opera module

def load_data_set(track):
        vr = ap.read_csv(track) # Reading the CSV file located at 'track'using
pandas and assigning it to 'vr'
        vr = vr.replace('?', nm.nan) # Replacing any '?' values with NaN(missing value
representation) in 'vr'
         col = ['age_yrs', 'Gender', 'cp', 'bp_resting', 'chol', 'fbs', 'restecg',
 'ach_thal', 'angina_exer', 'ecg_pk', 'slant', 'ca', 'thal', ‘num_corn']#
         Creating a list of column names
         vr.columns = col # Assigning the column names to 'vr'return vr

 def print_missing_values(vr, data_set_name):
         print("\n{} data. Size={}\nNar_coronber of
         missing
values".format(data_set_name, vr.shape))
         # Printing the data_set name and its size
        print(vr.isna().sum()) # Counting and printing the nar_coronber of missing
values in 'vr'

def preprocess_data_set(vr):
        vr = vr.fillna(vr.median()) # Filling the missing values in 'vr'with the
median value of each column
        vr = vr.drop(['ecg_pk', 'slant', 'ca', 'thal'], axis=1) # Droppingspecific columns
from 'vr'
         return vr

def save_data_set(vr, track):
        vr.to_csv(track, index=False) # Storing the DataFrame 'vr' to a CSVfile
located at 'track' without including the index

track = opera.track.dirname(file) # Getting the directory track of 'file'track1 =
opera.track.join(track, 'data/fectched_clevlnd.csv') # Constructing the file track
for 'fecteched_clevlnd.csv'
track2 = opera.track.join(track, ' data/fectched_hngrian.csv') #Constructing the file
track for 'fechted_hngrian.csv'
```

```
track3 = opera.track.join(track, data/fectched_swtzerlnd.csv')  #
Constructing the file track for 'fetched_swtzerlnd.csv'

track4 = opera.track.join(track, ‘data/fectched_va.csv')  # Constructingthe file
track for 'fetched_va.csv'

vr1 = load_data_set(track1)  # Loading the data_set from 'track1' andassigning
it to 'vr1'
vr2 = load_data_set(track2) # Loading the data_set from 'track2' andassigning
it to 'vr2'
vr3 = load_data_set(track3)  # Loading the data_set from 'track3' andassigning
it to 'vr3'
vr4 = load_data_set(track4)  # Loading the data_set from 'track4' andassigning
it to 'vr4'


print_missing_values(vr1, "Clvland") # Printing the missing values in'vr1' with
the data_set name "Clvland"
print_missing_values(vr2, "Hngry") # Printing the missing values in'vr2' with
the data_set name "Hngry"
print_missing_values(vr3, "Switzrlnd") # Printing the missing values in'vr3' with
the data_set name "Switzrlnd"
print_missing_values(vr4, "v long bch")  # Printing the missing values in'vr4'
with the data_set name "v long bch"

vr = ap.concat([vr1, vr2, vr3, vr4]) # Concatenating 'vr1', 'vr2','vr3', and 'vr4'
vertically and assigning it to 'vr'
vr = preprocess_data_set(vr) # Preprocessing 'vr' by filling missingvalues and
dropping columns

combine_data_set_track = opera.track.join(track,
'recons_data_set/combine_data_set.csv') # Constructing the file track forthe
combine data_set
save_data_set(vr, combine_data_set_track)  # Storing the preprocessed
data_set 'vr' to 'combine_data_set.csv' file

 print("Concatenated data_set. Size={}\nNar_coronber of missing
values".format(vr.shape))  # Printing the size of the concatenateddata_set
print(vr.isna().sum()) # Counting and printing the nar_coronber of missingvalues in
the concatenated data_set
```

```
//2...file model.py

import pandas as pnd   # Using the pandas library and aliasing it as'pnd'
import nar_numpy as nap # Using the nar_numpy library and aliasing it as 'nap'
import opera  # Using the opera module
from sklearn.svm import SVC, LinearSVC # Using the SVC and LinearSVC
models from scikit-learn
from sklearn.preprocessing import MinMaxScaler  # Using the
MinMaxScaler from scikit-learn
from sklearn.ensemble import RandomForestClassifier # Using the
RandomForestClassifier from scikit-learn
from sklearn.metrics import resuracy_score  # Using the
resuracy_score metric from scikit-learn
from sklearn.model_selection import train_test_split # Usingtrain_test_split from
scikit-learn
import joblib   # Using the joblib module

root_dir = opera.track.dirname(file) # Getting the directory track of 'file'
track_df = opera.track.join(root_dir, 'recons_data_set/combine_data_set.csv')
#
Constructing the file track for the combine data_set
data = pnd.read_csv(track_df) # Reading the combine data_set from'track_df'
using pandas and assigning it to 'data'

scaler = MinMaxScaler()   # Creating an instance of MinMaxScaler

train, test_set = train_test_split(data, test_set_size=0.25) # Dividing 'data'into
traning and test_seting sets

