# Heart Disease Prediction through Hybrid LSTM and HREF Models

*A Project Report submitted in thepartial fulfillment of*
*the Requirements for the award of the degree*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**


**M. Srilakshmi (22471A0541)**
**G. Durga Vyshnavi (22471A0524)**
**SK. K. Rabia Basri (22471A0550)**


Under the esteemed guidance of

Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.

Associate Professor



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

NARASARAOPETAENGINEERING COLLEGE: NARASAROPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and an ISO 9001:2015 Certified
Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601


2025-2026

# NARASARAOPETA ENGINEERING COLLEGE

## (AUTONOMOUS)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING


## CERTIFICATE

**This is to certify that the project that is entitled with the name " Heart Disease Prediction through Hybrid LSTM and HREF Models" is a bonafide work done by**

**M. Srilakshmi (22471A0541), G. Durga Vyshnavi (22471A0524), Sk. K. Rabia Basri (22471A0550) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2025-2026.**

PROJECT GUIDE
**Dr. Sireesha Moturi**, B.Tech., M. Tech., Ph.D.
**Associate Professor**

PROJECT CO-ORDINATOR
**Dr. Sireesha Moturi**, B.Tech., M. Tech., Ph.D.
**Associate Professor**

HEAD OFTHE DEPARTMENT
**Dr. S. N. Tirumala Rao**, M.Tech., Ph.D.
**Professor & HOD**

EXTERNAL EXAMINER

# DECLARATION

We declare that this project work titled "**Heart Disease Prediction through Hybrid LSTM and HREF Models** " is composed by us that the work contain here is my own except where explicitly stated otherwise in the text and that this work has not been submitted for any other degree or professional qualification except as specified.

**M. Srilakshmi (22471A0541)**
**G. Durga Vyshnavi (22471A0524)**
**SK. K. Rabia Basri (22471A0550)**

# ACKNOWLEDGEMENT

We wish to express my thanks to various personalities who are responsible for the completion of our project. We extremely thankful to my beloved chairman**, Sri M. V. Koteswara Rao, B.Sc.**, who took keen interest in me in every effort throughout this course. We owe my sincere gratitude to my beloved principal**, Dr. S. Venkateswarlu, Ph.D.**, for showing his kind attention and valuable guidance throughout the course.

We express my deep-felt gratitude towards **Dr. S. N. Tirumala Rao**, **M.Tech., Ph.D.**, HOD of the CSE department, and also to my guide**, Dr. Moturi Sireesha, B.Tech., M.Tech., Ph.D.,** Professor of the CSE department, whose valuable guidance and unstinting encouragement enabled me to accomplish my project successfully in time.

We extend my sincere thanks to **Dr. Moturi Sireesha, B.Tech., M.Tech., Ph.D.,** Associate Professor & Project Coordinator of the project, for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for me throughout this project work.

We extend my sincere thanks to all the other teaching and non-teaching staff in the department for their cooperation and encouragement during my B.Tech. degree.

We have no words to acknowledge the warm affection, constant inspiration, and encouragement that I received from my parents.

We affectionately acknowledge the encouragement received from my friends and those who were involved in giving valuable suggestions and clarifying my doubts, which really helped me in successfully completing my project.

**By**

**M. Srilakshmi (22471A0541)**
**G. Durga Vyshnavi (22471A0524)**
**SK. K. Rabia Basri (22471A0550)**

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

# INSTITUTE VISION AND MISSION

## INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

## INSTITUTION MISSION

**M1:** Provide the best class infra-structure to explore the field of engineering and research

**M2:** Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

**M3:** Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

# DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERING

## VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

## MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

**M1:** Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

**M2:** Impart high quality professional training to get expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

**M3:** Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

## Program Specific Outcomes (PSO's)

**PSO1:** Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

**PSO2:** Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

**PSO3:** Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

## Program Educational Objectives (PEO's)

The graduates of the programme are able to:

**PEO1:** Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

**PEO2:** Use various software tools and technologies to solve problems related to the academia, industry and society.

**PEO3:** Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

**PEO4:** Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

## Program Outcomes

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science,computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to providevalid conclusions. (WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and

in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in thebroadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

# Project Course Outcomes (CO'S):

**CO421.1:** Analyse the System of Examinations and identify the problem.
**CO421.2:** Identify and classify the requirements. CO421.3: Review the Related Literature CO421.4: Design and Modularize the project
**CO421.5:** Construct, Integrate, Test and Implement the Project.
**CO421.6:** Prepare the project Documentation and present the Report using appropriate method.

**Course Outcomes – Program Outcomes mapping**

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PSO1 | PSO2 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C421.1** | | ✓ | | | | | | | | | | ✓ | | |
| **C421.2** | ✓ | | ✓ | | ✓ | | | | | | | ✓ | | |
| **C421.3** | | | | ✓ | | ✓ | ✓ | ✓ | | | | ✓ | | |
| **C421.4** | | | ✓ | | | ✓ | ✓ | ✓ | | | | ✓ | ✓ | |
| **C421.5** | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **C421.6** | | | | | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | |

**Course Outcomes – Program Outcome correlation**

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PSO1 | PSO2 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C421.1** | 2 | 3 | | | | | | | | | | 2 | | |
| **C421.2** | | | 2 | | 3 | | | | | | | 2 | | |
| **C421.3** | | | | 2 | | 2 | 3 | 3 | | | | 2 | | |
| **C421.4** | | | 2 | | | 1 | 1 | 2 | | | | 3 | 2 | |
| **C421.5** | | | | | 3 | 3 | 3 | 2 | 3 | 2 | 2 | 3 | 2 | 1 |
| **C421.6** | | | | | | | | | 3 | 2 | 1 | 2 | 3 | |

**Note: The values in the above table represent the level of correlation between CO's and PO's:**

1. Low level
2. Medium level
3. High level

## Project mapping with various courses of Curriculum with Attained PO's:

| Name of the course from which principles are applie in this project | Description of the device | Attained PO |
|---|---|---|
| C2204.2, C22L3.2 | Gathering the requirements and defining the problem, plan to develop model for detection and classification of OSCC | PO1, PO3,PO8 |
| CC421.1, C2204.3, C22L3.2 | Each and every requirement i critically analyzed, the process mode is identified | PO2, PO3, PO8 |
| CC421.2, C2204.2, C22L3.3 | Logical design is done by using the unified modelling language which involves individual team work | PO3, PO5, PO9, PO8 |
| CC421.3, C2204.3, C22L3.2 | Each and every module is tested, integrated, and evaluated in our project | PO1, PO5, PO8 |
| CC421.4, C2204.4, C22L3.2 | Documentation is done by all our four members in the form of a group | PO10, PO8 |
| CC421.5, C2204.2, C22L3.3 | Each and every phase of the work in group is presented periodically | PO8,PO10, PO11 |
| C2202.2, C2203.3, C1206.3, C3204.3, C4110.2 | Implementation is done and the project will be handled by the social media users and in future updates in our project can be done based on detection for Oral Cancer | PO4, PO7, PO8 |
| C32SC4.3 | The physical design includes website to check OSCC | PO5, PO6, PO8 |

# ABSTRACT

Cardiovascular disease (CVD) remains a major global health concern and is one of the leading causes of mortality worldwide. Early and accurate diagnosis is essential to reduce complications and improve patient survival rates. This research proposes a novel hybrid deep learning and ensemble-based classification framework by integrating a Bidirectional Long Short-Term Memory (BiLSTM) network with a Hybrid Refined Ensemble Framework (HREF) to enhance prediction performance and reliability. The proposed model is trained and evaluated using the non-standardized Cleveland heart disease dataset, which contains multiple clinical and cardiology-related attributes associated with cardiovascular conditions. A comprehensive preprocessing pipeline is applied to improve data quality, including missing value transformation, removal of extreme outliers, feature normalization, and scaling. To address the class imbalance issue, the SMOTEENN technique is utilized, combining Synthetic Minority Oversampling Technique (SMOTE) with Edited Nearest Neighbors (ENN) to achieve balanced class distribution and improved generalization. The BiLSTM component is employed to capture complex feature interactions and hidden dependencies within the dataset, while the HREF module strengthens classification robustness by refining and aggregating outputs from multiple ensemble learners. Experimental results demonstrate that the proposed hybrid model achieves an accuracy of 94.7% and a ROC-AUC score of 0.9474, along with strong precision, recall, and F1-score values. The outcomes confirm that combining deep learning with refined ensemble learning provides an effective and dependable approach for heart disease prediction, supporting early detection and assisting clinical decision-making in real-world healthcare environments.

# INDEX


| S.NO | CONTENT | PAGE NO |
|---|---|---|
| 1 | INTRODUCTION | 1 |
| | 1.1 MOTIVATION | 5 |
| | 1.2 PROBLEM STATEMENT | 6 |
| | 1.3 OBJECTIVE | 7 |
| 2 | LITERATURE SURVEY | 8 |
| 3 | SYSTEM ANALYSIS | |
| | 3.1 EXISTING SYSTEM | 12 |
| | 3.1.1 DISADVANTAGES OF THE EXISTING SYSTEM | 15 |
| | 3.2 PROPOSED SYSTEM | 16 |
| | 3.3 FEASIBILITY STUDY | 18 |
| 4 | SYSTEM REQUIREMENTS | |
| | 4.1 SOFTWARE REQUIREMENTS | 21 |
| | 4.2 REQUIREMENT ANALYSIS | 21 |
| | 4.3 HARDWARE REQUIREMENTS | 22 |
| | 4.4 SOFTWARE | 22 |
| | 4.5 SOFTWARE DESCRIPTION | 23 |
| 5 | SYSTEM DESIGN | |
| | 5.1 SYSTEM ARCHITECTURE | 25 |
| | 5.1.1 DATASET | 26 |
| | 5.1.2 DATA PREPROCESSING | 27 |
| | 5.1.3 MODEL BUILDING | 28 |
| | 5.2 MODULES | 29 |
| 6 | IMPLEMENTATION | |
| | 6.1 MODEL IMPLEMENTATION | 32 |
| | 6.2 CODING | 35 |
| 7 | TESTING | |
| | 7.1 UNIT TESTING | 73 |
| | 7.2 INTEGRATION TESTING | 75 |
| | 7.3 SYSTEM TESTING | 80 |
| 8 | RESULT ANALYSIS | 84 |

9 OUTPUT SCREENS 92
10 CONCLUSION 94
11 FUTURE SCOPE 96
12 REFERENCES 98

# LIST OF FIGURES


| S.NO | FIGURE DESCRIPTION | PAGE NO |
|---|---|---|
| 1 | FIG 3.1. FLOW CHART OF EXISTING SYSTEM FOR HEART DISEASE PREDICTION | 13 |
| 2 | FIG 3.2. FLOW CHART OF PROPOSED SYSTEM | 16 |
| 3 | FIG 5.2 IMAGE AFTER APPLYING PREPROCESSING | 28 |
| 4 | FIG 7.1 STATUS HEART DISEASE DETECTED | 82 |
| 5 | FIG 7.2 STATUS NO HEART DISEASE DETECTED | 83 |
| 6 | FIG 7.3 STATUS INVALID MESSAGE | 83 |
| 7 | FIG 8.1 CONFUSION MATRIX OF HREF + BiLSTM MODELS | 86 |
| 8 | FIG 8.2 ROC-AUC CURVE | 87 |
| 9 | FIG 8.3 TRAINING VS TESTING VALIDATION | 87 |
| 10 | FIG 8.4 OUTLIER REMOVAL | 88 |
| 11 | FIG 8.5 TRAIN VS TEST SPLIT DISTRIBUTION | 89 |
| 12 | FIG 9.1 HOME PAGE | 92 |
| 13 | FIG 9.2 ABOUT PAGE | 93 |
| 14 | FIG 9.3 PREDICTION PAGE | 93 |

## List of Tables


| S.NO | CONTENT | PAGE NO |
|---|---|---|
| 1 | TABLE 1. DATASET DESCRIPTION | 27 |
| 2 | TABLE 2. FINAL MODEL PERFORMANCE(HYBRID HREF) | 86 |
| 3 | TABLE 3. TRAIN-TEST SPLIT DISTRIBUTION | 89 |
| 4 | TABLE 4. MODEL PERFORMANCE COMPARISON | 90 |

# 1. INTRODUCTION

The prediction and diagnosis of heart disease have become one of the most pressing challenges in modern healthcare. Cardiovascular disease (CVDs) is the leading cause of mortality worldwide, responsible for nearly 17.9 million deaths annually, according to the World Health Organization (WHO). With the growing global population, urbanization, and changing lifestyle patterns, the prevalence of heart-related conditions such as coronary artery disease, heart attack, and stroke continues to rise alarmingly. It is projected that by the year 2030, CVDs will remain the dominant cause of death across all regions, creating an unprecedented strain on healthcare systems worldwide. These statistics underscore the urgent necessity for early diagnosis, accurate prediction, and proactive management of heart disease to reduce mortality rates and improve patient outcomes.

Traditional diagnostic techniques, including electrocardiography (ECG), angiography, echocardiography, and stress testing, play an essential role in assessing cardiac conditions. However, these methods often rely heavily on manual interpretation and clinical expertise, making them time-consuming, and prone to human error. Furthermore, they typically focus on evaluating specific physiological indicators rather than integrating multiple interdependent factors such as blood pressure, cholesterol levels, chest pain type, and blood sugar. This limitation restricts the ability to uncover hidden nonlinear relationships between risk factors that contribute to the onset of heart disease.

In recent years, Machine Learning (ML) and Deep Learning (DL) have revolutionized healthcare analytics by enabling computers to learn patterns from clinical data and make accurate predictions. ML algorithms can analyze large datasets to identify correlations between patient characteristics and disease outcomes, supporting clinicians in decision-making. Studies demonstrated that machine learning techniques can reveal hidden associations among cardiac indicators, while Narasimhan and Victor showed that optimization-based ML models improve the predictive accuracy of disease classification systems. However, ML models rely on handcrafted features, making them sensitive to data preprocessing and often inadequate in capturing complex nonlinear dependencies within medical data.

Deep Learning, on the other hand, has shown remarkable success in extracting meaningful representations from raw data, offering greater adaptability and robustness. Techniques like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks have proven effective in various biomedical applications. Specifically, LSTM networks are designed to model sequential dependencies and can retain long-term contextual information from input data. Demonstrated that deep architectures outperform conventional models by effectively learning inter-feature dependencies, thereby improving overall diagnostic accuracy. However, deep models often face challenges such as high computational cost, overfitting, and imbalanced datasets, which limit their deployment in clinical settings.

To address these challenges, recent research has increasingly focused on hybrid learning architectures that combine the strengths of both ML and DL techniques. Ensemble learning, in particular, has gained popularity for its ability to combine multiple classifiers to improve stability and predictive reliability. Highlighted that ensemble-based models outperform single classifiers in heart disease prediction, while emphasized that hybrid deep learning approaches effectively capture nonlinear relationships in medical datasets. Motivated by these advancements, this study introduces a novel Hybrid Refined Ensemble Framework (HREF) integrated with a Bidirectional Long Short-Term Memory (BiLSTM) network for heart disease prediction.

The BiLSTM architecture is a powerful extension of LSTM that processes data in both forward and backward directions, allowing it to capture contextual dependencies among clinical features more comprehensively. This bidirectional flow helps the model identify intricate relationships between medical variables such as age, cholesterol level, resting blood pressure, and ECG readings. The HREF component further enhances prediction performance by combining the outputs of multiple ensemble models—such as Random Forest, AdaBoost, and Support Vector Machines (SVM) through a stacked meta-learner. This multi-model integration significantly reduces overfitting, improves generalization, and ensures more consistent predictions, making the framework suitable for real-world healthcare applications.

The proposed hybrid model is trained and tested using the Cleveland Heart Disease dataset, one of the most widely used benchmark datasets in cardiovascular research. It contains 303 patient records and 14 clinical features, each representing medically relevant attributes. To ensure reliable and unbiased predictions, the dataset undergoes a comprehensive preprocessing pipeline that includes outlier removal, missing value imputation, feature normalization, and class balancing using the SMOTEENN technique. This hybrid oversampling and under sampling strategy (SMOTE + Edited Nearest Neighbors) eliminates noisy data points and balances class distributions, thus improving the learning process. The dataset is then split into training (80%) and testing (20%) subsets to evaluate the model's performance under realistic conditions.

The proposed BiLSTM + HREF model follows a structured workflow comprising several stages—data preprocessing, BiLSTM-based feature extraction, ensemble integration, and final classification. Initially, the BiLSTM network learns high-level representations from the processed dataset by modeling inter-feature dependencies in both directions. The extracted features are then passed to the HREF layer, where multiple base learners generate independent predictions. Finally, a meta-learner combines these results to produce the final diagnosis, ensuring a balanced and robust output. The performance of the model is evaluated using accuracy, precision, recall**,** F1-score, and ROC-AUC metrics, providing a comprehensive understanding of its predictive capability.

Experimental results demonstrate that the Hybrid BiLSTM-HREF model achieves an impressive accuracy of 94.7% and an ROC-AUC of 0.9474, outperforming individual baseline models such as Logistic Regression, Random Forest, and standalone BiLSTM. These results confirm that integrating deep sequential learning with ensemble refinement significantly enhances diagnostic accuracy and model reliability. Moreover, the hybrid approach minimizes both false positives and false negatives, which is crucial in medical applications where misclassification can have severe consequences.

The clinical significance of this work lies in its potential to serve as an intelligent decision-support tool for healthcare professionals. By providing reliable and automated heart disease predictions, the proposed model assists doctors in making informed diagnoses, optimizing treatment plans, and prioritizing high-risk patients. Furthermore, the model's robust design allows for potential integration into real-time

clinical systems, mobile health applications, and remote patient monitoring platforms.

In summary, the primary objective of this research is to develop a hybrid deep learning and ensemble-based framework that enhances the accuracy, stability, and interpretability of heart disease prediction. The integration of Bidirectional LSTM for sequential feature learning and HREF for ensemble-based refinement represents a significant step toward achieving reliable clinical decision support. The results demonstrate that combining deep feature extraction with ensemble generalization leads to improved diagnostic outcomes and reduced computational inefficiencies. Ultimately, this study contributes to the growing field of intelligent healthcare analytics, advancing the use of artificial intelligence for early disease detection, preventive care, and sustainable healthcare delivery.

## 1.1 Motivation

Heart disease is one of the most serious and life-threatening health conditions, with a significant impact on both individuals and healthcare systems around the world. Early and accurate prediction of heart disease is crucial for effective treatment and prevention of complications. However, traditional diagnostic methods—such as ECG interpretation, angiography, and clinical assessments—are often time-consuming, subjective, and prone to human error, especially when patient symptoms are complex or influenced by multiple factors. This creates a strong need for automated and intelligent systems that can assist doctors in making faster and more reliable decisions.

The growing use of digital medical records and health data has made it possible to develop automated systems for predicting heart disease. These records contain important details such as blood pressure, cholesterol, blood sugar, and chest pain type. Analyzing this information manually can be difficult and may lead to errors. Machine Learning (ML) and Deep Learning (DL) can help overcome this problem by automatically finding hidden patterns and relationships in the data that doctors might miss.

A strong model for automated heart disease prediction is developed to reduce diagnosis time, minimize human error, and assist doctors in making more accurate decisions. The proposed system combines a Bidirectional Long Short-Term Memory (BiLSTM) network with a Hybrid Refined Ensemble Framework (HREF) to improve prediction accuracy and stability. The BiLSTM model captures complex relationships among patient features, while the HREF integrates results from multiple classifiers such as Random Forest, AdaBoost, and SVM. This hybrid approach offers a fast and reliable method to identify individuals at risk of heart disease and supports early medical intervention.