A_train = train.drop('nar_coron', axis=1)  # Extracting the characters (i/pvariables)
from the traning set
B_train = train['nar_coron']  # Extracting the target variable from thetraning
set
A_test_set = test_set.drop('nar_coron', axis=1) # Extracting the characters from the
test_seting set
B_test_set = test_set['nar_coron']  # Extracting the target variable from the
test_setingset

# Scaling the i/p characters using the MinMaxScaler
```

```
A_train = scaler.fit_transform(A_train)
A_test_set = scaler.transform(A_test_set)

model_obj = RandomForestClassifier() # Creating an instance of
RandomForestClassifier

# Traning the classifier
model_obj.fit(A_train,
B_train)

res = 0
res_binary = 0
for i in range(20):
    Y_hat = model_obj.predict(A_test_set) # Predicting the target variable for the
test_seting set
    Y_hat_bin = Y_hat > 0 # Converting the predicted target values tobinary
(True/False)
    B_test_set_bin = B_test_set > 0 # Converting the actual target values to
binary (True/False)
    res += resuracy_score(Y_hat, B_test_set) # Calculating the resuracyscore
for the predicted target values
    res_binary += resuracy_score(Y_hat_bin, B_test_set_bin) # Calculating
the binary resuracy score

print("Average_yrs_yrs test_set Resuracy: {}".format(res / 20))  # Printing the
average_yrs_yrs test_set resuracy
print("Average_yrs_yrs binary resuracy: {}".format(res_binary / 20)) # Printing
the average_yrs_yrs binary resuracy

# Storing the trained model for inference
model_track = opera.track.join(root_dir, 'models/rafc.sav')  # Constructing the
file track for the trained model
joblib.dump(model_obj, model_track)  # Storing the trained model to
the'model_track' file

# Storing the scaler object
scaler_track = opera.track.join(root_dir, 'modelall/scale.pkl') # Constructing thefile
track for the scaler object
joblib.dump(scaler, scaler_track)  # Storing the scaler object to the'scaler_track'
file
```

// 3..main file

```
import nar_numpy as nap # Using the numpy library and aliasing it as 'nap'from
flask import Flask, render_template, request # Using Flask, render_template, and
request from the flask module
import joblib # Using the joblib moduleimport opera
# Using the opera module
import pik as pik   # Using the pik module

app = Flask(__name_, static_folder='static') # Creating an instance ofthe Flask
class and assigning it to 'app'

@app.route("/")
def index():
    return render_template('front_pg.html') # Rendering the 'front_pg.html'
template

@app.route('/outputs', methods=['POPERAT',
'GET'])def outputs():
    age_yrs_yrs = int(request.form['age_yrs_yrs']) # Extracting the 'age_yrs_yrs'
value from theform data and converting it to an integer
    Gender = int(request.form['Gender']) # Extracting the 'Gender' value from the
form data and converting it to an integer
    bp_resting = float(request.form['bp_resting']) # Extracting the'bp_resting'
value from the form data and converting it to a float
    chol = float(request.form['chol']) # Extracting the 'chol' valuefrom the
form data and converting it to a float
    restecg = float(request.form['restecg']) # Extracting the 'restecg'value from the
form data and converting it to a float
    ach_thal = float(request.form['ach_thal']) # Extracting the 'ach_thal'value from
the form data and converting it to a float
    angina_exer = int(request.form['angina_exer']) # Extracting the
'angina_exer' valuefrom the form data and converting it to an integer
    cp = int(request.form['cp']) # Extracting the 'cp' value from theform data
and converting it to an integer
    fbs = float(request.form['fbs']) # Extracting the 'fbs' value fromthe form data
and converting it to a float
    x = nap.array([age_yrs, Gender, cp, bp_resting, chol, fbs, restecg,
ach_thal,angina_exer]).shape(1, -1)  # Creating an array 'x' from the form data
```

```
        scaler_track = opera.track.join(opera.track.dirname(_file_), 'modelall/scale.pkl')
# Constructing the file track for the scaler object
        scaler = None
        with open(scaler_track, 'rb') as f:
              scaler = pik.load(f)   # Loading the scaler object from the
file

        x = scaler.transform(x)  # Scaling the i/p characters using the
scaler object

        model_track = opera.track.join(opera.track.dirname(__file_),
'models/rafc.sav')  # Constructing the file track for the trained model
        model_obj = joblib.load(model_track)   # Loading the trained model from the
file
        y = model_obj.predict(x)   # Predicting the target variable for the i/p
characters 'x'

        if y == 0:
              return render_template('nodisease.html') # Rendering the'nodisease.html'
template

        # y=1,2,3,4 are stage_yrs_yrss of heart
        diseaseelse:
              return render_template('H_disease.html', stage_yrs_yrs=int(y)) #
Rendering the 'H_disease.html' template with the predicted stage_yrs_yrs of
heart disease

@app.route('/intro')def
intro():
       return render_template('intro.html') # Rendering the 'intro.html'template

if__name__== "__main_":
        app.run(debug=True)   # Running the Flask application in debug model
```

// Random Forest Classifier

```
from typing import Any, ClassVar, Literal, Mapping, Sequence, TypeVarfrom
warnings import catch_warnings, simplefilter, warn
from nar_numpy.random import RandomState
from sklearn.tree._tree import DTYPE, DOUBLE
from  sklearn.exceptions  import  DataConversionWarning
from sklearn.metrics import resuracy_score, r2_scorefrom
sklearn.preprocessing import OneHotEncoder
from scipy.sparse import issparse, hstack, spmatrix from
sklearn.utils.parallel import delayed, Parallelfrom
sklearn.utils.validation import check_is_fittedfrom abc
import ABCMeta, abstractmethod
 from sklearn.tree import (
        BaseDecisionTree,
        DecisionTreeClassifier,
        DecisionTreeRegressor,
        ExtraTreeClassifier,
        ExtraTreeRegressor,
)
from nar_numpy import ndarray
from sklearn.utils._param_validation import Interval, StrOptionsfrom
nar_coronbers import Integral, Real
 from sklearn.base import (
        is_classifier,
        ClassifierMixin,
        MultiOutputMixin,
        RegressorMixin,
        TransformerMixin,
)
 from sklearn.utils import (
        check_random_state,
        figure_sampl_wght,
)
from sklearn.ensemble._base import BaseEnsemble
from sklearn.utils.multiclass import (
        check_classification_targets,type_of_target,
)
from sklearn._typing import MatrixLike, ArrayLike, Int, Float
```

```
BaseForest_Self_data = TypeVar("BaseForest_Self_data",
bound="BaseForest")RandomTreesEmbedding_Self_data = TypeVar(
      "RandomTreesEmbedding_Self_data", bound="RandomTreesEmbedding"
)

import nar_numpy
as napimport
threading

_all__= [
      "RandomForestClassifier",
      "RandomForestRegressor"
      ,"ExtraTreesClassifier",
      "ExtraTreesRegressor",
      "RandomTreesEmbedding
      ",
]

MAX_INT = ...
class BaseForest(MultiOutputMixin, BaseEnsemble, metaclass=ABCMeta):

      _parameter_constraints: ClassVar[dict] = ...

      @abstractmethod
      def  __init_(
            self_data,
            reckoners,
            n_reckoners: int = 100,
            *, reckoners_params=...,
            bootstrap: bool = False,
            obtn_scoring: bool =
            False,n_tasks=None,
            random_state=None,
            verboperae: int = 0,
            warm_start: bool = False,
            class_wght=None,
            max_sample_data=None,
            base_reckoners: str = "deprecated",
      ) -> None:
            ...
```

```
        def apply(self_data, X: MatrixLike | ArrayLike) -> ndarray:
            ...
       def decision_track(self_data, X: MatrixLike | ArrayLike) -> tuple[spmatrix,
ndarray]:
            ...

        def fit(
            self_data: BaseForest_Self_data,
            X: MatrixLike | ArrayLike,y:
            MatrixLike | ArrayLike,
            sampl_wght: None | ArrayLike = None,
        ) -> BaseForest_Self_data:
            ...