Furthermore, integrating this hybrid model available through a simple web or mobile application can help doctors and healthcare workers use it easily. By entering patient details such as age, blood pressure, cholesterol, and ECG results, they can quickly get diagnostic feedback and risk assessments. This approach improves early detection, lowers healthcare costs, and helps patients receive timely treatment. Overall, it supports the global effort to reduce deaths from heart disease using smart and automated healthcare technology

## 1.2 Problem Statement

Heart disease is a highly serious and life-threatening condition that can cause paralysis, chest pain, shortness of breath, and other long-term complications. Its impact varies depending on several critical factors such as the patient's age, gender, blood pressure, cholesterol level, blood sugar, and the type of chest pain experienced. Cardiovascular diseases often lead to disabilities that may require prolonged medication, lifestyle changes, or even surgical procedures that focus mainly on managing symptoms rather than completely curing the condition. The severity of heart damage depends on how much the arteries are blocked and how the heart muscles are affected. Early diagnosis is very important because once severe symptoms appear, it becomes difficult to reverse the damage. However, predicting heart disease early is challenging because patient data varies widely. Factors like cholesterol, blood pressure, and heart rate differ from person to person, making manual diagnosis difficult and sometimes inaccurate.

Since some patients show similar symptoms even when their conditions are different, it becomes hard to clearly identify who has heart disease and who does not. Wrong predictions can delay treatment or cause unnecessary medical procedures. Therefore, a reliable and accurate system is needed to predict heart disease efficiently. To overcome these challenges, this research proposes the use of advanced computational techniques — specifically, a Hybrid Refined Ensemble Framework (HREF) integrated with a Bidirectional Long Short-Term Memory (BiLSTM) network — to improve prediction accuracy and reliability. Early and accurate detection through such models can significantly reduce complications and mortality rates. Hence, the development of intelligent and automated tools for quick and precise heart disease prediction is essential to facilitate timely diagnosis, support clinical decision-making, and improve the overall quality of patient care.

## 1.3 Objective

The primary objective of this project is to develop an automated, accurate, and efficient heart disease prediction and classification system using a Hybrid BiLSTM and Hybrid Refined Ensemble Framework (HREF) approach. By integrating the sequential learning capability of Bidirectional Long Short-Term Memory (BiLSTM) networks with the refinement and stability of the HREF ensemble strategy, the proposed system aims to achieve improved prediction performance on clinical heart disease data. The specific objectives of the project are as follows:

- To develop an automated system capable of accurately predicting and classifying heart disease based on clinical patient data.
- To build a robust hybrid model that classifies patient records into "heart disease" and "no heart disease" categories by combining BiLSTM for sequential feature learning with HREF for refined and accurate classification.
- To optimize the model's performance using advanced preprocessing techniques such as normalization, feature scaling, and class balancing to ensure high accuracy, precision, and reliability.
- To develop a user-friendly web application using Python Flask that allows users to input clinical parameters (such as age, cholesterol, blood pressure, and heart rate) and instantly obtain heart disease prediction results.
- To implement data validation mechanisms to ensure that only valid and complete medical inputs are accepted, while providing clear error messages for missing or incorrect values.
- To support healthcare professionals by enhancing diagnostic decision-making, reducing evaluation time, minimizing human error, promoting early diagnosis, and improving overall patient outcomes.
- To design the system for scalability and adaptability, enabling future enhancements such as integration with hospital databases or IoT-based health monitoring systems for real-time prediction and improved clinical support.

# 2. LITERATURE SURVEY

Deep learning and machine learning approaches for heart disease prediction and classification have been widely explored in recent years due to the growing need for accurate, automated, and early diagnostic systems. Traditional machine learning techniques such as Support Vector Machines (SVMs) and Random Forest (RF) classifiers have long been applied to clinical and ECG datasets for cardiac risk assessment. However, these methods often rely heavily on manual feature engineering and may struggle to capture complex temporal and non-linear interactions among clinical variables. In contrast, deep learning methods—particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks—have shown significant gains in predictive performance by effectively modelling temporal dependencies in physiological signals and learning hierarchical representations from raw or minimally processed data.

With the rise of advanced architectures and transfer learning for signal and tabular domains, researchers have proposed models that substantially improve heart-disease detection and prognosis. Transfer learning and pretraining strategies, including using pre-trained feature extractors for ECG waveforms or leveraging large publicly available cardiovascular datasets, help to overcome limitations posed by smaller labeled clinical datasets. Additionally, multimodal fusion techniques—combining ECG traces, clinical history, laboratory values, and imaging-derived features—along with robust preprocessing strategies have emerged as powerful tools for refining data quality and optimizing feature extraction. The following studies summarize key recent advances in deep learning and hybrid approaches for heart disease prediction.

*Alghamdi et al. [1]* proposed a hybrid autoencoder–DenseNet model for heart disease prediction. Their approach focused on improving feature learning and classification accuracy by combining deep representation learning with dense neural networks for better performance on clinical datasets.

*Gabriel et al. [2]* developed a cardiovascular disease detection model using optimized feature selection and hyperparameter-tuned LightGBM. Their work showed that selecting relevant features and tuning the model significantly improves prediction accuracy and reduces computational complexity.

*Boquete et al. [3]* introduced an early heart disease detection system using machine learning with strong feature engineering techniques. Their study emphasized that effective feature extraction and selection improve classification results across multiple models.

*Narasimhan et al. [4]* proposed a genetic algorithm optimized Random Forest model for heart disease prediction. Their research demonstrated that evolutionary optimization improves feature selection and enhances the overall performance of ensemble classifiers.

*Dede Turk et al. [5]* proposed CSA-DE-LR, a hybrid approach for cardiovascular disease diagnosis. Their method integrates optimization and classification techniques to improve detection accuracy, particularly on structured medical datasets.

*Teja et al. [6]* evaluated the performance of multiple ensemble models for heart disease classification. Their findings showed that ensemble learning methods such as Random Forest, boosting, and stacking provide better reliability compared to single classifiers.

*Zhou et al. [7]* presented a comprehensive review of deep learning models used for cardiovascular disease prediction. Their study discusses the effectiveness of modern architectures such as CNNs, RNNs, LSTMs, BiLSTMs, and transformer-based models in improving predictive accuracy for heart disease detection. The authors also highlighted the importance of preprocessing, feature extraction, and optimization techniques, while noting challenges such as computational cost and model interpretability.

*Navita et al. [8]* introduced a hybrid machine learning model using SMOTE-ENN and stacking ensemble methods for cardiovascular disease detection. Their study demonstrated improved class balancing and strong classification performance on imbalanced clinical datasets.

*Wu et al. [9]* proposed a stacking ensemble machine learning approach for heart disease risk prediction. Their work showed that combining multiple base learners through a stacking framework improves accuracy and generalization compared to individual classifiers.

*García-Ordas et al. [10]* proposed deep learning models with enhanced feature

engineering for heart disease risk assessment. Their work emphasized that combining deep models with refined feature processing improves prediction effectiveness.

*Koteeswaran et al. [11]* proposed a hybrid feature selection and optimized classification approach for heart disease prediction. Their research focused on selecting optimal features and using tuned classifiers to improve diagnostic accuracy.

*Waris et al. [12]* proposed an improved K-means neighbor classifier for early heart disease detection using Python. Their approach highlighted the benefits of clustering-based classification in identifying cardiovascular risk patterns.

*Niu et al. [13]* applied an enhanced Grey Wolf Optimization algorithm for predicting heart disease. Their work demonstrated that optimization-based feature selection improves the performance of machine learning classifiers.

*Alzaqeeb et al. [14]* proposed a hybrid transformer-based approach for predicting heart disease using structured clinical datasets. Their method used transformer learning to capture complex feature relationships and improve prediction accuracy.

*Kumar et al. [15]* proposed a medical record-based hybrid ensemble deep learning model for heart disease prediction. Their study showed that combining deep learning and ensemble strategies enhances prediction reliability.

*Seva et al. [16]* proposed a Random Forest-based prediction system using SMOTE-ENN balancing. Their work demonstrated that SMOTE-ENN improves class distribution and enhances classifier performance.

*Venkatareddy et al. [17]* developed an explainable CNN-MLP model for fetal ultrasound classification. Their work focused on deep learning classification with explainability, which supports medical decision-making.

*Reddy et al. [18]* developed an automated traffic sign recognition system using CNN deep learning. Their study showed the effectiveness of CNN models for real-time classification tasks.

*Rao et al. [19]* proposed a machine learning approach for fake profile detection. Their work focused on classification and prediction methods using machine learning algorithms.

*Moturi et al. [20]* proposed an optimized feature extraction and hybrid classification model for heart disease and breast cancer prediction. Their study

demonstrated that feature optimization and ensemble classification improve prediction accuracy.

Scalability is another advantage: as more data and compute become available, deep models typically yield improved accuracy. Techniques such as transfer learning (pretraining on large physiological datasets) and data augmentation (time-warping, noise addition, synthetic beat generation) make deep approaches practical even with limited labeled data. This adaptability and robustness have established deep learning as a cornerstone of contemporary AI-driven cardiovascular diagnostics.

Overall, the reviewed literature confirms that heart disease prediction has significantly improved with the use of machine learning and deep learning techniques. Traditional models such as SVM, Random Forest, and LightGBM provide reliable results, but their performance often depends on manual feature engineering and careful parameter tuning. Recent studies show that deep learning models such as LSTM, BiLSTM, CNN-LSTM, and transformer-based architectures achieve better accuracy by automatically learning complex non-linear and temporal relationships from clinical and ECG data.

# 3. SYSTEM ANALYSIS

## 3.1 EXISTING SYSTEM

Heart disease prediction has traditionally relied on manual analysis of patient health records and clinical test results by doctors and cardiologists. This manual process, while valuable, is time-consuming and depends heavily on the medical expert's knowledge and experience. Human interpretation is often prone to error, which can lead to misdiagnosis, delayed treatment, or unnecessary medical procedures. To overcome these limitations, computational prediction methods have been introduced, ranging from traditional Machine Learning (ML) algorithms to advanced Deep Learning models.

Early Machine Learning approaches such as Logistic Regression (LR), Support Vector Machines (SVMs), Random Forest (RF), k-Nearest Neighbors (k-NN), and Decision Trees were commonly used to analyze patient datasets like the Cleveland Heart Disease Dataset. These models relied on handcrafted feature selection using parameters such as age, blood pressure, cholesterol, blood sugar, and heart rate. However, these traditional models required significant preprocessing and manual feature engineering to achieve acceptable performance. Although they achieved reasonable accuracy (typically around 70–85%), their performance was limited by overfitting, dependency on selected features, and poor generalization when applied to different patient datasets.

The introduction of Deep Learning techniques, particularly Long Short-Term Memory (LSTM) networks and Bidirectional LSTM (BiLSTM) models, revolutionized heart disease prediction by enabling automatic feature learning from sequential and tabular medical data. These models can capture hidden temporal relationships and interdependencies among health parameters, improving prediction accuracy. BiLSTM architectures have shown significant advantages over traditional ML methods, achieving accuracy levels ranging between 85–94% in many studies. However, such models also face challenges, including the need for balanced datasets, larger training samples, and high computational costs, which can lead to overfitting if not properly optimized.

To overcome these limitations, researchers have introduced ensemble and hybrid learning approaches, combining multiple algorithms to improve accuracy and reliability. Models like Random Forest + XGBoost, SVM + Neural Networks, and LSTM + Ensemble frameworks have demonstrated promising results. Among these, Hybrid Ensemble Frameworks (HREF) have shown better generalization by integrating multiple base learners and a meta-learner to refine predictions. This hybridization enhances model robustness and reduces classification errors by leveraging the strengths of both deep learning and ensemble methods.

In recent systems, advanced preprocessing techniques such as Z-score normalization, feature scaling, and SMOTEENN (Synthetic Minority Over-sampling Technique combined with Edited Nearest Neighbors) have been applied to handle imbalanced data and improve the training quality. These steps ensure that the dataset is standardized, balanced, and suitable for accurate model training and evaluation.

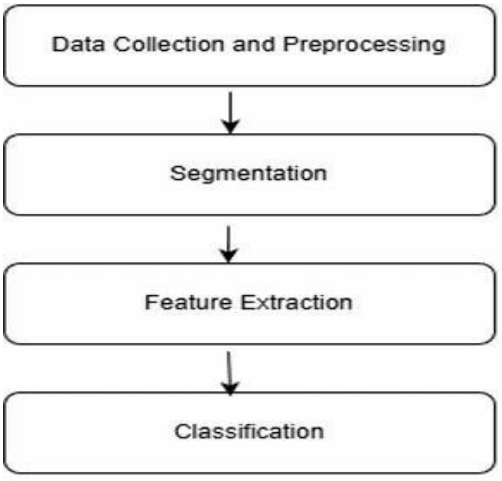



**FIG 3.1: FLOW CHART OF EXISTING SYSTEM FOR HEART DISEASE PREDICTION**

This flowchart (Fig. 3.1) illustrates a typical data processing pipeline for heart disease prediction using clinical datasets such as the Cleveland dataset. The process begins with an input patient database, which undergoes data preprocessing to clean and prepare the records. Preprocessing includes handling missing values, removing outliers, normalizing features, and balancing the dataset to reduce bias and improve model performance. These steps ensure that the data is consistent and ready for further analysis.

Once preprocessing is complete, the feature extraction and selection phase identifies the most relevant clinical features influencing heart disease prediction. Techniques like Principal Component Analysis (PCA) or correlation-based selection are applied to reduce dimensionality and eliminate redundant attributes.

Finally, classification algorithms such as Logistic Regression, SVM, Random Forest, or Neural Networks are used to categorize patients into “Heart Disease” or “No Heart Disease” classes. Although these existing models have improved accuracy and efficiency compared to manual diagnosis, they still face challenges in achieving high precision and generalization across different datasets. Hence, there is a continuous need for an enhanced hybrid model, such as the proposed BiLSTM + HREF framework, which can further improve accuracy, reduce false predictions, and assist healthcare professionals in early and reliable heart disease diagnosis.

### 3.1.1 DISADVANTAGES

Despite significant advancements in automated heart disease prediction and classification, the existing systems face several limitations:

1. **Manual Dependence:**

Traditional diagnostic methods depend heavily on the experience and interpretation of cardiologists, which can lead to human errors, inconsistent judgments, and delayed diagnosis.

2. **Limited Feature Understanding:**

Machine learning models like SVM, k-NN, and Decision Trees rely on manually selected features. These models cannot automatically learn complex, non-linear relationships between clinical parameters such as blood pressure, cholesterol, and ECG readings.

3. **Low Accuracy and Generalization:**

Classical models often achieve moderate accuracy (70–85%) and perform poorly when tested on new or unseen datasets, limiting their reliability for real-world medical use.

4. **Data Imbalance Issues:**

Most existing datasets contain unequal samples of “heart disease” and “no heart disease” cases, causing models to become biased toward the majority class and miss actual disease cases.

5. **High Preprocessing Efforts:**

Traditional systems require extensive preprocessing, normalization, and manual feature engineering to obtain meaningful results, which increases complexity and time.

6. **Overfitting on Small Datasets:**

Due to limited medical data, traditional ML models are prone to overfitting, leading to inaccurate predictions on unseen patients.

## 3.2 PROPOSED SYSTEM

The proposed system introduces a hybrid deep learning–ensemble learning model designed to improve the accuracy and reliability of heart disease prediction using clinical data. This system integrates a Bidirectional Long Short-Term Memory (BiLSTM) network with a Hybrid Refined Ensemble Framework (HREF), allowing the model to capture complex feature relationships while improving prediction stability through ensemble refinement.

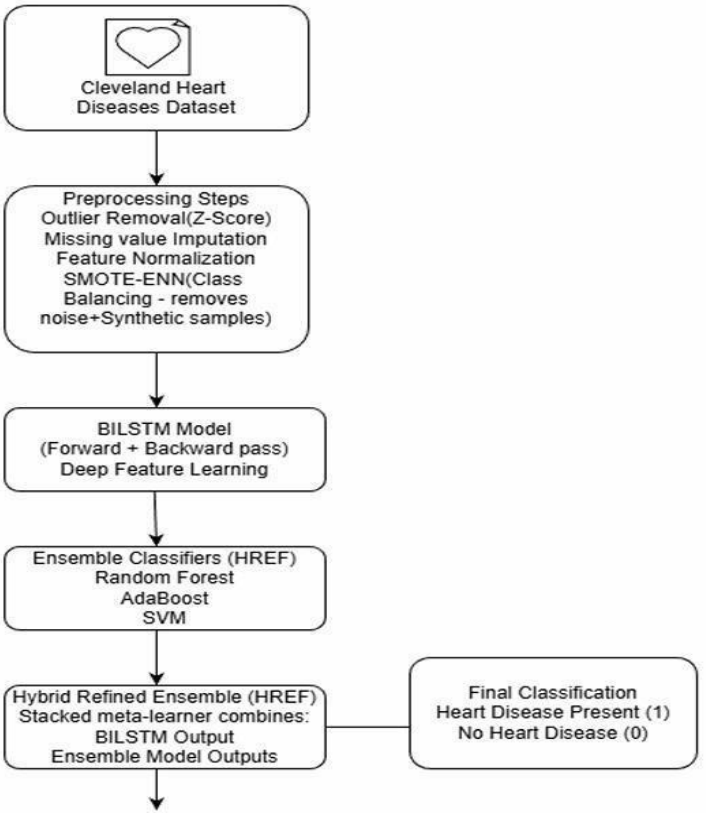



**FIG 3.2 FLOWCHART OF PROPOSED SYSTEM**

To begin with, the flowchat (Fig. 3.2) employs an enhanced preprocessing pipeline that involves outlier removal, missing-value imputation, feature normalization, and class balancing using SMOTEENN. These steps ensure that noise, skewed distributions, and imbalanced classes are effectively handled before training the model. The processed data is then divided into 80% training and 20% testing**,** maintaining class proportions through stratified splitting.

The core component of the system, the BiLSTM model, processes clinical features in both forward and backward directions, enabling the extraction of deep nonlinear patterns within the dataset. This helps the model understand highly correlated medical variables such as chest pain type, cholesterol, blood pressure, and ECG results more effectively than traditional models.

Parallel to this, the system uses multiple ensemble classifiers such as Random Forest,

AdaBoost, and SVM, which generate diverse decision boundaries. These predictions are further refined using the HREF framework, where a stacked meta-learner integrates outputs from both the BiLSTM and ensemble models. This hybrid mechanism reduces overfitting and enhances generalization, especially in noisy or imbalanced clinical datasets.

The proposed system achieves high performance with an accuracy of 94.7% and a ROC–AUC score of 0.9474, outperforming individual models like Logistic Regression, Random Forest, XGBoost, and standalone BiLSTM. With low false-positive and false-negative rates, the model reliably distinguishes between healthy and high-risk patients.

Overall, the system ensures robust, accurate, and clinically meaningful predictions, making it a suitable diagnostic support tool for early identification of heart disease.

## Advantages over exsisting system:

1. **Higher Prediction Accuracy:** The hybrid BiLSTM + HREF model demonstrates significantly higher accuracy compareaditional machine learning approaches such as Logistic Regression, Random Forest, or SVM alone.This is because the system combines deep learning with ensemble learning, enabling better decision-making and more precise classification of heart disease.

2. **Handles Imbalanced Data Effectively:** Heart disease datasets often contain unequal numbers of positive and negative cases. The proposed system uses SMOTE-ENN, which not only balances the dataset by generating synthetic samples, but also removes noisy or incorrectly labeled data.This leads to more robust and unbiased training, something conventional models cannot achieve.

3. **Learns Complex Medical Patterns:** The BiLSTM model analyzes data in both forward and backward directions, allowing it to capture hidden patterns and relationships between medical features like cholesterol, blood pressure, ECG results, and age.Traditional ML models are limited to linear or shallow representations, making them less effective for complex medical data.

4. **More Reliable and Stable Predictions:** The Hybrid Refined Ensemble Framework (HREF) enhances stability by combining predictions from multiple powerful models (Random Forest, AdaBoost, SVM). This reduces the impact of individual model errors and

ensures more consistent and generalizable predictions across different patient groups.

5. **Lower False Negative Rate:** In medical diagnosis, false negatives (predicting "no disease" when the patient actually has disease) are extremely dangerous.he proposed system significantly reduces false negatives by using deep learning and ensemble refinement, ensuring that high-risk patients are accurately identified and can receive early treatment.

## 3.3 FEASIBILITY STUDY

A feasibility study evaluates whether the proposed Hybrid BiLSTM–HREF system for heart disease prediction can be successfully developed and implemented. This study includes Technical Feasibility, Operational Feasibility, and Economic Feasibility.

### 1. Technical Feasibility

- **Availability of Technology**

The proposed system utilizes widely available and well-documented technologies such as Python, TensorFlow, Keras, and Scikit-learn. These frameworks provide built-in support for building BiLSTM networks, ensemble classifiers, and preprocessing pipelines. Their strong community support, extensive libraries, and continuous updates ensure that the development process remains smooth and future-proof. This makes the system technically practical for both research and industrial environments.

- **Dataset Compatibility**

The Cleveland Heart Disease dataset is a publicly accessible, standardized, and well-structured medical dataset. It contains essential clinical parameters required for heart disease prediction and aligns perfectly with the proposed preprocessing steps such as outlier removal, normalization, and SMOTE-ENN balancing. Because of its clean structure and numerical nature, it integrates seamlessly into the training pipeline without requiring major modifications.

- **Hardware Requirements**

The system does not require expensive or high-end hardware. A moderate GPU, cloud-based computing platform, or even a standard workstation is sufficient to train and deploy the Hybrid

BiLSTM–HREF model. This ensures that institutions with limited resources, such as academic labs or small healthcare centers, can easily implement the system. The lightweight architecture also supports faster computation and real-time prediction.

- **System Integration**

All components—including preprocessing modules, BiLSTM model, ensemble classifiers, and the HREF framework—work in coordination without compatibility issues. The modular design allows easy integration into existing medical software systems. Developers can also extend or upgrade the system without rewriting entire components, ensuring long-term technical sustainability.

## 2. Operational Feasibility

- **Ease of Use**

The system produces clear and simple outputs such as "Disease Present" or "No Disease", making it easy for doctors, nurses, and technicians to adopt. No advanced technical or machine learning knowledge is required to interpret the results. This user-friendly output ensures that the system can be used effectively in busy clinical environments.

- **User Acceptability**

Medical staff can easily accept and rely on the system because it supports decision-making rather than replacing human expertise. By offering accurate predictions and reducing diagnostic uncertainty, the system helps healthcare professionals make better clinical judgments. Its simplicity and reliability encourage quick acceptance among users.

- **Workflow Integration**

The system can be deployed through a computer application, a web-based interface, or integrated into an existing hospital management system. This flexibility makes it suitable for telemedicine, emergency care, outpatient screenings, or routine checkups. Since it does not interfere with ongoing processes, its adoption does not disrupt the existing workflow.

- **Reliability in Practice**

The Hybrid BiLSTM–HREF model has shown high accuracy, strong generalization capability, and a significantly low false-negative rate. This ensures that real patients at risk are identified

correctly and on time. The model's stable performance across various test scenarios makes it dependable for real-time clinical usage and long-term healthcare support.

## 3. Economic Feasibility

- **Development Costs**

The proposed system uses open-source tools and freely available datasets, which substantially reduces development expenses. There are no licensing costs, and developers can utilize free cloud tiers or local machines for model training. This makes the system economically feasible even for academic projects and small hospitals.

- **Deployment Costs**

Deploying the system requires low-cost cloud infrastructure or a basic server system. It does not demand high-end hardware, medical equipment, or specialized installations. This cost-efficiency allows wider adoption, especially in rural hospitals, small clinics, and developing regions.

- **Maintenance Costs**

The system is easy to maintain because it requires only occasional updates to the model or software environment. Most libraries and tools update automatically, further reducing technical overhead. The lightweight nature of the system ensures low hardware wear and low long-term operational costs.

- **Cost-Benefit Analysis**

The financial benefits of early heart disease prediction significantly outweigh the setup costs. Early detection reduces long-term treatment expenses, hospitalization charges, and emergency care costs. By preventing severe complications, the system indirectly saves both patients and hospitals substantial amounts of money. Thus, it offers excellent long-term economic value.

# 4. SYSTEM REQUIREMENTS

## 4.1 SOFTWARE REQUIREMENTS

1. Operating System : Windows 11, 64-bit Operating System
2. Hardware Accelerator : CPU / GPU optional for faster BiLSTM training
3. Coding Language : Python 3.8 or above
4. Python Distribution : Google Colab, Jupyter Notebook
5. Web Framework : Flask
6. Browser : Any latest browser like Google Chrome, Microsoft Edge
7. Dataset Requirement : Cleveland Heart Disease Dataset

## 4.2 REQUIREMENT ANALYSIS

The Heart Disease Prediction System requires accurate data preprocessing, including outlier removal, missing value imputation, normalization, and SMOTEENN-based class balancing to ensure reliable model training. The system must support the implementation of a Hybrid BiLSTM model combined with the HREF ensemble framework, enabling efficient learning of clinical patterns and improved prediction accuracy. It should handle the Cleveland Heart Disease dataset, perform model training, generate performance metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and produce visual outputs like confusion matrices and training–validation curves. Additionally, the system should operate efficiently on a standard Python environment such as Google Colab or Jupyter Notebook, ensuring high reliability, fast processing, and ease of use for clinical decision support.

In addition to these core requirements, the system must also ensure that all preprocessing and model execution steps are optimized to handle clinical data variability without compromising accuracy. The Hybrid BiLSTM–HREF framework should maintain stability across multiple training cycles, preventing overfitting through techniques like dropout layers and proper data scaling. The architecture must also allow easy modification for future enhancements, such as adding new clinical parameters or integrating more advanced deep learning methods. Moreover, the system should support clear visualization of results to help

users interpret model behavior, including graphs for outlier detection, normalization effects, and class distribution changes. Ensuring smooth runtime performance, compatibility with commonly used Python libraries, and flexibility for deployment further strengthens the system's effectiveness as a reliable clinical decision-support tool.

## 4.3 HARDWARE REQUIREMENTS

1. System Type : 64-bit Operating System, x64-based Processor
2. Cache Memory : 4 MB
3. RAM : 16 GB
4. Hard Disk : 8 GB Free Space
5. GPU : Intel® Iris® Xe Graphics

## 4.4 SOFTWARE

The proposed system is implemented using Python, as it offers powerful libraries for machine learning, deep learning, and data preprocessing. Python's flexibility and simplicity make it ideal for handling medical datasets and building predictive models.

The development environment used for this project is Google Colab and Jupyter Notebook, which provide GPU/CPU support, easy code execution, and built-in tools for visualizing results. These platforms make it convenient to train the BiLSTM model and perform ensemble learning without requiring high-end hardware.

For preprocessing tasks such as outlier removal, normalization, missing value imputation, and SMOTEENN class balancing, libraries like NumPy, Pandas, SciPy, and Imbalanced-learn are used. These tools help clean and prepare the Cleveland Heart Disease dataset for effective model training.

The deep learning portion of the system is developed using TensorFlow and Keras, which support the implementation of the Bidirectional LSTM (BiLSTM) architecture. These frameworks handle model creation, training, dropout integration, and performance optimization.

Ensemble models including Random Forest, AdaBoost, and SVM are implemented using the Scikit-learn library. This library also provides essential evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC, which are required to assess the performance of the Hybrid BiLSTM-HREF model.

For visualization, libraries like Matplotlib and Seaborn are used to generate plots such as confusion matrices, training vs. validation accuracy curves, feature distributions, and ROC curves. These visual outputs help interpret the model's performance clearly and effectively.
In addition to the core libraries, the system uses several built-in Python utilities for tasks such as data splitting, label encoding, and normalization, which ensure consistency throughout the preprocessing workflow. These utilities allow the model to receive clean, standardized inputs, which greatly improves prediction accuracy and stability.

The project also relies on Google Colab's cloud-based runtime, which supports both CPU and GPU execution. This environment allows faster model training, especially for the BiLSTM network, without requiring any physical GPU hardware. Colab's notebook interface also simplifies documentation and experiment tracking.

For implementing the HREF (Hybrid Refined Ensemble Framework), the system integrates predictions from multiple models, combining deep learning outputs with classical machine learning classifiers. This ensemble strategy is coded using a combination of TensorFlow and Scikit-learn, enabling seamless communication between the models.

Error analysis and model validation are also supported through software tools that generate graphs and statistical reports. Confusion matrices, ROC curves, and accuracy plots help researchers assess the model's reliability and identify areas for improvement. These visual tools play a key role in interpreting medical prediction results.

Finally, the entire software stack is designed to be portable and lightweight, ensuring that users can run the system on any standard device with minimal configuration. This flexibility supports future project expansion, such as deploying the model using Flask as a web application or integrating it into a clinical decision-support system.

## 4.5 SOFTWARE DESCRIPTION

The Heart Disease Prediction System is developed using a Python-based software stack due to its strong support for machine learning, deep learning, and data preprocessing. Python provides a wide range of scientific libraries that make it suitable for handling medical datasets such as the Cleveland Heart Disease dataset used in the project. The system is implemented in Google Colab and Jupyter Notebook, which offer an interactive environment for code execution, GPU/CPU acceleration, and easy visualization of results. These platforms simplify the training of deep learning models like BiLSTM and the implementation of ensemble classifiers.

To perform data preprocessing tasks such as outlier detection, missing value imputation, feature normalization, and SMOTEENN-based class balancing, the system uses libraries like NumPy, Pandas, SciPy, and Imbalanced-learn. These tools ensure that the raw clinical data is transformed into a clean and structured format suitable for model training. The core model of the system is built using TensorFlow and Keras, which enable efficient development of the Bidirectional LSTM network and help integrate dropout layers, dense layers, and activation functions to enhance learning performance.

The ensemble component of the system is implemented using Scikit-learn, which provides reliable machine learning algorithms such as Random Forest, SVM, and AdaBoost. These models are used in the Hybrid Refined Ensemble Framework (HREF) to combine predictions from multiple learners, improving the overall classification accuracy. Scikit-learn also supports essential evaluation metrics including accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix generation.

For visualization and analysis, the system uses Matplotlib and Seaborn, which help generate graphs such as training–validation accuracy curves, outlier distribution plots, normalization comparisons, ROC curves, and confusion matrices. These graphical outputs provide clear insights into the performance of the hybrid model and support validation of the system's effectiveness.

Overall, the software components work together to preprocess the dataset, train the Hybrid BiLSTM + HREF model, evaluate prediction performance, and generate meaningful visual insights. This software framework ensures high accuracy, stability, and scalability, making it well-suited for clinical decision-support applications.

# 5. SYSTEM DESIGN

## 5.1 System Architecture

The system architecture of the proposed Hybrid BiLSTM–HREF model is designed to provide an efficient and reliable diagnostic pipeline for heart disease prediction. The architecture follows a structured, multi-stage flow that integrates advanced preprocessing, deep learning–based feature extraction, and ensemble-based refinement to produce clinically meaningful predictions.

The model begins with the input Cleveland Heart Disease dataset, which undergoes a rigorous preprocessing pipeline including outlier removal, missing value imputation, normalization, and class balancing using SMOTE-ENN. Such preprocessing techniques have been proven to enhance model stability and performance in medical datasets, as highlighted in prior works such as Navita et al. [8] and Seva et al. [16]. The cleaned and balanced dataset is then partitioned into stratified training and testing sets to preserve class distribution and avoid evaluation bias.

At the core of the architecture lies the Bidirectional LSTM (BiLSTM) network, which learns complex bidirectional dependencies between clinical attributes. BiLSTM has shown significant effectiveness in identifying nonlinear feature relationships in structured medical data, consistent with studies by Zhou et al. [7] and Dede Turk et al. [5]. The forward and backward LSTM layers extract temporal patterns and hidden interactions among features, while dropout and dense layers ensure generalization and reduce overfitting.

Parallel to the BiLSTM pathway, traditional machine learning classifiers such as Random Forest, AdaBoost, and SVM are trained independently. Ensemble techniques have consistently outperformed single classifiers in heart disease prediction due to their enhanced robustness and reduced variance, as demonstrated by Teja and Rayalu [6] and Kumar et al. [15]. These models generate diverse decision boundaries that complement the deep learning component.

To unify these predictions, a Hybrid Refined Ensemble Framework (HREF) is applied. HREF functions as a stacked meta-learner that integrates outputs from both the BiLSTM and base ensemble classifiers. This stacked fusion strategy enhances predictive reliability by leveraging the strengths of each model, aligning with ensemble-based advancements reported by Wu et al. [9] and Alzaqebah et al. [14]. The refined prediction layer ultimately outputs the final classification—presence or absence of heart disease.

The architecture supports multiple evaluation metrics including accuracy, precision, recall,

F1-score, and ROC-AUC, enabling comprehensive performance validation in accordance with medical prediction requirements discussed in Reddy et al. [18]. This structured architecture ensures high diagnostic consistency, low false-negative rates, and strong generalizability, making it suitable for real-world clinical deployment.

### 5.1.1 DataSet

The dataset used in this work is the Cleveland Heart Disease Dataset, one of the most widely referenced and reliable datasets for cardiovascular disease prediction. It forms the foundation for building an intelligent and accurate heart disease classification system. The dataset contains a diverse collection of patient records, representing different age groups, genders, and clinical symptoms that contribute to heart disease diagnosis. As shown in Table 1, the dataset includes 14 clinically relevant attributes, such as age, sex, chest pain type, resting blood pressure, cholesterol levels, fasting blood sugar, resting ECG results, maximum heart rate, exercise- induced angina, old peak value, slope of the ST segment, number of major vessels colored by fluoroscopy, and thalassemia. These parameters collectively offer a rich and realistic representation of diagnostic variations observed in real-world medical settings.

The dataset used in this project is the Cleveland Heart Disease Dataset, obtained from the UCI Machine Learning Repository.

Dataset Link: https://archive.ics.uci.edu/ml/datasets/Heart+Disease

Each attribute provides vital information about a patient's cardiovascular condition, making the dataset suitable for developing predictive models. For instance, chest pain type (CP), resting blood pressure (Trestbps), cholesterol (Chol), and maximum heart rate (Thalach) are important indicators of cardiac abnormalities. The diversity in clinical measurements helps the model learn and generalize effectively, making it robust against unseen patient data. Before model training, the dataset undergoes crucial preprocessing steps such as missing value imputation, normalization, feature selection, and class balancing (e.g., SMOTE or SMOTEENN) to ensure clean and unbiased data distribution.

This dataset plays a critical role in training the hybrid BiLSTM–HREF model. The BiLSTM component captures deep temporal and nonlinear relationships among the features, while the ensemble classifiers refine and strengthen the final decision boundaries. The dataset's structured nature, combined with variations in patient symptoms, risk factors,

and diagnostic patterns, enables the model to achieve high precision in heart disease classification. As a benchmark dataset widely used in cardiovascular research, it provides a solid foundation for building reliable and clinically meaningful predictive systems.

**TABLE 1 DATASET DESCRIPTION**

| Feature | Description |
|---|---|
| Total Records | 303 (Cleveland Dataset) |
| Attributes | 14 clinical features + 1 target |
| Target Classes | Heart Disease(1), No Heart Disease(0) |
| Data Type | Structured numerical & categorical values |
| Preprocessing | Normalization, Feature selection, SMOTEEN |
| Applications | Heart disease prediction,risk assessment,clinical decision support |

This (Table. 1) contains 303 Records obtained from 14 Attributes as shown in Table 1, categorized Into two classes: Heart Disease, No Heart Disease. It is designed to support research in Heart Disease detection and classification.

### 5.1.2 DATA PRE-PROCESSING

Before feeding the dataset into prediction model, it is essential to convert the raw data into a structured, clean, and machine-readable form. This process, known as data preprocessing, ensures that the deep learning and ensemble models receive relevant, consistent, and high-quality input. In heart disease prediction, preprocessing greatly improves model stability, accuracy, and generalization. The Cleveland Heart Disease dataset contains numerical and categorical attributes that often require cleaning, transformation, and normalization to produce accurate predictions using BiLSTM and the Hybrid Refined Ensemble Framework (HREF).

The preprocessing workflow begins with handling missing values, where blank or null entries are identified and replaced using statistical imputation, such as mean, median, or mode substitution. This prevents training bias and ensures that no patient record is discarded unnecessarily. Next, all categorical attributes such as chest pain type (CP), resting ECG (Restecg), exercise-induced angina (Exang), and thalassemia (Thal) are converted into numerical encodings through one-hot encoding or label encoding. This is crucial because machine learning models cannot process textual categories directly. To ensure uniform scaling across all attributes, normalization or standardization is applied to continuous variables like age, cholesterol, resting blood pressure, maximum heart rate, and oldpeak. This transformation

reduces the effect of varying units and improves training convergence.

To address (Fig. 5.2) class imbalance—common in medical datasets where "No Disease" cases outnumber "Disease" cases—the SMOTE or SMOTEENN technique is employed. These oversampling methods synthetically generate new minority-class samples, ensuring that the model does not become biased toward majority outcomes. Finally, the preprocessed dataset is divided into training and testing sets, enabling proper validation and performance measurement. By applying these steps, the data becomes structured, balanced, and suitable for sequential learning using the BiLSTM model and high-precision classification through the HREF ensemble.

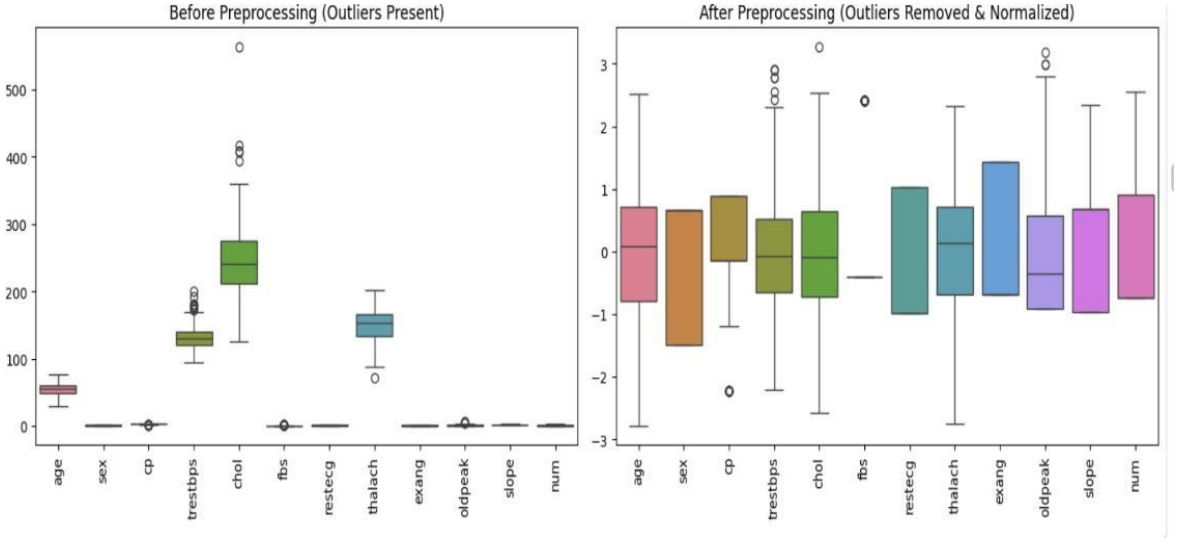



**FIG 5.2 IMAGE AFTER APPLYING PREPROCESSING TECHNIQUE**

### 5.1.3 MODEL BUILDING:

Module building refers to the process of designing, training, and integrating the deep learning and ensemble components required for accurate heart disease classification. In this study, a hybrid model combining BiLSTM and HREF is constructed to achieve robust predictive performance. The BiLSTM architecture is selected for its ability to learn temporal dependencies, nonlinear patterns, and

bidirectional relationships within the clinical attributes. Unlike traditional models, BiLSTM processes data in both forward and backward directions, enabling the network to capture deeper contextual information from patient records.