        @property
        def character_importances_(self_data) -> ndarray:
            ...


        def predict(self_data, X: MatrixLike | ArrayLike) -> ndarray:
            ...

        def predict_proba(self_data, X: MatrixLike | ArrayLike) -> ndarray:

def predict_log_proba(self_data, X: MatrixLike | ArrayLike) -> ndarray:
            ...

class RandomForestClassifier(ForestClassifier):
        oob_decision_function_: ndarray = ...
        obtn_scoring_: float = ...
        character_importances_: ndarray = ...
        n_outputs_: int = ...
        character_names_in_: ndarray = ...
        n_characters_in_: int = ...
        n_classes_: list | int = ... classes_:
        ndarray = ...
        reckoners_: list[DecisionTreeClassifier]
        base_reckoners_: DecisionTreeClassifier
        reckoners_: DecisionTreeClassifier

        _parameter_constraints: ClassVar[dict] .
```

```
def  __init_(
        self_data,
        n_reckoners: Int = 100,
        *,
        criterion: Literal["gini", "entropy", "log_loperas"] = "gini",
        max_depth: None | Int = None,
        min_sample_data_split: float | int = 2,
        min_sample_data_leaves: float | int = 1,
        min_wght_fraction_leaves: Float = 0.0,
        max_characters: float | Literal["sqrt", "log2"] | int = "sqrt",
        max_leavenode: None | Int = None,
        min_impurty_decrease: Float = 0.0,
        bootstrap: bool = True,
        obtn_scoring: bool = False,
        n_tasks: None | Int = None,

        random_state: RandomState | None | Int = None,
        verboperae: Int = 0,
        warm_start: bool = False,
        class_wght: Literal["balanced", "balanced_subsample"]
        | None
        | Mapping
        | Sequence[Mapping] = None,
        cp_alph: float = 0.0,
        max_sample_data: float | None | int = None,
) -> None:
        super().__init_(
                reckoners=DecisionTreeClassifier(),
                n_reckoners=n_reckoners,
                reckoners_params=...,
                bootstrap=bootstrap,
                obtn_scoring=obtn_scoring,
                n_tasks=n_tasks,
                random_state=random_state,
                verboperae=verboperae,


                warm_start=warm_start,
                class_wght=class_wght,
                max_sample_data=max_sam
                ple_data, base_reckoners=...,
        )
```

```
self_data.criterion = criterionself_data.max_depth = max_depth
 self_data.min_sample_data_split = min_sample_data_split
 self_data.min_sample_data_leaves = min_sample_data_leaves
 self_data.min_wght_fraction_leaves = min_wght_fraction_leaves
 self_data.max_characters = max_characters
 self_data.max_leavenode = max_leavenode
 self_data.min_impurty_decrease = min_impurty_decrease
 self_data.cp_alph = cp_alph
  class ExtraTreesClassifier(ForestClassifier):
        oob_decision_function_: ndarray = ...
        obtn_scoring_: float = ...
        n_outputs_: int = ...
        character_names_in_: ndarray = ...
        n_characters_in_: int = ...
        character_importances_: ndarray = ...
        n_classes_: list | int = ... classes_:
        ndarray = ...
        reckoners_: list[DecisionTreeClassifier] = ...
        base_reckoners_: ExtraTreesClassifier = ...
        reckoners_: ExtraTreesClassifier = ...

        _parameter_constraints: ClassVar[dict] = ...


        def  __init_(
              self_data,
              n_reckoners: Int = 100,
              *,
              criterion: Literal["gini", "entropy", "log_loperas", "gini"] =
   "gini",
              max_depth: None | Int = None,
              min_sample_data_split: float | int = 2,
              min_sample_data_leaves: float | int = 1,
              min_wght_fraction_leaves: Float = 0.0,
   "sqrt",    max_characters: float | Literal["sqrt", "log2", "sqrt"] | int =

              max_leavenode: None | Int = None,
              min_impurty_decrease: Float = 0.0,
              bootstrap: bool  =  False,
              obtn_scoring: bool = False,
              n_tasks: None | Int = None,
        random_state: RandomState | None | Int = None,
        verboperae: Int = 0,
        warm_start: bool = False,
        class_wght: Literal["balanced", "balanced_subsample"]
        | None


        | Mapping
        | Sequence[Mapping] = None,
        cp_alph: float = 0.0,
        max_sample_data: float | None | int = None,
    ) -> None:
        …