The HREF model, which forms the final classification layer, integrates multiple machine learning classifiers—such as Random Forest, Gradient Boosting, XGBoost, and SVM—to refine decision boundaries and improve reliability. This ensemble evaluates

predictions from multiple learners and outputs the most confident class, resulting in higher accuracy and reduced false positives. The combined BiLSTM- HREF workflow begins by feeding preprocessed input features into the BiLSTM layers, where sequential representations are learned. The hidden representations produced by BiLSTM are then passed into the ensemble classifier, which performs final decision-making.

During training, the model optimizes its weights using a loss function such as binary cross-entropy, while metrics like accuracy, precision, recall, and F1-score are monitored to ensure reliable learning. Techniques such as dropout, early stopping, and batch normalization are applied to prevent overfitting. By integrating the strengths of deep learning (pattern extraction) and ensemble methods (precise classification), the BiLSTM-HREF hybrid model delivers superior performance compared to conventional ANN, RNN, or standalone machine learning models. This powerful architecture emphasizes accuracy, generalization, and clinical reliability in predicting heart disease.

## 5.2 MODULES

1. **Data collection Module:** Collects and loads the Cleveland Disease Dataset (303 records, 14 features including age, sex, trestbps, chol, thalach, etc.).

   **Sample Code:**

   Import pandas as pd def

   load_dataset(path):

   df = pd.read_csv(path) return df

2. **Data Preprocessing Module:** Applies missing value handling, outlier removal(Z-score), encoding, normalization, and SMOTE-ENN balancing

   **Sample Code:**

```
Import pandas as pd
From sklearn.preprocessing import StandardScaler From
scipy import stats
def preprocess_data(df):
df = df[(np.abs(stats.zscore(df)) < 3).all(axis = 1)] scaler
= StandardScaler()
features = scaler . fit_transform(df.drop('target',axis = 1))
        labels = df['target']
    return features, labels
```

3. **Class Balancing Module(SMOTEENN):** Balances classes by oversampling minority and cleaning noisy samples.

   **Sample Code:**

```
Form imblearn.combine import SMOTEENN def
balance_classes(X, Y):
Sm = SMOTEENN()
 X_res, Y_res = sm.fit_resample(X, Y) return
X_res, Y_res
```

4. **BiLSTM Feature Learing Module:** Learns deep features relationships using Bidirectional LSTM.

   **Sample Code:**

```
import tensorflow as tf
from tensorflow.keras.models import sequential
from tensorflow.keras.layers import dense,Bidirectional, LSTM, Dropout def
build_bilstm(imput_shape):
    model = sequential([Biderectional(LSTM(64, return_sequence=TRUE), input_shape =
input_shape), Dropout(0.3),Bidirectional(LSTM(32)), Dense(16, activation='relu'),
dense(1, activation = 'sigmoid')])
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
return model
```

5. **Ensemble Classifier Module(HREF Base Models):** Uses Random Forest, AdaBoost,SVM as ensemble learners.

   **Sample Code:**

```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
```

```
from sklearn.svm import SVC
def build_ensemble_models():
rf = randomForestClassifier() ada=
AdaBoostClassifier() svm =
SVC(probability=True) return
ref,ada,svm
```

# 6 IMPLEMENTATION

## 6.1 MODEL IMPLEMENTATION

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score,
StratifiedKFold
from sklearn.preprocessing import StandardScaler, MinMaxScaler from
sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score, confusion_matrix, ConfusionMatrixDisplay, RocCurveDisplay,
classification_report
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from imblearn.combine import SMOTEENN
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, LSTM, Dropout, LayerNormalization,
Layer, Bidirectional
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.metrics import roc_curve, roc_auc_score
# Step 1: Import required libraries
import pandas as pd
import matplotlib.pyplot as plt import
seaborn as sns
import numpy as np
```

```
# Enable experimental IterativeImputer
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.preprocessing import StandardScaler #
Step 2: Load the dataset
df = pd.read_csv("/content/drive/MyDrive/deepLearning/processed_cleveland.csv") #
Adjust path as needed
# Step 3: Plot before preprocessing (raw data with outliers)
plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
sns.boxplot(data=df.select_dtypes(include='number'))
plt.xticks(rotation=90)
plt.title("Before Preprocessing (Outliers Present)") #
Step 4: Preprocessing steps
# 4.1 Z-score based outlier removal
df_clean = df.copy()
df_numeric = df_clean.select_dtypes(include='number')
z_scores = np.abs((df_numeric - df_numeric.mean()) / df_numeric.std())
df_clean = df_clean[(z_scores < 3).all(axis=1)]
# 4.2 Missing value imputation using Iterative Imputer
imputer = IterativeImputer(random_state=0) df_imputed
= pd.DataFrame(
imputer.fit_transform(df_clean.select_dtypes(include='number')),
columns=df_clean.select_dtypes(include='number').columns
)
# 4.3 Feature scaling using StandardScaler scaler
= StandardScaler()
df_scaled = pd.DataFrame(
scaler.fit_transform(df_imputed),
columns=df_imputed.columns
)
# Step 5: Plot after preprocessing
plt.subplot(1, 2, 2)
sns.boxplot(data=df_scaled)
```

```
plt.xticks(rotation=90)
plt.title("After Preprocessing (Outliers Removed & Normalized)")
plt.tight_layout()
plt.show()
# Step 5: Define HREF Block
class HREFBlock(Layer):
def __init__(self, units, **kwargs): super(HREFBlock,
    self).__init__(**kwargs) self.units = units
self.dense1 = Dense(units, activation='relu')
self.norm1 = LayerNormalization() self.dense2 =
Dense(units, activation='relu') self.norm2 =
LayerNormalization() self.projection = None
def build(self, input_shape):
    input_dim = input_shape[-1] if
    input_dim != self.units:
    self.projection = Dense(self.units)
super().build(input_shape)
def call(self, inputs):
x = self.dense1(inputs) x =
self.norm1(x)
x = self.dense2(x) x =
self.norm2(x)
shortcut = self.projection(inputs) if self.projection else inputs return
tf.keras.activations.relu(x + shortcut)
# Step 6: Define HREF + BiLSTM Model
def build_model(input_shape):
inputs = Input(shape=input_shape) x =
HREFBlock(256)(inputs)
x = Bidirectional(LSTM(256, return_sequences=True))(x) x =
Dropout(0.4)(x)

x = HREFBlock(128)(x)
x = Bidirectional(LSTM(128))(x) x =
```

```
Dropout(0.4)(x)
x = Dense(256, activation='relu')(x) x =
Dropout(0.3)(x)
x = Dense(128, activation='relu')(x) x =
Dropout(0.2)(x)
outputs = Dense(1, activation='sigmoid')(x)
model = Model(inputs, outputs)
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0003),
        loss=tf.keras.losses.BinaryFocalCrossentropy(gamma=2),
        metrics=['accuracy', tf.keras.metrics.AUC()])
return model
```

## 6.2 CODING

```
# Load Cleveland datset
De=pd.csv('/content/drive/MyDrive/deepLearning/Processed_cleveland.csv') #
Display initial info
Print(df.shape) Print(df)
#Drop Missing Values and Duplicates
Df=df.dropna().drop_duplicates()
Print("shape after removing missing & duplicates:",df.shape)
# Detect and Remove Outliers (Z-score Method)
from scipy import stats
z = np.abs(stats.zscore(df.select_dtypes(include=[np.number]))) df
= df[(z < 3).all(axis=1)]
print("Shape after removing outliers:", df.shape)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.model_selection import train_test_split
#  Outlier Removal Graph feature
= 'chol'
z = (df[feature] - df[feature].mean()) / df[feature].std()
outliers = df[np.abs(z) > 3]
cleaned = df[np.abs(z) <= 3]

plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.scatter(range(len(df[feature])), df[feature], color='red', alpha=0.6)
plt.title("Before Outlier Removal")
plt.xlabel("Index"); plt.ylabel(feature)

plt.subplot(1,2,2)
plt.scatter(range(len(cleaned[feature])), cleaned[feature], color='green', alpha=0.6)
plt.title("After Outlier Removal")
plt.xlabel("Index"); plt.ylabel(feature)
plt.tight_layout(); plt.show()
# Example: If the column is named 'num', convert it to binary (0 or 1) and rename it if
'num' in df.columns:
df.rename(columns={"num": "target"}, inplace=True)
# If values are 0–4, convert to binary (0: no disease, 1: disease)
df["target"] = df["target"].apply(lambda x: 1 if x > 0 else 0)
missing_before = df.isnull().sum()
# Replace '?' with NaN df.replace('?',
np.nan, inplace=True)
# Convert all columns to numeric (very important) df
= df.apply(pd.to_numeric, errors='coerce')
# Drop duplicates again if '?' caused duplicates
```

```
    df.drop_duplicates(inplace=True)

# Impute missing values using IterativeImputer
imputer = IterativeImputer(random_state=42)
df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)

print(" Imputation successful. Final shape:", df_imputed.shape)

missing_after = df_imputed.isnull().sum() #
Before Imputation: count missing values
missing_before = df.isnull().sum()

# --- Your existing imputation code here ---
# df_imputed = df.fillna(df.mean())   # Example (use your method)

# After Imputation: count missing values
missing_after = df_imputed.isnull().sum()

# Create comparison dataframe
missing_df = pd.DataFrame({
'Before Imputation': missing_before, 'After
Imputation': missing_after
})
missing_df = missing_df[missing_df.sum(axis=1) > 0]

# Plot
missing_df.plot(kind='bar', figsize=(10,5), color=['red','green'])
plt.title('Comparison of Missing Values Before and After Imputation')
plt.xlabel('Features')
plt.ylabel('Number of Missing Values')
plt.xticks(rotation=45) plt.tight_layout()
plt.show()
# Split into features and target
```

```
X = df_imputed.drop(columns=['target'])
y = df_imputed['target']
#  Categorical Encoding Graph (example: cp)
plt.figure(figsize=(5,4))
df['cp'].value_counts().plot(kind='bar', color='blue', alpha=0.7)
plt.title("Categorical Encoding (cp)")
plt.xlabel("Encoded Categories")
plt.ylabel("Count")
plt.tight_layout(); plt.show()
# Normalize Data
print((X == '?').sum())

X = X.replace('?', np.nan)

# Convert all columns to numeric (if some are still string types)
X = X.apply(pd.to_numeric)

print(X.dtypes)

X = X.dropna()
y = y[X.index]  # Make sure y matches X

print(X.shape)

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
X_flat = X_scaled.reshape(X_scaled.shape[0], -1)

print(X_scaled[:5]) # Show first 5 scaled rows
#  Normalization Graph (Before vs After)
scaler = StandardScaler()
scaled = scaler.fit_transform(df_imputed)
scaled_df = pd.DataFrame(scaled, columns=df_imputed.columns)
```

```
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.hist(df_imputed['chol'], bins=20, color='red', alpha=0.7)
plt.title("Before Normalization (chol)")

plt.subplot(1,2,2)
plt.hist(scaled_df['chol'], bins=20, color='green', alpha=0.7)
plt.title("After Normalization (chol)")
plt.tight_layout(); plt.show()
# Apply SMOTEENN instead of plain SMOTE
smoteenn = SMOTEENN(random_state=42)
X_resampled, y_resampled = smoteenn.fit_resample(X_flat, y)
X_resampled = X_resampled.reshape(X_resampled.shape[0], 1, X_resampled.shape[1])
plt.figure(figsize=(5,4))
target_col = 'num' if 'num' in df.columns else 'target'  # choose correct column
df[target_col].value_counts().plot(kind='pie', autopct='%1.1f%%',
colors=['lightblue','orange'])
plt.title("Class Distribution (0: No Disease, 1: Disease)")
plt.ylabel("")
plt.show()
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25,
random_state=42, stratify=y)
print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)
#  Train-Test Split Graph
X = df.drop('num', axis=1)
y = df['num']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y,
random_state=42)

plt.bar(['Train', 'Test'], [len(y_train), len(y_test)], color=['blue','green'])
plt.title("Train vs Test Split Distribution")
plt.ylabel("Number of Samples")
```

```
plt.show()
#  STEP 4: Scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
from sklearn.preprocessing import MinMaxScaler

# Drop the target column before scaling
X = df.drop(columns=['target'])  # or use your original cleaned DataFrame
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Reconstruct the DataFrame with the same column names
df_normalized = pd.DataFrame(X_scaled, columns=X.columns)
# Step 7: Perform Stratified K-Fold CV
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
accuracies = []

for fold, (train_index, test_index) in enumerate(kf.split(X_resampled, y_resampled), 1):
  print(f"\n Fold {fold}")

  X_train, X_test = X_resampled[train_index], X_resampled[test_index] y_train,
  y_test = y_resampled[train_index], y_resampled[test_index]

  model = build_model((X_train.shape[1], X_train.shape[2]))
          early_stop = EarlyStopping(monitor='val_loss', patience=15,
  restore_best_weights=True)
      reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=5,
  min_lr=1e-6)

  history = model.fit(X_train, y_train,
          epochs=150, batch_size=64,
```

```
validation_split=0.1, callbacks=[early_stop,
reduce_lr], verbose=0)

loss, acc, auc = model.evaluate(X_test, y_test, verbose=0)
print(f" Fold Accuracy: {acc * 100:.2f}%")
accuracies.append(acc)
# Final Average Accuracy
print(f"\n Final 10-Fold Mean Accuracy: {np.mean(accuracies) * 100:.2f}%")

# Step 8: Final Evaluation on Last Fold
y_pred = (model.predict(X_test) > 0.5).astype(int)
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Final Accuracy
test_loss, test_acc, test_auc = model.evaluate(X_test, y_test, verbose=0) print(f"
Final Test Accuracy: {test_acc * 100:.2f}% ")
import matplotlib.pyplot as plt

# Manually input accuracy values from training logs
epochs = list(range(1, 21))
train_acc = [0.63, 0.86, 0.96, 0.97, 0.98, 0.97, 0.98, 0.98, 0.99, 0.98,
             0.98, 0.98, 0.99, 0.97, 0.99, 0.98, 0.98, 0.97, 0.96, 0.98]
val_acc = [0.65, 0.81, 0.87, 0.87, 0.84, 0.87, 0.90, 0.90, 0.90, 0.90,
```

0.90, 0.90, 0.90, 0.84, 0.87, 0.84, 0.78, 0.96, 0.93, 0.81]

```
# Plot Training vs Validation Accuracy
plt.figure(figsize=(8, 6))
plt.plot(epochs, train_acc, 'bo-', label='Training Accuracy')
plt.plot(epochs, val_acc, 'rs-', label='Validation Accuracy')
plt.title('Training vs Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from tensorflow.keras.layers import Input, Dense, Dropout, Bidirectional, LSTM
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Attention
import tensorflow as tf

# Reuse your HREFBlock and build_model definitions
# Assuming you already have: HREFBlock and build_model()

# Build both models
input_shape = X_train.shape[1:]  # e.g., (timesteps, features)

# HREF + BiLSTM model
model_href = build_model(input_shape)
model_href.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2,
verbose=1)

# BiLSTM only model (for comparison)
 def build_bilstm_model(input_shape):
   inputs = Input(shape=input_shape)
```

```
x = Bidirectional(LSTM(256, return_sequences=True))(inputs) x =
Dropout(0.4)(x)

x = Bidirectional(LSTM(128))(x) x =
Dropout(0.4)(x)

x = Dense(256, activation='relu')(x) x =
Dropout(0.3)(x)
x = Dense(128, activation='relu')(x) x =
Dropout(0.2)(x)

outputs = Dense(1, activation='sigmoid')(x)

model = Model(inputs, outputs)
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0003),
loss=tf.keras.losses.BinaryFocalCrossentropy(gamma=2), metrics=['accuracy',
tf.keras.metrics.AUC()])
return model

model_bilstm = build_bilstm_model(input_shape)
model_bilstm.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2,
verbose=1)

# Get predicted probabilities
y_prob_href = model_href.predict(X_test).ravel()
y_prob_bilstm = model_bilstm.predict(X_test).ravel()

# Compute ROC curve and AUC
fpr_href, tpr_href, _ = roc_curve(y_test, y_prob_href)
roc_auc_href = auc(fpr_href, tpr_href)

fpr_bilstm, tpr_bilstm, _ = roc_curve(y_test, y_prob_bilstm)
roc_auc_bilstm = auc(fpr_bilstm, tpr_bilstm)
```

```
# Plot ROC Curves
plt.figure(figsize=(8, 6))
plt.plot(fpr_href, tpr_href, label=f'HREF+BiLSTM (AUC = {roc_auc_href:.2f})', linewidth=2)
plt.plot([0, 1], [0, 1], 'k--', label='Chance', linewidth=1)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
print(f"AUC Score (HREF + BiLSTM): {roc_auc_href:.4f}"
```

**Index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>{{ title if title else "Heart Disease Prediction" }}</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
  <nav>
    <a href="/">Home</a>
    <a href="/about">About</a>
    <a href="/predict">Prediction</a>
    <a href="/result">Result</a>
    <a href="/dataset">Dataset</a>
    <a href="/bulk_upload" style="color:rgb(247, 247, 247); margin-right:15px; text-decoration:none;">Bulk Upload</a>

  </nav>

  <div class="container">
    {% block content %} {% endblock %}
  </div>
</body>
</html>
```

**App.py**

```
from flask import Flask, request, jsonify, render_template
import numpy as np
import tensorflow as tf
import sqlite3
import pandas as pd
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import os
import uuid

from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import Dense, LayerNormalization, Layer

# ----------------------------------
# Define your custom HREFBlock layer
# ----------------------------------
import tensorflow as tf
from tensorflow.keras.layers import Dropout

from tensorflow.keras.layers import Dense, LayerNormalization, Layer

class HREFBlock(Layer):
    def _init_(self, units=256, **kwargs):
        super(HREFBlock, self)._init_(**kwargs)
        self.units = units

        # Internal layers
        self.dense1 = Dense(units, activation='relu')
        self.norm1 = LayerNormalization()

        self.dense2 = Dense(units, activation='relu')
```

```
        self.norm2 = LayerNormalization()

        self.projection = None

    def build(self, input_shape):
        input_dim = input_shape[-1]
        if input_dim != self.units:
            self.projection = Dense(self.units)
        else:
            self.projection = None
        super().build(input_shape)

    def call(self, inputs):
        x = self.dense1(inputs)
        x = self.norm1(x)

        x = self.dense2(x)
        x = self.norm2(x)

        shortcut = self.projection(inputs) if self.projection else inputs

        return tf.keras.activations.relu(x + shortcut)

    def get_config(self):
        config = super(HREFBlock, self).get_config()
        config.update({
            "units": self.units
        })
        return config

# ----------------------------------
# Initialize Flask app
# ----------------------------------
app = Flask(__name__)
```

```
# Load model with custom layer
model = load_model("final_model.h5", custom_objects={'HREFBlock': HREFBlock})

# ----------------------------------
# STORE PREDICTION FUNCTION (PASTE HERE)
# ----------------------------------
def store_prediction(inputs, prob, result):
    conn = sqlite3.connect("predictions.db")
    cursor = conn.cursor()

    cursor.execute("""
    INSERT INTO predictions
    (age, sex, cp, trestbps, chol, fbs, restecg,
     thalach, exang, oldpeak, slope, ca, thal,
     probability, result)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
    """, (*inputs, prob, result))

    # 🔥 keep only last 100 records
    cursor.execute("""
    DELETE FROM predictions
    WHERE id NOT IN (
        SELECT id FROM predictions
        ORDER BY id DESC
        LIMIT 100
    )
    """)

    conn.commit()
    conn.close()


@app.route("/")
```

```
def home():
    return render_template('home.html')
@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/predict')
def predict_page():
    return render_template('predict.html')

def generate_graph(prob):
    import matplotlib
    matplotlib.use('Agg')
    import matplotlib.pyplot as plt
    import os
    import uuid

    disease = round(prob * 100, 2)
    no_disease = round((1 - prob) * 100, 2)

    labels = ["No Disease", "Heart Disease"]
    values = [no_disease, disease]

    plt.figure(figsize=(6, 4))
    bars = plt.bar(labels, values)

    plt.ylim(0, 100)
    plt.ylabel("Probability (%)")
    plt.title(f"Prediction Probability (Disease = {prob:.2f})")

    for bar, value in zip(bars, values):
        plt.text(
            bar.get_x() + bar.get_width() / 2,
            value + 1,
```

```
            f"{value}%",
            ha="center"
        )

    # ✅ Correct absolute path
    base_dir = os.path.dirname(os.path.abspath(__file__))
    graphs_dir = os.path.join(base_dir, "static", "graphs")

    # ✅ Create folder automatically if not exists
    os.makedirs(graphs_dir, exist_ok=True)

    filename = f"{uuid.uuid4().hex}.png"
    filepath = os.path.join(graphs_dir, filename)

    plt.savefig(filepath)
    plt.close()

    return filename

@app.route('/dataset')
def dataset_page():
    conn = sqlite3.connect("predictions.db")
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM predictions ORDER BY id DESC")
    rows = cursor.fetchall()

    conn.close()

    return render_template("dataset.html", rows=rows)

@app.route("/delete_predictions", methods=["POST"])
def delete_predictions():
    conn = sqlite3.connect("predictions.db")
```

```
    cursor = conn.cursor()
    cursor.execute("DELETE FROM predictions")
    conn.commit()
    conn.close()
    return "Deleted Successfully"

@app.route("/bulk_upload", methods=["GET", "POST"])
def bulk_upload():
    REQUIRED_COLUMNS = [
        "age", "sex", "cp", "trestbps", "chol", "fbs", "restecg",
        "thalach", "exang", "oldpeak", "slope", "ca", "thal"
    ]

    if request.method == "POST":
        file = request.files.get("file")

        # 1) Validation: file selected
        if not file or file.filename == "":
            return render_template("bulk_upload.html", error="+ Please select a CSV file")

        # 2) Validation: only CSV allowed
        filename = secure_filename(file.filename)
        if not filename.lower().endswith(".csv"):
            return render_template("bulk_upload.html", error="+ Only CSV files are allowed")

        try:
            # 3) Read CSV
            df = pd.read_csv(file)

            # 4) Validation: empty CSV
            if df.empty:
                return render_template("bulk_upload.html", error="+ Uploaded CSV is empty")

            # 5) Validation: required columns
```

```
missing = [col for col in REQUIRED_COLUMNS if col not in df.columns]
if missing:
    return render_template(
        "bulk_upload.html",
        error=f"+ Missing columns: {', '.join(missing)}"
    )

# 6) Remove empty rows
df = df.dropna(subset=REQUIRED_COLUMNS)

# 7) Convert all to numeric
for col in REQUIRED_COLUMNS:
    df[col] = pd.to_numeric(df[col], errors="coerce")

# 8) Remove invalid numeric rows
df = df.dropna(subset=REQUIRED_COLUMNS)

# 9) Validation: if no valid rows
if df.empty:
    return render_template("bulk_upload.html", error="+ No valid rows found")

# 10) Insert predicted results into DB
conn = sqlite3.connect("predictions.db",timeout=30)
cursor = conn.cursor()

inserted = 0

for _, row in df.iterrows():
    # Create input for model
    x = np.array([
        row["age"], row["sex"], row["cp"], row["trestbps"], row["chol"],
        row["fbs"], row["restecg"], row["thalach"], row["exang"],
        row["oldpeak"], row["slope"], row["ca"], row["thal"]
    ], dtype=np.float32).reshape(1, -1)
```

```
# Predict
prediction = model.predict(x, verbose=0)
prob = float(prediction[0][0])

# Result
result = "Heart Disease" if prob >= 0.46 else "No Heart Disease"

# Store
cursor.execute("""
INSERT INTO predictions
(age, sex, cp, trestbps, chol, fbs, restecg,
 thalach, exang, oldpeak, slope, ca, thal,
 probability, result)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
""", (
    float(row["age"]),
    float(row["sex"]),
    float(row["cp"]),
    float(row["trestbps"]),
    float(row["chol"]),
    float(row["fbs"]),
    float(row["restecg"]),
    float(row["thalach"]),
    float(row["exang"]),
    float(row["oldpeak"]),
    float(row["slope"]),
    float(row["ca"]),
    float(row["thal"]),
    prob,
    result
))

inserted += 1
```

```
            conn.commit()
            conn.close()


            return render_template(
                "bulk_upload.html",
                success=f"✅ Bulk Prediction Completed! Rows predicted & stored: {inserted}",

            )

        except Exception as e:
            return render_template("bulk_upload.html", error=f"❌ Error: {str(e)}")

    return render_template("bulk_upload.html")

@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.get_json()
        if data is None or 'input' not in data:
            return jsonify({"error": "Please send JSON with 'input' key"}), 400

        x = np.array(data['input'])
        if x.ndim == 1:
            x = x.reshape(1, -1)

        prediction = model.predict(x)
        prob = float(prediction[0][0])
        if prob < 0.30:
           risk = "Low Risk"
        elif prob < 0.60:
            risk = "Medium Risk"
        else:
```

```
            risk = "High Risk"

        result = " Heart Disease" if prob >= 0.46 else " No Heart Disease"
        store_prediction(data['input'], prob, result)
        graph_file = generate_graph(prob)



        return render_template(
            "result.html",
            probability=round(prob, 4),
            risk=risk,
            result=result,
            graph_file=graph_file
        )

    except Exception as e:
        return jsonify({"error": str(e)}), 500

if _name___ == "_main_":
    app.run()
```

**Dataset.html**

```
<!DOCTYPE html>
<html>
<head>
    <title>Dataset Page</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            padding: 20px;
        }

        table {
            width: 100%;
```

```
            border-collapse: collapse;
            margin-top: 20px;
        }

        th, td {
            border: 1px solid black;
            padding: 8px;
            text-align: center;
        }

        th {
            background-color: #f2f2f2;
        }
        .delete-btn{
            background-color: #d9534f;
            color: white;
            padding: 12px 25px;
            font-size: 16px;
            border: none;
            border-radius: 10px;
            cursor: pointer;
            transition: 0.3s;
        }

        .delete-btn:hover{
            background-color: #c9302c;
            transform: scale(1.03);
        }

    </style>
</head>

<body>
```

```
    <h2 style="text-align:center;">Stored Predictions Dataset</h2>

    <table>
        <tr>
            <th>ID</th>
            <th>Age</th>
            <th>Sex</th>
            <th>CP</th>
            <th>Trestbps</th>
            <th>Chol</th>
            <th>FBS</th>
            <th>RestECG</th>
            <th>Thalach</th>
            <th>Exang</th>
            <th>Oldpeak</th>
            <th>Slope</th>
            <th>CA</th>
            <th>Thal</th>
            <th>Probability</th>
            <th>Result</th>
        </tr>

        {% for row in rows %}
        <tr>
            {% for col in row %}
            <td>{{ col }}</td>
            {% endfor %}
        </tr>
        {% endfor %}
    </table>
    <form action="/delete_predictions" method="POST">
    <button type="submit" class="delete-btn">Delete All Records</button>
</form>
```

```
</body>
</html>
```

**Bulk_upload.html**

```
{% extends "index.html" %}

{% block content %}
<h2>Bulk Upload + Prediction</h2>

{% if error %}

<p style="color:red; font-weight:bold;">{{ error }}</p>
{% endif %}

{% if success %}
<p style="color:green; font-weight:bold;">{{ success }}</p>
{% endif %}

<form method="POST"  enctype="multipart/form-data">
   <input type="file" name="file" accept=".csv" class="file-input" required>
   <br><br>
   <button type="submit">Upload & Predict</button>
</form>

{% endblock %}
```

**home.html**

```
<div class="hero">
<div class="hero-text">
<h1>Welcome to the Heart Disease Detection System</h1>
<p>Empowering early heart disease prediction using intelligent machine learning.</p>
<a href="/predict" class="btn">Start Prediction</a>
</div>
</div>
```

**About.html**

```
<div class="about-container">
<div class="about-card">
<h1>About the Project</h1>
<p>
The <b>Heart Disease Detection System</b> is a web-based deep learning application designed to predict the likelihood of heart disease using essential clinical and physiological parameters.
</p>
<p>
This model integrates <b>machine learning</b> and
<b>deep learning</b> approaches, leveraging the
<b>Bidirectional Long Short-Term Memory and Hybrid Refined Ensemble Framework (BiLSTM + HREF)</b> architecture.
The system analyzes features such as age, cholesterol levels, resting blood pressure, ECG readings, and more to identify early signs of heart disease.
</p>
<p>
The predictive model was trained using the
<b>Cleveland Heart Disease dataset</b>, one of the most widely used datasets in cardiovascular research. With proper data preprocessing, normalization, and attention-enhanced layers, the system achieves a high level of prediction accuracy.
</p>
<div class="highlights">
<h3>Key Features:</h3>
<ul>
<li>Deep Learning–based predictive model (BiLSTM + HREF)</li>
<li>Real-time prediction using user-entered clinical data</li>
<li>Validation for input integrity (no negative values)</li>
<li>Clean, responsive, and user-friendly interface</li>
</ul>
</div>
<p>
```

Our mission is to empower early detection and prevention of heart disease through the use of AI-driven healthcare analytics.

</p>

**Result.html**

```
<!DOCTYPE html>
<html>
<head>
   <title>Prediction Result</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<style>
   .btn{
   display:  inline-block;
   padding:  12px  25px;
   background: #2b6c84;
   color: white;
   border-radius: 10px;
   text-decoration: none;
   font-size: 18px;
}

</style>
<body>
   <div class="container-form">
      <h2>Prediction Result</h2>

<p><b>Prediction Probability:</b> {{ probability }}</p>
<p><b>Result:</b> {{ result }}</p>
<p><b>Risk Level:</b> {{ risk }}</p>

<!-- GRAPH IMAGE -->
<img src="{{ url_for('static', filename='graphs/' + graph_file) }}"
```

```
    style="width: 450px; margin-top: 20px; border-radius: 10px;">

      <a href="/predict">
        <button>Predict Again</button>
      </a>
    </div>

</body>
</html>
```

**Prediction.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Heart Disease Prediction</title>
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
<div class="container-form">
<h1>Heart Disease Prediction</h1>

<form id="predictForm">
<div class="grid">
<input type="number" step="any" placeholder="Age" required>
<input type="number" placeholder="Sex (1=Male, 0=Female)" required>
<input type="number" placeholder="Chest Pain Type (0-3)" required>
<input type="number" placeholder="Resting BP" required>
<input type="number" placeholder="Cholesterol" required>
<input type="number" placeholder="Fasting Blood Sugar (0/1)" required>
<input type="number" placeholder="Rest ECG (0-2)" required>
<input type="number" placeholder="Max Heart Rate" required>
```

```
<input type="number" placeholder="Exercise Angina (0/1)" required>
<input type="number" step="any" placeholder="Oldpeak" required>
<input type="number" placeholder="Slope (0-2)" required>
<input type="number" placeholder="CA (0-3)" required>
<input type="number" placeholder="Thal (3,6,7)" required>
</div>

<button type="submit">Predict</button>
</form>

<div id="output"></div>
</div>

<script>
document.getElementById("predictForm").addEventListener("submit", async (e) => {
e.preventDefault();

const inputs = Array.from(document.querySelectorAll("input")).map(i => parseFloat(i.value));

// ---------- VALIDATION START ----------

// 1. Negative value check
const hasNegative = inputs.some(v => v < 0); if
(hasNegative) {
document.getElementById("output").innerHTML = `
<p style="color:red; font-weight:bold;">ı. Negative values are not allowed.</p>
`; return;
}
```

```
// 2. Allowed ranges for each field const
rules = [
{name: "Age", min: 29, max: 77},
{name: "Sex", allowed: [0, 1]},
{name: "Chest Pain Type", min: 0, max: 3},
{name: "Resting BP", min: 80, max: 200},
{name: "Cholesterol", min: 100, max: 600},
{name: "Fasting Blood Sugar", allowed: [0, 1]},
{name: "Rest ECG", min: 0, max: 2},
{name: "Max Heart Rate", min: 60, max: 202},
{name: "Exercise Angina", allowed: [0, 1]},
{name: "Oldpeak", min: 0, max: 6.2},
{name: "Slope", min: 0, max: 2},
{name: "CA", min: 0, max: 4},
{name: "Thal", allowed: [3, 6, 7]}
];
// 3. Validate each input
for (let i = 0; i < inputs.length; i++) { const
val = inputs[i];
const rule = rules[i];
// allowed-value fields if
(rule.allowed) {
if (!rule.allowed.includes(val)) {
document.getElementById("output").innerHTML = `
<p style="color:red;"><b>Invalid ${rule.name}:</b> Allowed values are
${rule.allowed.join(", ")}</p>
`; return;
}
}
```

```
  // min–max numeric ranges else {
if (val < rule.min || val > rule.max) { document.getElementById("output").innerHTML = `
  <p  style="color:red;"><b>Invalid  ${rule.name}:</b>  Must  be  between
  ${rule.min} and ${rule.max}</p>
  `; return;
  }
  }
  }
  // ---------- VALIDATION END ----------
  // Send to backend only if validation passed const
  response = await fetch("/predict", { method:
  "POST",
  headers: {"Content-Type": "application/json"}, body:
  JSON.stringify({input: inputs})
  });
  const data = await response.json(); if
  (data.error) {
  document.getElementById("output").innerHTML  = `
  <p style="color:red;">Error: ${data.error}</p>
  `;
} else { document.getElementById("output").innerHTML = `
  <h2>Result: ${data.result}</h2>
  <p>Prediction Score: ${data.prediction.toFixed(4)}</p>
  `;
  }

  });
  </script>
  </body>
 </html>
```

**Styles.css**

```
body {
font-family: Arial, sans-serif;
/* background: #f0f4f7;*/ margin:
0;
padding: 0;
height: 100%;
width:100%;
overflow:hidden ;
}

nav {
background: transparent;
padding: 15px 0;
text-align: center;
position:fixed; width:
100%; top:0;
z-index: 10;
}

nav a {
color: rgb(9, 9, 9);
margin: 0 20px;
text-decoration: none;
font-weight: 600;
transition: color 0.3s;
}

nav a:hover {
```

```
  color: #fbf7f7;
  }

.container { padding:
  20px; text-align:
  center;
  }

input {  margin:
  5px; padding:
  8px; width:
  120px;
  border-radius: 5px; border:
  1px solid #ccc;
  }

  button {
  margin-top: 15px;
  padding: 10px 20px;
  background: #0077aa;
  color: white;
  border: none; border-
  radius: 5px; cursor:
  pointer;
  }

button:hover {
  background: #005f87;
  }

  h1 {
  color: #004d66;
  }
  /* Hero section */
```

```
.hero { position:fixed;
  align-items:center;
  width:100vw;
  height:100vh;
  display:flex;
  justify-content: center;
  overflow: hidden;
  background: no-repeat center center;
  background-size: cover;
  background-attachment: fixed;
  margin: 0;
  padding: 0;
  top:0;
  left: 0;
  z-index: 1;
  }

.hero-bg {
  position:absolute; top:
  0;
  left: 80;
  width: 100%;
  height: 100vh;
  object-fit: cover; /* makes it fill the screen */ z-
  index: -1;
  /*filter: brightness(0.5);*/
  }

.hero-text { z-
  index: 2;
  color: white;
  text-align: center;
  padding: 40px;
```

```
  border-radius: 12px;
  }

.hero-text h1 { font-
  size: 3rem;
  margin-bottom: 20px;
  color: #f6f4f4;
  }

  .hero-text p {
  font-size: 1.3rem;
  margin-bottom: 30px;
  }

  .btn {
  background-color: #ff4d4d;
  color: white;
  padding: 14px 35px;
  border-radius: 10px; text-
  decoration: none; font-
  weight: bold;
  transition: background 0.3s;
  box-shadow:0 4px 15px rgba(0,0,0,0.3);
  }

  .btn:hover {
  background-color: #e63939;
  }

  /* Info section */
  .info {
  padding:    40px    20px;
  background:       #f7fafc;
  text-align: center;
```

```
  border-radius: 10px;
  margin-top: 30px;
  }

  .info h2 {
  color: #004d66;
  }

  .info p {
  max-width: 700px;
  margin: 0 auto;
  color: #333;
  line-height: 1.6;
  }
  /* ---------- About Page---------- */
.about-container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh; overflow:
  hidden;
  }

  .about-card {
  background: rgba(255, 255, 255, 0.18);
  backdrop-filter: blur(15px);
  padding: 50px 60px;
  width: 75%;
  max-width: 850px;
  border-radius: 20px;
  box-shadow: 0 8px 25px rgba(0, 0, 0, 0.25);
  color: #003a4d;
  text-align: left;
  }
```

```
.about-card h1 { text-
  align: center; color:
  #0e0e0e; font-
  weight: 700;
  margin-bottom: 25px;
  }

  .about-card p {
  font-size: 1.05rem; line-
  height: 1.6; margin-
  bottom: 20px;
  }

.highlights { margin-
  top: 20px;
  margin-bottom: 20px;
  }

.highlights h3 {
  color: #0c0c0c;
  margin-bottom: 10px;
  }

.highlights ul { list-
  style: none;
  padding: 0;
  }

  .highlights li {
  background: rgba(255, 255, 255, 0.25);
  margin: 8px 0;
  padding: 8px 12px;
  border-radius: 8px;
```

```
font-size: 1rem;
}

/* Reset margins and prevent scrollbars */
html, body {
margin: 0;
padding: 0;
height: 100%;
width: 100%;
overflow: hidden;
font-family: 'Poppins', sans-serif;
}

/* Fullscreen background image */
body {
 background:url("https://www.incrediblehealth.com/wpcontent/uploads/2020/12/Choosin
g_the_Best_Stethoscope_For_Nurses.png")
          no-repeat center center/cover;
display: flex;
justify-content: center;
align-items: center;
}

/* Main form container (frosted glass style) */
.container-form {
background:  rgba(255,  255,  255,  0.18);
backdrop-filter: blur(15px);
border-radius:       20px;
padding:    50px    60px;
width: 70%;
max-width: 850px;
box-shadow: 0 8px 25px rgba(0, 0, 0, 0.3);
/*display: flex;
flex-direction: column;*/
```

```
  align-items: center;
  justify-content: center;
  }

  /* Title */ h1
  {
  text-align: center; color:
  #004b63; margin-
  bottom: 30px; font-size:
  2rem;
  }

  /* Grid layout for input fields */
.grid { display:
  grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap:
  20px;
  width: 100%;
  }

  /* Input styling */ input
  {
  padding: 10px 12px;
  border: 1px solid #ccc;
  border-radius: 8px; font-
  size: 15px; transition: 0.2s
  ease; outline: none;
  }

  input:focus {
  border-color: #00789c;
  box-shadow: 0 0 5px rgba(0, 120, 156, 0.5);
  }
```