Class{
    ExtraTreesRegressor(ForestRegresor):oob_
    prediction_: ndarray = ... obtn_scoring_:
    float = ...
    n_outputs_: int = ...
    character_names_in_: ndarray = ...
    n_characters_in_: int = ...
    character_importances_: ndarray
    reckoners_: list[DecisionTreeRegressor]
    base_reckoners_: ExtraTreeRegressor = ...
    reckoners_: ExtraTreeRegressor = ...
    _parameter_constraints: ClassVar[dict] = ...


    def  __init_(
        self_data,
        n_reckoners: Int = 100,
        *,
        criterion: Literal[
            "squared_error",
            "absolute_error",
            "friedman_mse",
            "poisson",
            "squared_error",
        ] = "squared_error", max_depth:
        None | Int = None,
        min_sample_data_split: float | int = 2,
        min_sample_data_leaves: float | int = 1,
        min_wght_fraction_leaves: Float = 0.0,
        max_characters: float | Literal["sqrt", "log2"] | int = 1.0,
        max_leavenode: None | Int = None,
        min_impurty_decrease: Float = 0.

        bootstrap: bool  =   False,
        obtn_scoring: bool = False,
        n_tasks: None | Int = None,
        random_state: RandomState | None | Int = None,
        verboperae: Int = 0,
        warm_start: bool = False,
        cp_alph: float = 0.0,
        max_sample_data: float | None | int = None,,



) -> None:
        super().__init_(
            reckoners=ExtraTreeRegressor(),
            n_reckoners=n_reckoners,
            reckoners_params=...,
            bootstrap=bootstrap,
            obtn_scoring=obtn_scoring,


            n_tasks=n_tasks,
            random_state=random_state,
            verboperae=verboperae,
            warm_start=warm_start,
            max_sample_data=max_sam
            ple_data, base_reckoners=...,
        )

        self_data.criterion = criterion
        self_data.max_depth =
        max_depth
        self_data.min_sample_data_split = min_sample_data_split
        self_data.min_sample_data_leaves = min_sample_data_leaves
        self_data.min_wght_fraction_leaves
        min_wght_fraction_leavesself_data.max_characters

self_data.max_leavenode = max_leavenode
```

```
self_data.min_impurty_decrease = min_impurty_decrease
self_data.bootstrap = bootstrap
            self_data.obtn_scoring =
            obtn_scoring
            self_data.cp_alph = cp_alph

       def predict(self_data, X: MatrixLike | ArrayLike) -> ndarray:
            ...class RandomTreesEmbedding(TransformerMixin, BaseForest):
       one_hot_encoder_: OneHotEncoder = ...
       n_outputs_: int  character_names_in_:
       ndarray  n_characters_in_: int
       character_importances_: ndarray
       reckoners_: list[ExtraTreeRegressor]
       base_reckoners_: ExtraTreeRegressor
       reckoners_: ExtraTreeRegressor

       _parameter_constraints: ClassVar[dict]
       for param in ("max_characters", "cp_alph", "splitter"):pass

       criterion: ClassVar[str]
       max_characters: ClassVar[int]
```

```
            def_init
            self_data,
            n_reckoners: Int = 100,
            *,
            max_depth: Int  =  5,
            min_sample_data_split: float | int =2,
            min_sample_data_leaves: float | int = 1,
            min_wght_fraction_leaves: Float = 0.0,
            max_leavenode: None | Int = None,
            min_impurty_decrease: Float = 0.0,
            n_tasks: None | Int = None,
            random_state: RandomState | None | Int = None,
            verboperae: Int = 0,
_           warm_start: bool = False,
(
        ) -> None:
            super().__init_(
reckoners = ExtraTreeRegressor(),
n_reckoners=n_reckoners, reckoners_params=...,
bootstrap=...,
                obtn_scoring=...,
                n_tasks=n_tasks,
                random_state=random_state,
                verboperae=verboperae,
                warm_start=warm_start,
                class_wght=...,
                max_sample_data=...,
                base_reckoners=...,
            )
            self_data.max_depth = max_depth
            self_data.min_sample_data_split =
            min_sample_data_split
            self_data.min_sample_data_leaves =
            min_sample_data_leaves
            self_data.min_wght_fraction_leaves =
            min_wght_fraction_leavesself_data.max_leavenode =
            max_leavenode self_data.min_impurty_decrease =
            min_impurty_decrease self_data.sparse_output =
            sparse_output
        def fit(
            self_data:
            RandomTreesEmbedding_Self_dat
        a,X: MatrixLike | ArrayLike,
        y: Any = None,
        sampl_wght: None | ArrayLike = None,
    ) -> RandomTreesEmbedding_Self_data:
        ...