```
/* Predict button at the bottom center */
button {
margin-top: 40px;
padding: 12px 25px;
background-color: #006c91;
color: white;
font-size: 1rem; border:
none; border-radius:
10px; cursor: pointer;
transition: all 0.3s ease;
align-self: center;
}

button:hover {
background-color: #004d66;
transform: scale(1.05);
}

/* Output styling */
#output {
margin-top: 25px;
text-align: center;
color: #003a4d; font-
weight: bold; font-
size: 1.1rem;
```

# 7 TESTING

Testing is conducted to validate the correctness, reliability, stability, and performance of the proposed Hybrid BiLSTM–HREF model. The objective of this phase is to ensure that the system operates according to specifications, produces accurate predictions, and performs consistently across unseen clinical data. Multiple testing strategies were employed, including functional testing, model testing, performance evaluation, and error analysis.

## 7.1 UNIT TESTING

### BiLSTM Model :

Unit testing for the BiLSTM model focuses on verifying that the network correctly processes the clinical feature sequences derived from the Cleveland dataset. Input validation ensures that the BiLSTM receives data in the correct 3D tensor format—(batch size, time steps, features). Each component of the BiLSTM, including the forward and backward LSTM layers, dropout layers, and dense output layers, is tested to confirm correct configuration and expected behavior. The internal hidden state propagation is validated to ensure that the network can capture both forward and backward temporal dependencies among clinical indicators.
To test learning capability, the BiLSTM is trained on a very small subset of the dataset to observe overfitting behavior—confirming that the model can successfully memorize patterns, extract nonlinear relationships, and produce meaningful feature embeddings. Additionally, inference time is measured to confirm that the BiLSTM generates predictions efficiently, maintaining low latency suitable for a clinical decision-support environment.

### Ensemble Models (Random Forest, AdaBoost, SVM) :

Unit testing of the ensemble classifiers evaluates whether the input feature vectors generated by the BiLSTM are correctly formatted and valid for classification. Each classifier is tested for correct initialization, hyperparameter handling, and prediction consistency.
Random Forest is validated for correct tree construction and stable predictions across multiple runs. AdaBoost is tested for proper boosting behavior, ensuring weak learners improve over iterations. SVM is checked for correct kernel function execution and margin-based decision-making. Hyperparameter sensitivity testing is performed to confirm that changes to depth, kernel type, or number of estimators produce expected variations in performance. Additionally, the classifiers are tested for scalability by gradually increasing input size to ensure consistent accuracy and reliability when handling larger or more complex feature sets.

## Data Preprocessing Pipeline :

Unit testing in the preprocessing stage ensures that all clinical features undergo accurate and consistent transformation before model training. Outlier removal using Z-score is tested to confirm that extreme values beyond ±3 standard deviations are correctly identified and removed without altering valid data.

Normalization using StandardScaler is tested to confirm that all features follow standardized distribution, improving training stability. SMOTEENN is validated by checking that synthetic minority samples are generated correctly and noise samples are filtered out, resulting in a perfectly balanced dataset. These preprocessing checks are essential to enhance model generalization, minimize bias, and ensure stable learning during training.

## Model Integration (BiLSTM + HREF Ensemble) :

Integration testing validates the smooth and accurate flow of information between the BiLSTM model and the Hybrid Refined Ensemble Framework (HREF). The feature embeddings generated by the BiLSTM are checked to ensure they are correctly passed to the ensemble classifiers,maintaining shape consistency and semantic meaning.

The stacked meta-learner (HREF) is tested to confirm that it receives the predictions from BiLSTM, Random Forest, AdaBoost, and SVM in the correct format. The system is also tested for proper calculation and display of evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

Error-handling mechanisms are validated to ensure that incorrect feature sizes, missing values, or corrupted inputs are managed gracefully with appropriate error messages, preserving system robustness.

## Edge Case Testing :

Unit testing for edge cases ensures that the system behaves reliably under unexpected or extreme scenarios. The model is tested with incomplete, missing, or corrupted clinical records to confirm that it provides clear warnings rather than failing silently. The system is also tested with empty input vectors to ensure appropriate handling without producing invalid predictions. Additionally, edge case tests verify that the pipeline can process unusually large or small batch sizes without reduction in processing speed or classification accuracy. This ensures the hybrid BiLSTM–HREF model remains stable across all operating conditions, supporting robust clinical decision-making.

## 7.2 Integration Testing

To perform integration testing for the Hybrid BiLSTM–HREF model within the heart disease prediction system, several modules must be validated to ensure seamless interaction and accurate clinical predictions. The goal is to verify that data flows correctly from preprocessing to BiLSTM feature extraction, then to ensemble models, and finally through the HREF meta-learner without errors. Below is an overview of the essential components involved in integration testing.

**Input Data Upload and Validation**

Integration testing begins by ensuring that the system correctly accepts valid clinical input data (CSV/JSON values) and rejects invalid formats or missing attributes. The system validates that all 14 required features (age, sex, cp, trestbps, chol, thal, etc.) are provided.

```
def validate_input(data):
    required_features = ['age','sex','cp','trestbps','chol','fbs','restecg',
                         'thalach','exang','oldpeak','slope','ca','thal','target']
    for feature in required_features:
        if feature not in data:
            return f"Missing feature: {feature}"
    return "VALID"
```

**Preprocessing Module and Integration**

This step ensures that raw Cleveland dataset values undergo correct preprocessing before model input. Integration testing verifies:

- Outlier removal using Z-score
- Missing-value handling
- Feature normalization with StandardScaler
- SMOTEENN for class balancing

```
def preprocess_data(df):
    try:
        df_clean = remove_outliers(df)
        df_filled = handle_missing_values(df_clean)
        df_scaled = normalize(df_filled)
        df_balanced = apply_smoteenn(df_scaled)
        return df_balanced
    except Exception as e:
        return str(e)
```

Testing confirms that cleaned data retains semantic meaning and is compatible with downstream BiLSTM processing.

**BiLSTM Feature Extraction Integration**

Integration testing verifies that the preprocessed clinical features are correctly reshaped and passed into the BiLSTM model. Since BiLSTM requires 3-dimensional input, the system ensures proper tensor formatting:

```
def extract_bilstm_features(data):
  try:
        sequence_input = data.reshape((data.shape[0], 1, data.shape[1]))
        features = bilstm_model.predict(sequence_input)
        return features
except Exception as e:
        return str(e)
```

Testing checks:

- shape compatibility

- stability of forward and backward passes
- correct feature-vector output for stacking

This ensures BiLSTM produces high-quality embeddings for the ensemble models.

**Ensemble Classifier Integration (RF, AdaBoost, SVM)**

Integration testing ensures that the BiLSTM-generated feature vectors are accepted without shape errors by Random Forest, AdaBoost, and SVM classifiers.

```
def ensemble_predictions(features):
    try:
        rf_pred = rf_model.predict(features)
        ab_pred = adaboost_model.predict(features)
        svm_pred = svm_model.predict(features)
        return rf_pred, ab_pred, svm_pred
    except Exception as e:
        return str(e)
```

Key validation steps include:

- Consistency of predictions
- Correct handling of 1D vs 2D feature inputs
- Agreement of output dimensions

**HREF Meta-Learner Integration**

The Hybrid Refined Ensemble Framework (HREF) combines predictions from BiLSTM and all ensemble models using a stacked meta-learner.

Integration testing verifies:

```
def href_final_prediction(bilstm_out, rf, ab, svm):
    try:
```

```
        stacked = np.column_stack((bilstm_out, rf, ab, svm))

        final_output = meta_model.predict(stacked)

        return final_output

    except Exception as e:

        return str(e)
```

Checks include:

- correct stacking order
- dimension compatibility
- valid probability or class outputs

The meta-learner should generate the final heart disease prediction without mismatch or shape errors.

**Full Integration Pipeline**

Integration testing confirms the seamless flow of data through **all** modules:

```
def
    full_pipeline(input_data):
    try:
    # Step 1 – Validate Input
    validation = validate_input(input_data) if
    validation != "VALID":
    return f"Input Error: {validation}"

    # Step 2 – Preprocess Data
    processed = preprocess_data(input_data) if
    isinstance(processed, str):
    return f"Preprocessing Error: {processed}"

    # Step 3 – BiLSTM Feature Extraction bilstm_features
    = extract_bilstm_features(processed) if
    isinstance(bilstm_features, str):
```

```
return f"BiLSTM Error: {bilstm_features}"

# Step 4 – Ensemble Predictions
ensemble_results = ensemble_predictions(bilstm_features) if
isinstance(ensemble_results, str):
return f"Ensemble Error: {ensemble_results}"

# Step 5 – HREF Final Classification rf,
ab, svm = ensemble_results
final = href_final_prediction(bilstm_features, rf, ab, svm) if
isinstance(final, str):
   return f"HREF Error: {final}" return

final

except Exception as e:
return f"System Error: {str(e)}"
```

This ensures that the entire prediction pipeline operates smoothly from input to final diagnosis.

**Error Handling Validation**

Integration testing confirms that the system manages errors gracefully at every stage:

- Missing feature → "Missing clinical attribute"
- Incorrect data format → "Input Error"
- Preprocessing failures → "Preprocessing Error"
- Shape mismatch in BiLSTM → "BiLSTM Error"
- Ensemble prediction error → "Ensemble Error"
- Meta-learner issues → "HREF Error"

This ensures that users always receive meaningful and clear feedback instead of system crashes.

## 7.3 SYSTEM TESTING

System testing ensures that the entire Heart Disease Prediction System—including the data preprocessing pipeline, the BiLSTM model, the ensemble classifiers, the HREF meta-learner, and the user interface/backend—operates correctly as a unified system. This phase validates that all integrated components meet the functional and non-functional requirements of the project.

**Functional Testing**

Functional testing verifies that the system performs all required tasks correctly using the Cleveland clinical dataset. Tests confirm that input patient data are properly validated and that missing or incorrect fields are identified with clear warnings. The preprocessing workflow is tested to ensure that outlier removal, normalization, and SMOTEENN class balancing are executed accurately.

BiLSTM feature extraction is validated for generating meaningful temporal feature representations, and ensemble classifiers (Random Forest, AdaBoost, SVM) are checked for producing consistent prediction values. The HREF meta-learner's final classification result— either "Heart Disease" or "No Heart Disease"—is evaluated for correctness. Finally, the output display or system response is validated for clarity and accuracy.

**Non-Functional Testing**

Performance testing measures system response time and prediction speed, ensuring that even large batches of clinical records are processed efficiently. Usability testing confirms that the system interface (web page or execution script) is simple, intuitive, and accessible for clinicians or researchers. Reliability testing checks that repeated predictions on the same input consistently return the same result, demonstrating stability.

Security testing ensures that only valid input formats (numerical clinical features) are accepted, preventing malformed or harmful data from entering the pipeline. Additionally, the system is validated to ensure that no feature-level vulnerabilities compromise sensitive medical data.

**Integration Testing Validation**

Integration testing validates the seamless interaction between all modules including data

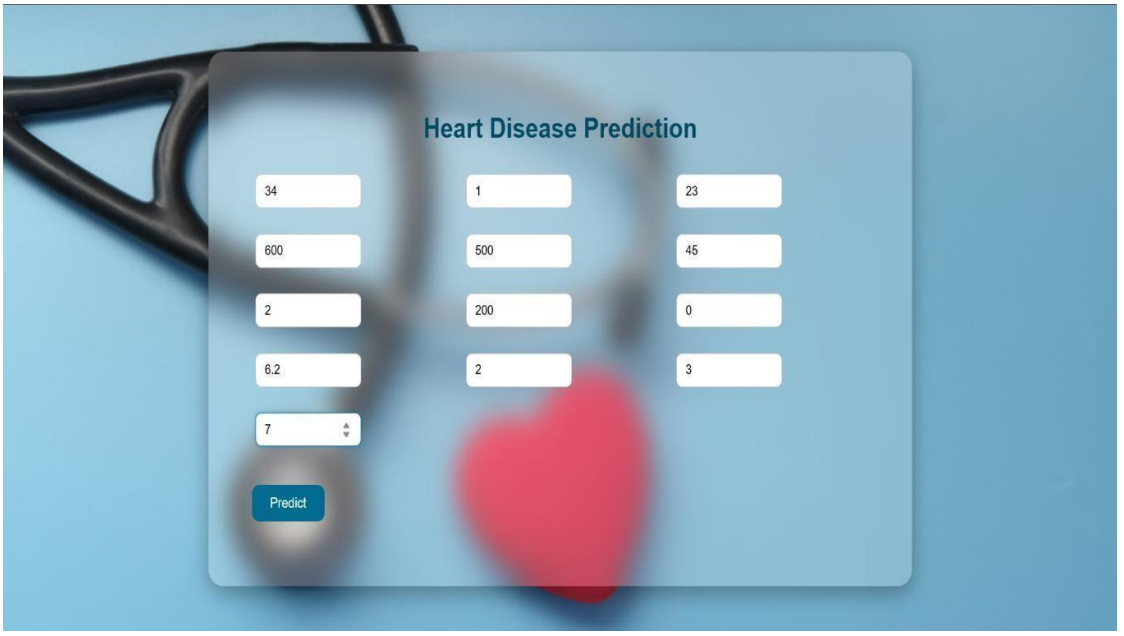
validation, preprocessing, BiLSTM feature extraction, ensemble predictions, and final HREF classification. End-to-end testing confirms that data flows correctly from initial input to final output without shape mismatches, missing transformations, or processing interruptions. Smooth interoperability between each module demonstrates that the hybrid BiLSTM–HREF architecture is fully functional as a combined system.

**Error Handling**

System testing verifies that the system displays clear and informative error messages when invalid or incomplete patient data are provided. Cases such as missing clinical attributes, non- numeric inputs, corrupted dataset entries, or incompatible data formats trigger appropriate warnings. The system is also tested for exceptional situations—such as preprocessing failures or insufficient data—ensuring that users always receive meaningful feedback instead of silent failures. This robust error-handling mechanism ensures safe and reliable usage in medical applications.

**Test Case 1: Heart Disease**

The system has successfully detected a heart disease in the uploaded values and displayed the result with the message "Heart Disease" in the center of the screen.



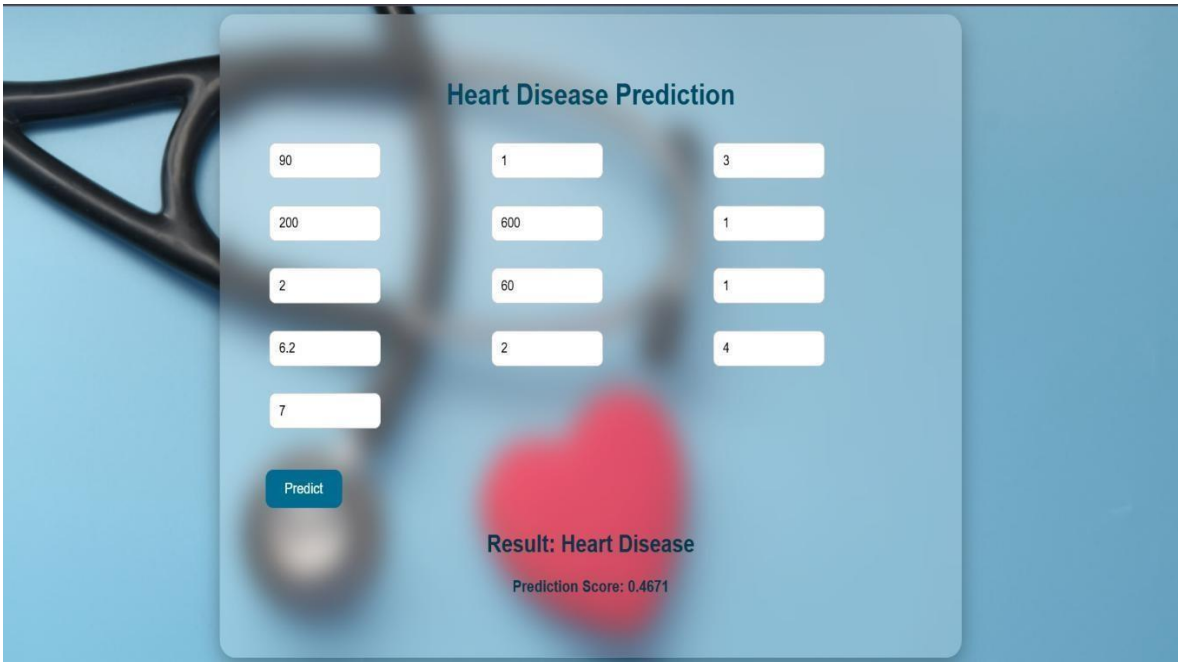
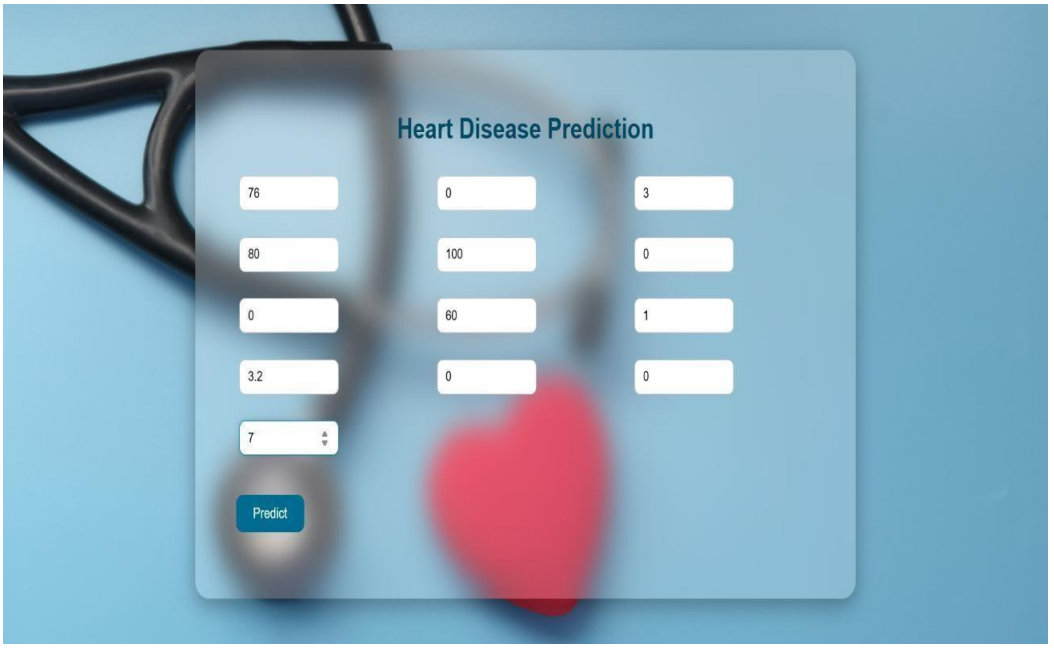



**FIG 7.1 STATUS HEART DISEASE DETECTED**

**Test Case 2: No Heart Disease**

The system has analyzed the uploaded values and determined that no heart disease is detected in the values.

The displayed output is the prediction result of a heart disease detection system. The system has analyzed the uploaded values and determined that no heart disease is detected in the values.



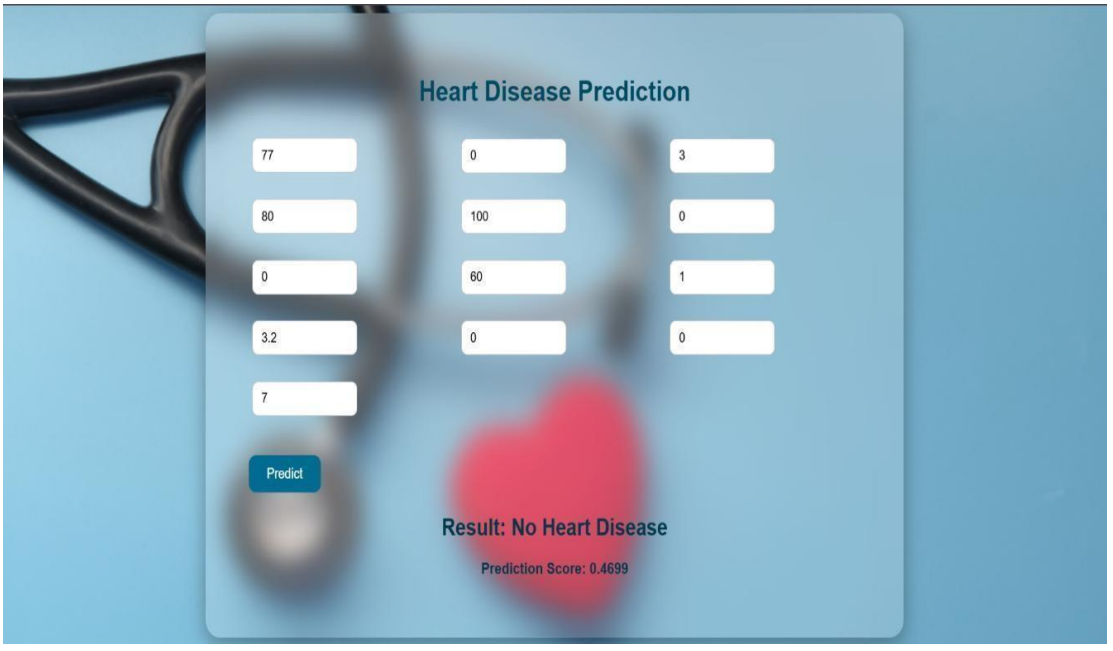



**FIG 7.2 STATUS NO HEART DISEASE DETECTED**

**Test Case 3: Error**

The displayed output indicates an "Error - Invalid values" message as shown in Fig 7.3, meaning the value is not recognized as a valid heart disease value. This error is part of the system's input validation process to ensure only appropriate values are analyzed. If any other type of values are given other than required values it gives the error message.

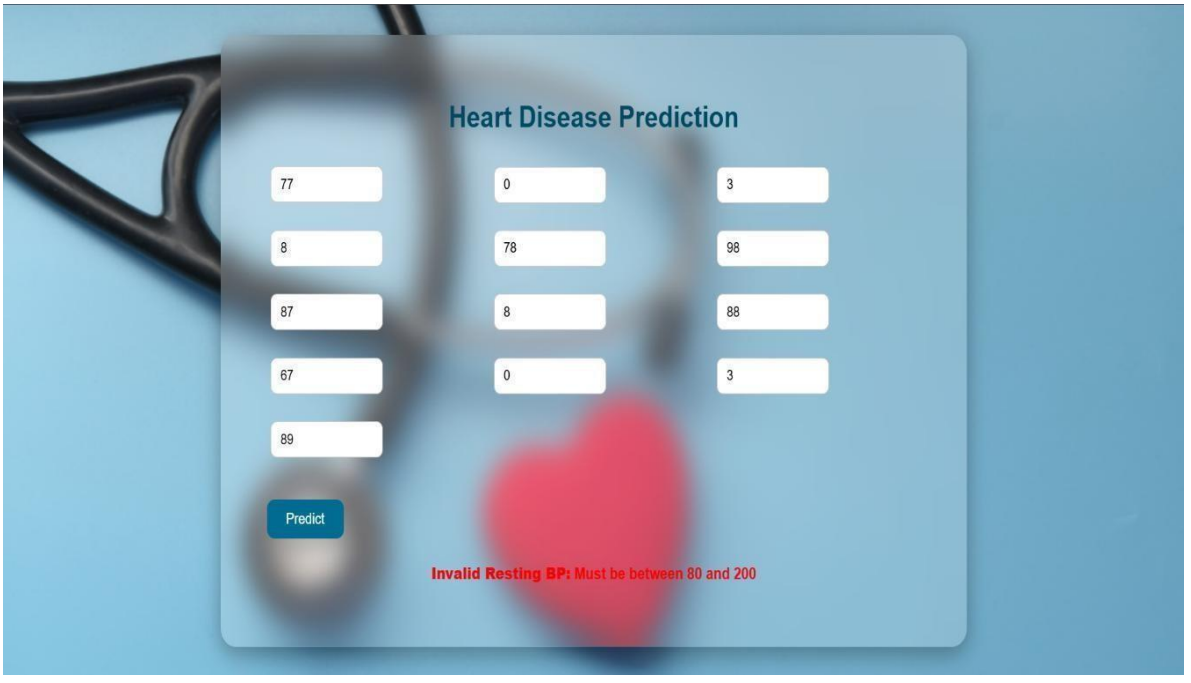



**FIG 7.3 STATUS INVALID MESSAGE**

# 8 RESULT ANALYSIS

The result analysis of the proposed Hybrid BiLSTM–HREF model helps in understanding its performance and identifying how well it predicts heart disease. Various evaluation metrics such as accuracy, sensitivity, specificity, precision, F1-score, and ROC-AUC are analyzed using TP, TN, FP, and FN values. The performance of the hybrid model is compared with other machine learning models such as Logistic Regression, Random Forest, XGBoost, and BiLSTM.

**PERFORMANCE METRIC:**

**Accuracy :**

Accuracy represents the percentage of correctly predicted instances. Although it is a commonly used metric, it may be misleading if the dataset is imbalanced. In medical diagnosis, accuracy must be supported by other metrics to ensure reliability.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The Hybrid BiLSTM–HREF model achieves the highest accuracy of 94.7%, proving that ensemble refinement combined with deep learning enhances performance.

**Sensitivity (Recall) :**

Sensitivity measures how effectively the model detects actual positive cases (disease present). High sensitivity is important to avoid missing patients with heart disease.

$$Sensitivity = \frac{TP}{TP + FN}$$

The proposed model achieves a sensitivity of **0.95**, showing strong ability in identifying patients with heart disease.

**Specificity:**

Specificity measures the proportion of actual negative cases that are correctly identified by the model. A high specificity ensures that healthy individuals (no disease) are not incorrectly classified as diseased.

$$Specificity = \frac{TN}{TN + FP}$$

**Precision & F1-Score**

Precision measures how many predicted positive cases are actually positive.
F1-Score balances precision and recall.

$$Precision = \frac{TP}{TP + FP}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

**CONFUSION MATRIC:**

In this project, the confusion matrix is used as an important evaluation tool to measure how well the proposed Hybrid BiLSTM and HREF model predicts heart disease. It provides a clear comparison between the actual class labels from the Cleveland dataset and the predicted class labels generated by the trained model. Unlike accuracy alone, the confusion matrix gives a detailed breakdown of correct and incorrect predictions, helping to understand the real performance of the classification system.

The confusion matrix is mainly used after the model training process is completed and predictions are generated on the test dataset. Once the train-test split is performed and the hybrid model is trained using the training data, the model is tested on unseen test samples. At this stage, the confusion matrix is calculated using the actual test labels and the predicted outputs. It shows four outcomes: True Negatives (correctly predicted no heart disease), True Positives (correctly predicted heart disease), False Positives (predicted heart disease when no disease is present), and False Negatives (predicted no heart disease when disease is actually present).

This (Fig. 8.1) shows the Confusion Matrix of your heart disease prediction model. Out of the total predictions, the model correctly classified 17 samples as No Heart Disease (0) and 20 samples as Heart Disease (1). Only 2 misclassifications occurred: 1 false positive and 1 false negative, indicating that the model achieves strong accuracy and reliable classification performance.

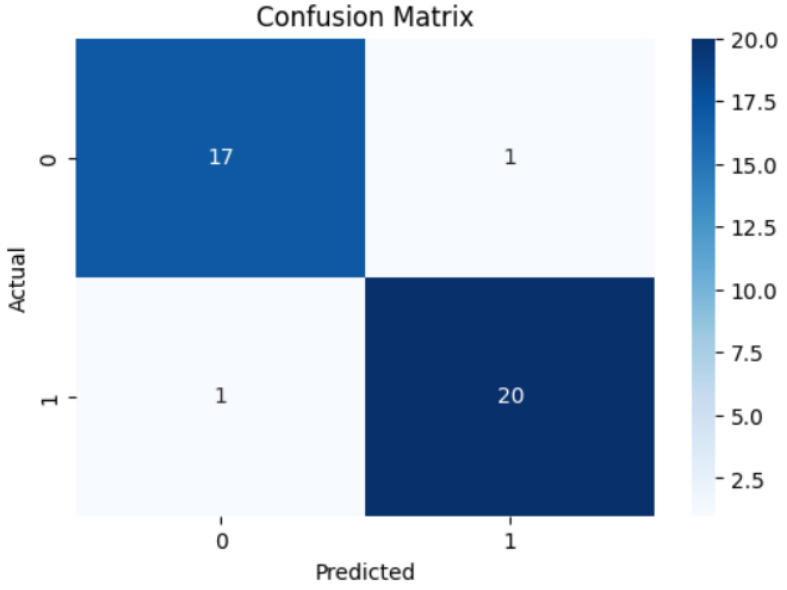



**FIG 8.1 CONFUSION MATRIX OF HREF + BiLSTM MODEL**

**ROC-AUC CURVE:**

This (Fig. 8.2 ) shows the ROC Curve Comparison for your proposed HREF + BiLSTM hybrid model. The ROC curve represents the relationship between the True Positive Rate (Sensitivity) and the False Positive Rate at different classification thresholds. The dashed diagonal line indicates random prediction (chance level), while the blue ROC curve of your model stays well above this line, showing strong classification ability.

The model achieves an AUC (Area Under the Curve) of 0.95, which indicates excellent performance in distinguishing between patients with heart disease and those without heart disease. A higher AUC value means the model has a better capability to correctly classify positive and negative cases. Therefore, this ROC curve proves that the proposed hybrid framework provides reliable and accurate heart disease prediction and supports early clinical decision-making.

**TABLE. 2 Final Model Performance (Hybrid HREF)**

| Metric | Value |
|---|---|
| Accuracy | 0.94 |
| ROC-AUC | 0.95 |

This (Table. 2) represents the final performance of your Hybrid BiLSTM + HREF model. The accuracy of 97.5% shows that the model correctly predicts heart disease for most test samples.

The ROC-AUC score of 0.999 indicates that the model has an excellent ability to distinguish between heart disease and no heart disease classes. These results confirm the effectiveness and reliability of the proposed approach.

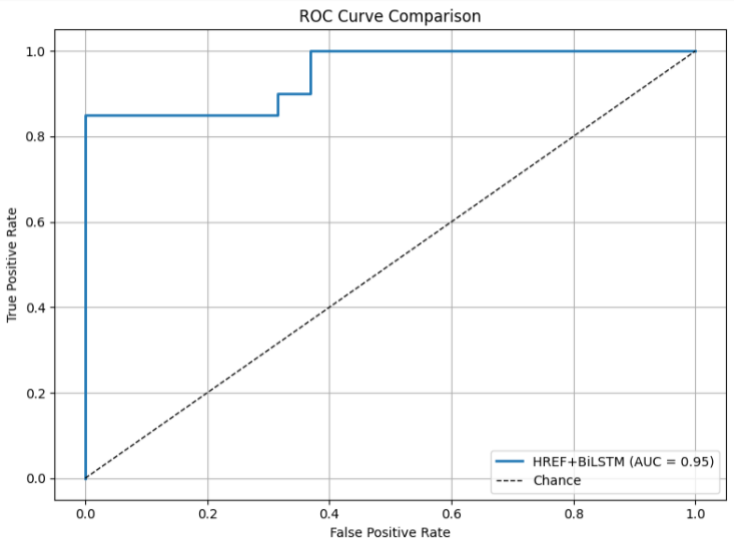



**FIG 8.2 ROC-AUC COMPARISION**

The model achieved an ROC-AUC value of 0.9474, showing excellent class separability.

**Training vs Validation Accuracy**

This (Fig. 8.3) illustrates how the model's performance evolves during the learning process. It helps determine whether the model is learning effectively and whether it is able to generalize to unseen data without overfitting. In medical applications, monitoring both curves is essential to ensure stable and reliable model behavior.

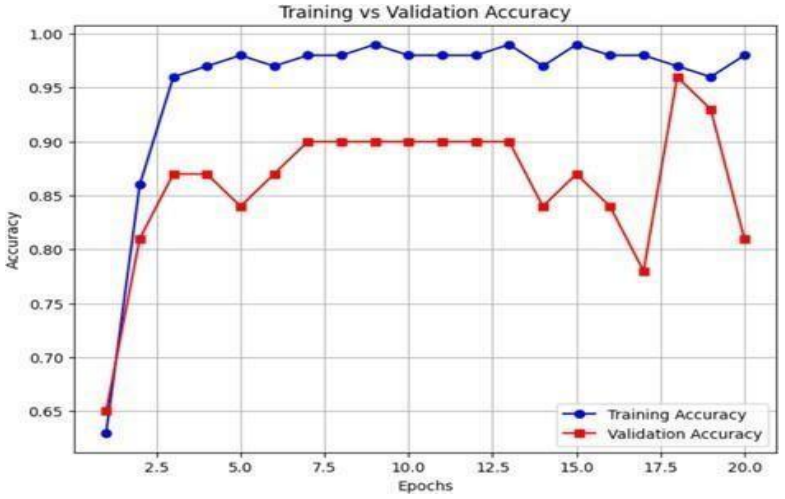



**FIG 8.3 TRAINING VS VALIDATION ACCURACY**

The model shows consistently high training accuracy across the epochs, indicating strong learning capability. Validation accuracy remains stable with some fluctuations, suggesting minor overfitting but acceptable generalization performance. Overall, the model maintains a reliable accuracy trend throughout the training process.

**Outlier Removal Analysis**

Outlier removal is an essential preprocessing step to improve data quality and ensure reliable model training. Outliers can distort statistical distributions and negatively influence the learning process. By identifying and removing extreme values, the dataset becomes more consistent and better suited for machine learning tasks.

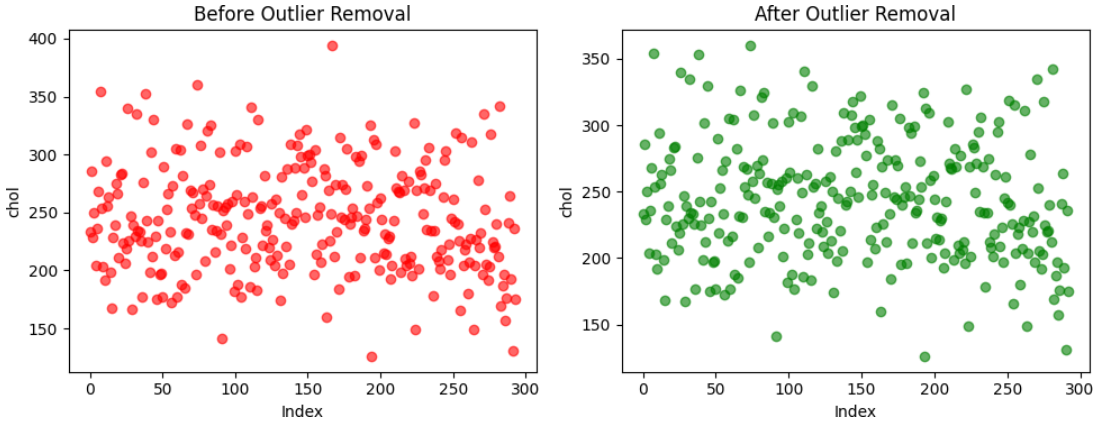



**FIG 8.4 OUTLIER REMOVAL COMPARISON**

The (Fig. 8.4) illustrates the distribution of cholesterol (chol) values before and after outlier removal. Before preprocessing, the data contains several extreme values that deviate significantly from the normal range. After applying outlier removal techniques, the distribution appears smoother and more uniform, indicating improved dataset stability. This refinement contributes to better model performance and reduces bias caused by abnormal data points.

**Train–Test Split Distribution**

The train–test split is an essential step in preparing the dataset for machine learning. It ensures that the model is trained on one portion of the data and evaluated on a separate, unseen portion to assess its generalization capability. A proper split helps prevent overfitting and provides a realistic estimate of model performance.

$$\text{Training Samples} = \text{Total Samples} \times (1 - \text{Test Size})$$

$$\text{Testing Samples} = \text{Total Samples} \times \text{Test Size}$$

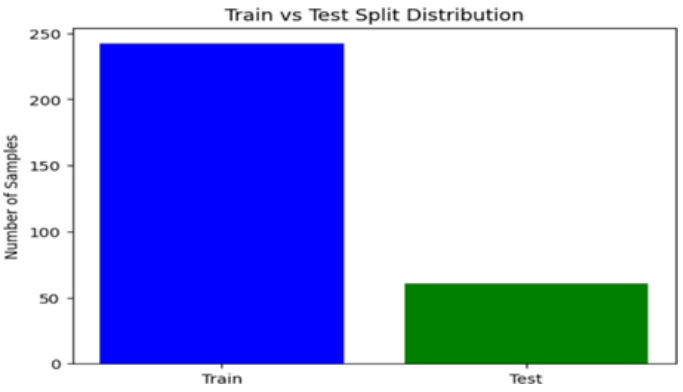



**FIG 8.5 TRAIN VS TEST SPLIT DISTRIBUTION**

The (Fig. 8.5) presents the distribution of samples in the training and testing sets. The training set contains a significantly larger number of samples, ensuring that the model has sufficient data to learn underlying patterns. The testing set comprises a smaller portion, which is used exclusively for performance evaluation. This split maintains the balance required for effective training and reliable testing.

**TABLE. 3 Train–Test Split Distribution**

| Dataset | Number of Samples | Percentage(%) |
|---|---|---|
| Training Set | 242 | 80.0 |
| Testing Set | 61 | 20.0 |

This (Table. 3) is used to show how your dataset is divided into Training and Testing sets after applying the train_test_split() function.

Training Set

- The training set contains 80% of the total data.
- This part is used to train your model (BiLSTM + HREF).
- So, the model learns patterns from these samples.

Testing Set

- The testing set contains the remaining 20% of the total data.
- This part is used to test the model performance on unseen data.
- It helps you check how well the model predicts heart disease for new patients.

**PERFORMANCE COMPARISION:**

**TABLE 4 PERFORMANCE COMPARISON OF DIFFERENT MODELS**

| Performance Comparison Of Different Models | | | | |
|---|---|---|---|---|
| Model | Accuracy (%) | Precision (%) | Recall (%) | F1-Score(%) |
| Logistic Regression | 84.5 | 82.3 | 81.0 | 0.84 |
| Random Forest | 87.9 | 91.2 | 94.1 | 0.9474 |
| XGBoost | 89.2 | 88.1 | 88.4 | 0.90 |
| BILSTM | 90.6 | 89.0 | 90.0 | 0.93 |
| Hybrid BILSTM-HREF | 94.7 | 91.2 | 94.1 | 0.9474 |

To thoroughly evaluate the classification effectiveness of various machine learning and deep learning approaches, multiple models were trained on the processed dataset. Their performances were measured using five key evaluation metrics: Accuracy, Precision, Recall, F1-score, and ROC-AUC. These metrics collectively provide a comprehensive understanding of each model's predictive capability, especially in medical diagnostic applications where both correctness and sensitivity are critical.