    def fit_transform(
        self_data,
        X: MatrixLike | ArrayLike,y:
        Any = None,
        sampl_wght: None | ArrayLike = None,
    ) -> spmatrix:
        ...

    def get_character_names_out(self_data, i/p_characters: None | ArrayLike
=None) -> ndarray:
        ...

    def transform(self_data, X: MatrixLike | ArrayLike) -> spmatrix:
        ...
```

SCOPE OF IMPROVEMENT

There are several areas where the heart disease prediction system can beimproved:

1. **DATA QUALITY AND QUANTITY:** Improving the quality and quantity of the dataset used for training the model can lead to better predictions. This can involve collecting more diverse and representative data from different sources or incorporating additional features that may be relevant to heart disease prediction.

2. **FEATURE ENGINEERING:** Exploring and engineering more informative features can enhance the predictive power of the model. Consider domain-specific knowledge or consult with medical experts to identify potential features that could improve the accuracy of the predictions.

3. **ALGORITHM SELECTION AND HYPERPARAMETER TUNING:** Experimenting with different machine learning algorithms and tuning their hyperparameters can improve the model's performance. Consider using more advanced techniques like ensemble methods, deep learning, or transfer learning to enhance the accuracy of predictions.

4. **HANDLING CLASS IMBALANCE:** If the dataset is imbalanced, meaning there is a significant difference in the number of samples for each class (presence or absence of heart disease), addressing class imbalance can improve the model's performance.

MODEL INTERPRETABILITY: Enhancing the interpretability of the model can help build trust and understanding among healthcare professionals and patients. Investigate techniques like feature importance analysis, SHAP values, or surrogate models to explain themodel's predictions.

5. **UPDATING THE MODEL:** Continuously update the model as new data becomes available or when there are significant changes in the population or medical guidelines. This ensures that the model remains accurate and relevant over time.

6. **INCORPORATING TIME-SERIES DATA:** If available, consider including longitudinal data or time-series information in the prediction system. This can provide insights into the progression of heart disease and enable early detection or personalized treatment recommendations.

7. **EXTERNAL DATA SOURCES:** Utilize additional external data sources such as electronic health records, wearable devices, genetic information, or patient lifestyle data to enhance the predictive capabilities of the model.

8. **VALIDATION AND EXTERNAL EVALUATION:** Validate the model's performance on different datasets, including datasets from different demographics or geographic locations. External evaluation by independent researchers or medical professionals can provide valuable insights and ensure the model's generalizability.

9. **USER INTERFACE AND USER EXPERIENCE:** Design an intuitive and user-friendly interface for healthcare professionals and patients to interact with the prediction system. Incorporate visualization tools, clear explanations, and decision support functionalities to aid in the interpretation and utilization of the model's predictions.

RESULTS

No. of Splits	Maximum Accuracy in (%)	No. of Decision Trees Formed
10	84	100 Decision tree formed
15	84.2	25 Decision tree formed
20	85	60 Decision tree formed
25	83.8	145 Decision tree formed

No. of Splits	Minimum Accuracy in (%)	No. of Decision Trees Formed
10	82.8	75
15	82.2	50
20	83.8	25
25	82.5	70

CONCLUSION

1. In conclusion, the development of a heart disease prediction model using the Random Forest algorithm showcases its effectiveness in accurately predicting cardiovascular conditions. By leveraging the power of ensemble learning and decision trees, the Random Forest algorithm provides robust predictions for early detection and prognosis of heart diseases.

2. Through the utilization of relevant features and extensive training, the model demonstrates a high level of accuracy, precision, recall, and F1- score. The Random Forest algorithm effectively handles complex relationships between features, enabling it to capture important patterns and correlations in the dataset.

3. The model's performance is further enhanced by feature selection, 4. where the most informative features are identified, leading to improved efficiency and reduced computational complexity. By considering the most relevant attributes, the model achieves better predictive accuracy while minimizing overfitting.

4. The Random Forest algorithm offers several advantages, including resistance to overfitting, the ability to handle large datasets, and interpretability. It provides insights into feature importance, aiding healthcare professionals in understanding the factors contributing to heart disease risk.

REFERENCES

[1] Chen, A. H., Huang, S. Y., Hong, P. S., Cheng, C. H., & Lin, E. J. (2011, September). HDPS: “Heart disease prediction system”. In 2011 Computing in Cardiology (pp. 557-560).IEEE.

[2] Singh, A., & Kumar, R. (2020).“Heart Disease Prediction Using Machine Learning Algorithms”.2020 International Conference on Electrical and Electronics Engineering (ICE3). doi:10.1109/ice348803.2020.9122958

[3] Akram Ahmed Mohammed, RajkumarBasa, Anirudh Kumar Kuchuru, Shiva Prasad Nandigama ,ManeeshwarGangolla, “Random Forest Machine Learning technique to predict Heart disease” European Journal of Molecular & Clinical Medicine ISSN 2515-8260 Volume 7, Issue 4, 2020

[4] RachitMisra ,Pulkit Gupta , Prashuk Jain, “Rachit Misra1 , Pulkit Gupta2 , Prashuk Jain”, July 2021| IJIRT | Volume 8 Issue 2 | ISSN: 2349-6002

[5] Akram Ahmed Mohammed1 , Rajkumar Basa2 , Anirudh Kumar Kuchuru3 , Shiva Prasad Nandigama4 , Maneeshwar Gangolla5 1,2,3,4,5Department of Electronics and Communication Engineering, VidyaJyothi Institute of Technology, Hyderabad, Telangana, 500075, India.

[6] International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 2, Issue 2, July 2022

[7] International Journal of Engineering Research & Technology (IJERT)

ISSN: 2278-0181 Published by, www.ijert.org NCRACES - 2019 Conference ProceedingsAnusha G C, Apoorva M S, Deepthi N, Dhanushree V Student, Department of Computer Science & Engineering, GSSSIETW, MysuruRummanaFirdaus Assistant Professor, Department of Computer Science & Engineering, GSSSIETW, Mysuru

[8] Journal of Network and Innovative Computing ISSN 2160-2174Volume 4 (2016) pp. 175-184 © MIR Labs, www.mirlabs.net/jnic/index.html

[9] Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., & Froelicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. American Journal of Cardiology, 64,304--310.

[10] David W. Aha & Dennis Kibler. "Instance-based prediction of heart-disease presence with the Cleveland database."

[11] Gennari, J.H., Langley, P, & Fisher, D. (1989). Models of incremental concept formation. Artificial Intelligence, 40, 11--61.

[12] www.thelancet.com/journals/eclinm/article/PIIS25895370%2821%29005 30-7/fullte

[13] (Shubhankar Rawat, Published in Towards Data science , Aug,10,2019)

Code

by Alok Tripathi

eeee

ORIGINALITY REPORT

12%	10%	1%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	github.com Internet Source	6%
2	Submitted to Gitam University Student Paper	1%
3	Submitted to Associatie K.U.Leuven Student Paper	1%
4	Submitted to Eastern University Student Paper	1%
5	www.geeksforgeeks.org Internet Source	1%
6	Submitted to University of Wales Institute, Cardiff Student Paper	1%
7	itgit.cu.ac.kr Internet Source	1%
8	Submitted to University of Hong Kong Student Paper	<1%
9	dataskunkworks.com Internet Source	<1%

10	www.codegrepper.com Internet Source	<1 %
11	Submitted to Birkbeck College Student Paper	<1 %
12	Submitted to CSU, San Jose State University Student Paper	<1 %
13	library.samdu.uz Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		

report

by Alok Tripathi

report

ORIGINALITY REPORT

11 %	9%	1%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	5%
2	Submitted to Intercollege Student Paper	1%
3	Submitted to iGroup Student Paper	1%
4	Submitted to Madan Mohan Malaviya University of Technology Student Paper	1%
5	Submitted to University of North Texas Student Paper	1%
6	www.techasoft.com Internet Source	<1%
7	www.researchgate.net Internet Source	<1%
8	www.irjmets.com Internet Source	<1%
9	Minje Seok, Wooseong Kim. "Sarcopenia Prediction for Elderly People Using Machine	<1%

Learning: A Case Study on Physical Activity",
Healthcare, 2023
Publication

10	www.ijraset.com Internet Source	<1%
11	www.javatpoint.com Internet Source	<1%
12	www.mirlabs.net Internet Source	<1%

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		