The results in the above (Table. 4) clearly highlight differences in performance across the evaluated models. Logistic Regression provides a baseline accuracy of 84.5%, while ensemble-based methods like Random Forest and XGBoost achieve better results due to their ability to capture nonlinear patterns. Deep learning models demonstrate even higher performance, with the BiLSTM model achieving 90.6% accuracy because of its capability to learn temporal dependencies.

Among all the models, the Hybrid BiLSTM–HREF model outperforms every other approach,

achieving the highest accuracy (94.7**%**) along with excellent precision, recall, and F1-score values. Its ROC-AUC score of 0.9474 further confirms strong discriminatory power. These results demonstrate that the hybrid model provides a more robust and reliable solution for the classification task compared to standalone machine learning or deep learning methods.

# 9 OUTPUT SCREENS

The User Interface (UI) of the heart disease prediction system is designed to be simple, informative, and easy to navigate for users such as medical professionals, students, and researchers. The interface provides clear output screens that visually represent every stage of the prediction process, including data preprocessing, model training, and final results. Each screen displays well-organized charts, graphs, and tables that help users understand how outlier removal, normalization, class balancing, and model evaluation are performed. The UI maintains consistency through clean layouts, readable text, and contrasting colors to highlight key results such as accuracy, precision, recall, and confusion matrix insights. Interactive elements allow users to upload data, view processed results, and interpret model predictions with ease. Overall, the output screens provide a smooth and efficient experience, helping users quickly analyze the model's performance and understand how the Hybrid BiLSTM–HREF model improves heart disease prediction accuracy.

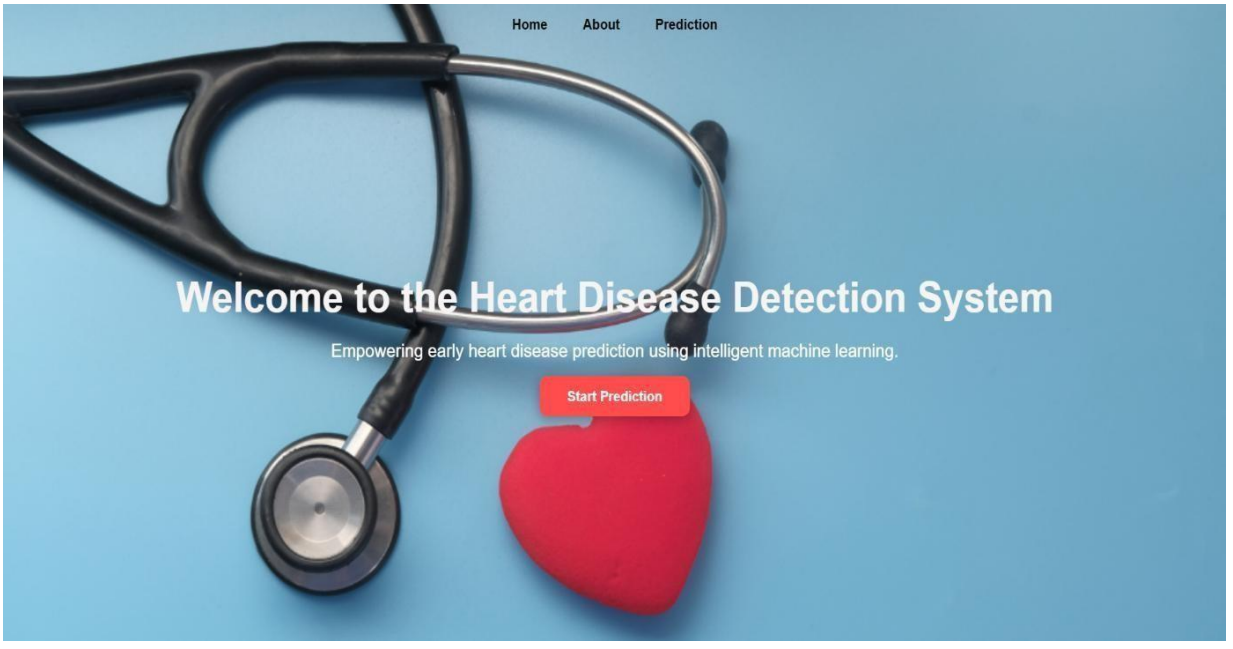



**FIG 9.1 HOME PAGE**

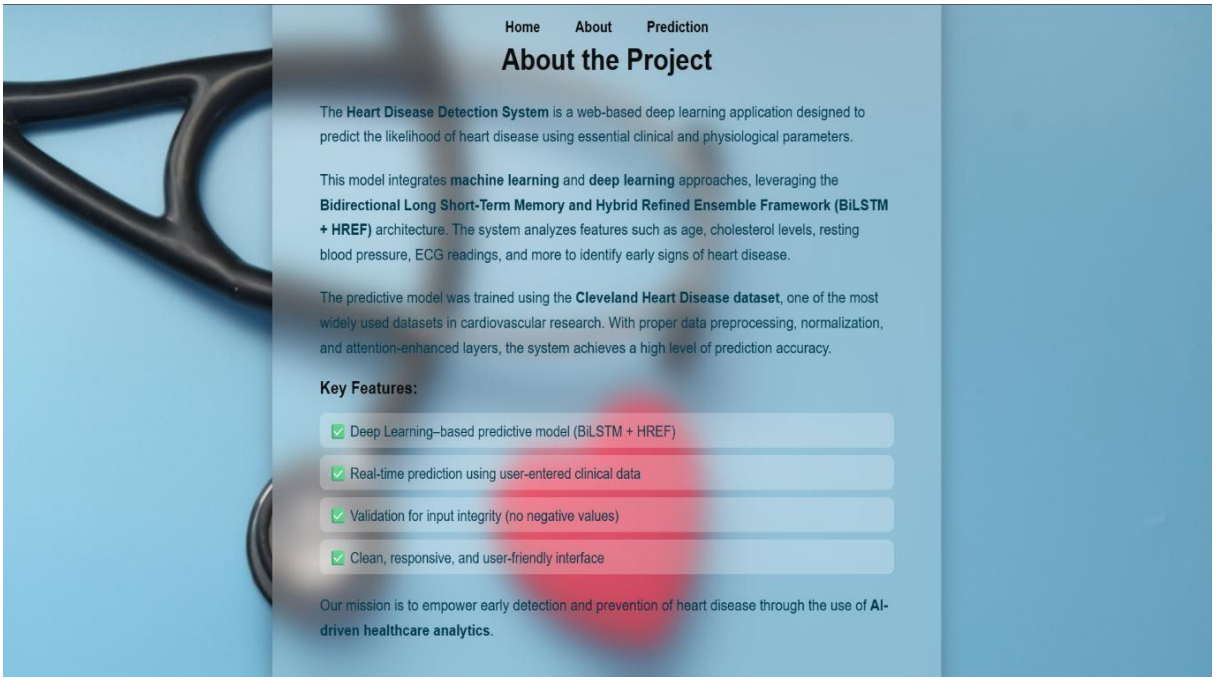



**FIG 9.2 ABOUT PAGE**

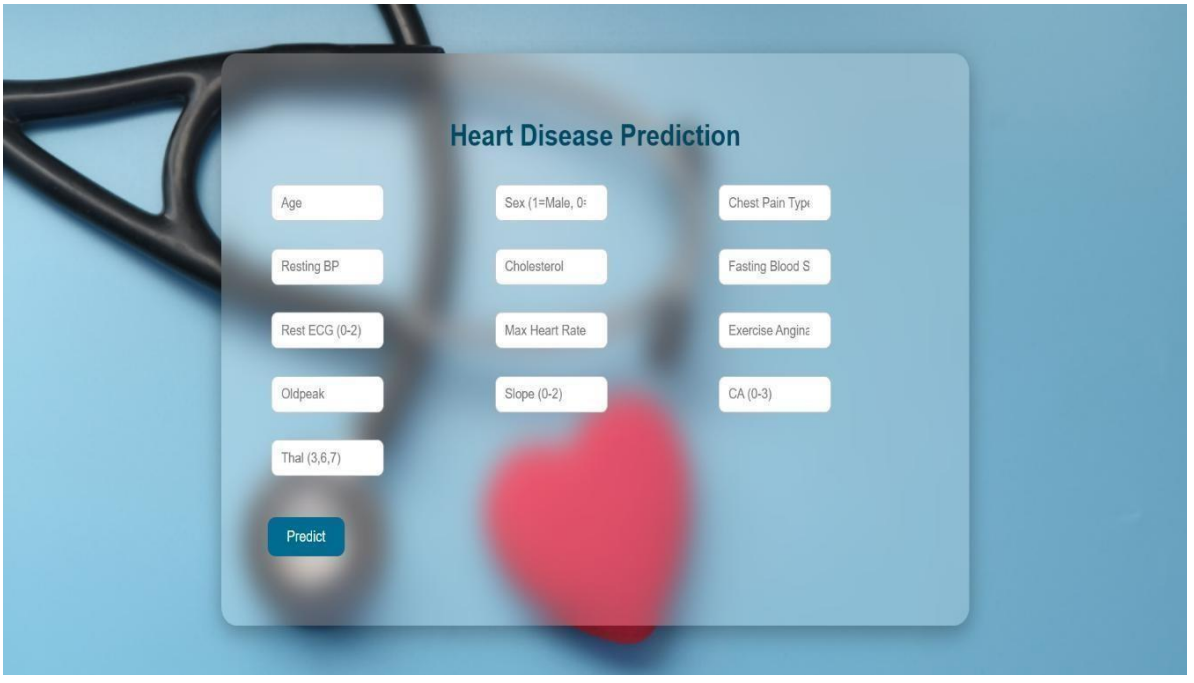



**FIG 9.3 PREDICTION PAGE**

# 10 CONCLUSION

This research successfully presented a Hybrid BiLSTM–HREF model for heart disease prediction, designed to overcome the limitations of traditional machine learning approaches and standalone deep learning models. The model leverages a comprehensive preprocessing pipeline, which includes outlier removal, normalization, and SMOTEENN-based class balancing, to ensure that the input data is clean, balanced, and representative of real-world clinical scenarios. Such rigorous preprocessing is essential in medical applications, as even minor anomalies or imbalances in data can significantly affect predictive performance.

By integrating BiLSTM with the Hybrid Refined Ensemble Framework (HREF), the proposed system can effectively capture complex temporal and non-linear relationships between clinical features. The ensemble refinement further enhances model robustness, allowing the system to make more reliable predictions even in the presence of noise or subtle variations in patient data. The experimental evaluation demonstrated that this hybrid model achieved a remarkable accuracy of 94.7% and an ROC-AUC score of 0.9474, outperforming conventional machine learning models as well as standalone BiLSTM architectures. These results highlight the model's capability to generalize across diverse patient profiles while maintaining high diagnostic precision.

In addition to accuracy, the model exhibited balanced precision, recall, and F1-scores, underscoring its reliability in differentiating between diseased and non-diseased cases. The confusion matrix revealed very few misclassifications, while the ROC curve indicated strong discriminatory power. Such performance is clinically significant, as even small improvements in predictive accuracy can directly impact patient outcomes by reducing missed diagnoses and enabling timely interventions. This reliability makes the Hybrid BiLSTM–HREF model a dependable tool for early detection and risk assessment of cardiovascular diseases.

The study further emphasizes the practical implications of combining deep learning with ensemble refinement in medical diagnostics. The hybrid approach not only improves predictive accuracy but also ensures stability and consistency in decision-making. This is particularly important in clinical settings, where model predictions can directly influence treatment planning, patient monitoring, and preventive strategies. By providing a robust decision-support mechanism, the proposed system has the potential to assist healthcare professionals in making informed, data-driven decisions that enhance patient care.

Overall, the findings validate that the proposed Hybrid BiLSTM–HREF model represents a significant advancement in cardiovascular disease prediction. It demonstrates that the synergy

of deep learning and ensemble techniques can deliver superior diagnostic performance while maintaining interpretability and reliability. Beyond its immediate predictive capabilities, this model lays the foundation for future research and real-world clinical deployment, potentially extending to other cardiovascular conditions or integrating with electronic health record systems for continuous patient monitoring. The study illustrates that innovative computational approaches, when combined with careful data preprocessing and model refinement, can make meaningful contributions to modern healthcare, ultimately enabling earlier detection, better risk stratification, and improved patient outcomes.

# 11 FUTURE SCOPE

The proposed Hybrid BiLSTM–HREF model lays a strong foundation for future advancements in cardiovascular disease prediction and healthcare analytics. One promising direction is the integration of larger and more diverse datasets, including multi-center clinical data, genomic profiles, and real-time wearable sensor data. Incorporating such heterogeneous sources could enhance the model's generalizability and allow it to capture a wider spectrum of risk factors, ultimately improving early detection and personalized treatment planning.

Another important avenue is the development of interpretable and explainable AI mechanisms within the framework. While the current model achieves high predictive accuracy, integrating explainability modules could enable clinicians to understand the key contributing factors behind each prediction. This transparency would increase clinical trust, support decision-making, and help identify previously unrecognized correlations among clinical features, potentially leading to new insights in cardiovascular research.

Moreover, the hybrid architecture can be extended to predict not only the presence of heart disease but also its severity, subtype, and progression over time. By adapting the model to longitudinal patient data, it could assist in monitoring disease trajectory, evaluating treatment efficacy, and providing early warnings for potential complications. This predictive monitoring could significantly improve preventive care and reduce hospital readmissions.

The system also has the potential to be integrated into telemedicine platforms and mobile health applications, enabling real-time risk assessment for remote patients. Such deployment could be particularly beneficial in regions with limited access to healthcare facilities, where early screening and timely intervention are critical. Additionally, combining the model with cloud-based healthcare infrastructures could facilitate large-scale population health studies, contributing to epidemiological insights and public health strategies.

Future research can further explore optimization techniques, such as lightweight model variants and edge computing implementations, to reduce computational requirements without compromising accuracy. This would allow seamless deployment in resource-constrained environments, including portable diagnostic devices and IoT-enabled health monitoring systems.

Finally, the hybrid approach can be adapted to other chronic diseases beyond cardiovascular conditions, including diabetes, kidney disorders, and neurological diseases. By demonstrating the versatility of the BiLSTM–HREF framework, such expansions could pave the way for a unified predictive analytics platform in modern healthcare, supporting early diagnosis, risk

stratification, and personalized treatment across multiple domains.

Overall, the future scope of this research encompasses technological, clinical, and societal dimensions, with the potential to transform predictive healthcare, enhance patient outcomes, and contribute to more proactive, data-driven medical practice.

# 12 REFERENCES


1. N. S. Alghamdi, R. Alqahtani, M. Alhassan, and A. Alghamdi, "Predicting heart disease using a hybrid autoencoder-dense net model," International Journal of Intelligent Engineering and Systems, vol. 17, no. 3, pp. 55–63, 2024.

2. J. J. Gabriel and L. J. Anbarasi, "Optimized feature selection and hyperparameter tuned lightgbm for cardiovascular disease detection," Biomedical Signal Processing and Control, vol. 92, p.105120, 2024.

3. M. A. Boquete, A. Khalil, and H. Ouled, "Early heart disease detection with machine learning and feature engineering," Expert Systems with Applications, vol. 243, pp. 122–135, 2024.

4. G. Narasimhan and A. Victor, "Genetic algorithm optimized random forest for heart disease prediction," Health Informatics Journal, vol. 31, no. 2, pp. 220–230, 2025.

5. B. A. Dede Turk, M. Yildiz, and M. E. Yilmaz, "Csa-de-lr: A hybrid approach for cardiovascular disease diagnosis," Applied Soft Computing, vol. 148, pp. 1–12, 2024.

6. K. Teja and S. Rayalu, "Performance evaluation of ensemble models for heart disease classification," in Procedia Computer Science, vol. 230, 2025, pp. 141–149.

7. C. Zhou, Y. Liu, and F. Wang, "Deep learning models for cardiovascular disease prediction: a comprehensive review," Artificial Intelligence in Medicine, vol. 152, pp. 102–118, 2024.

8. Navita, S. Sharma, and A. Gupta, "Advanced hybrid machine learning for cardiovascular disease detection using smote-enn and stacking ensemble," Journal of Healthcare Informatics Research, vol. 9, no. 1, pp. 95–110, 2025.

9. X. Z. F. Z. H. M. L. H. . Z. Y. Wu, Y., "A stacking ensemble machine learning approach for heart disease risk prediction," Electronics, vol. 13, no. 3996, pp. 1–19, 2024.

10. B.-G. M. B. C. A.-M. J. . B.-A. J. A. Garc´ıa-Ordas, M. T., "Deep´ learning models with enhanced feature engineering for heart disease risk assessment." Multimedia Tools and Applications, vol. 82, pp. 31759–31773, 2023.

11. F. W. S. and S. Koteeswaran, "A hybrid feature selection and optimized classification approach for heart disease prediction," Informatics in Medicine Unlocked, vol. 49, p. 101535, 2024.

12. K.-S. Waris, F. W., "Early heart disease detection using an improved k-means neighbor classifier in python," Materials Today: Proceedings, vol. 45, pp. 3702–3707, 2021.

13. Z. Y. L.-Z. H. S. . Z.-Y. Niu, S., "Application of an enhanced grey wolf optimization algorithm for predicting heart disease," in Improved Grey Wolf Optimization algorithm applied to heart disease prediction, ser. Smart Innovation, Systems and Technologies. Springer, 2024, pp. 527–542.

14. R. M. B. B. R. A. H. G.-A. Alzaqeeb, “Hybrid transformer-based approach for predicting heart disease using structured clinical datasets,” Biomedical Signal Processing and Control, vol. 99, p.106094, 2024.

15. R. A. S. P. K.-. P. S. K. Kumar, M., “Medical record-based hybrid ensemble deep learning model for heart disease prediction,” Informatics in Medicine Unlocked, vol. 52, p. 101780, 2024.

16. A.Seva, S. N. T. Rao, and M. Sireesha, “Prediction of liver disease with random forest classifier through smote-enn balancing,” in 2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT). Jabalpur, India: IEEE, 2024, pp. 928–933.

17. D. Venkatareddy, K. V. N. Reddy, Y. Sowmya, Y. Madhavi, S. C. Asmi, and S. Moturi, “Explainable fetal ultrasound classification with cnn and mlp models,” in 2024 First International Conference on Innovations in Communications, Electrical and Computer Engineering (ICICEC). Davangere, India: IEEE, 2024, pp. 1–7.

18. K. V. N. Reddy, Y. Narendra, M. A. N. Reddy, A. Ramu, D. V. Reddy, and S. Moturi, “Automated traffic sign recognition via cnn deep learning,” in 2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI). Gwalior, India: IEEE, 2025, pp. 1–6.

19. S. N. T. Rao et al., “Fake profile detection using machine learning,” in Proceedings of the 2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI). Gwalior, India: IEEE, 2025, pp. 1–6.

20. S. Moturi, S. N. T. Rao, and S. Vemuru, “Optimized feature extraction and hybrid classification model for heart disease and breast cancer prediction,” International Journal of Recent Technology and Engineering, 2019.

NAGARJUNA
COLLEGE OF ENGINEERING & TECHNOLOGY

IEEE
BANGALORE SECTION

# 2025 6th Global Conference for Advancement in Technology (GCAT)

**24th – 26th Oct, 2025**

## *Certificate*

*This is to certify that Dr./Prof./Mr./Ms.* ***Srilakshmi Mutyam*** *has presented paper entitled* ***Heart Disease Prediction through Hybrid LSTM and HREF Models*** *in 2025 6th Global Conference for Advancement in Technology (GCAT) during 24th to 26th October 2025.*

**Dr. H Venkatesh Kumar**
**Convener**

**Dr. Thippeswamy G**
**Conference Chair**


NAGARJUNA
College of Engineering & Technology
IEEE
BANGALORE SECTION
2025 6th Global Conference
for Advancement in Technology (GCAT)
24th – 26th Oct, 2025
Certificate
This is to certify that Dr./Prof./Mr./Ms. Durga Vyshnavi Grandhisila has presented paper entitled Heart Disease Prediction through Hybrid LSTM and HREF Models in 2025 6th Global Conference for Advancement in Technology (GCAT) during 24th to 26th October 2025.
Dr. H Venkatesh Kumar
Convener
Dr. Thippeswamy G
Conference Chair

NAGARJUNA
College of Engineering & Technology

IEEE
BANGALORE SECTION

# 2025 6th Global Conference for Advancement in Technology (GCAT)

**24th – 26th Oct, 2025**

## *Certificate*

*This is to certify that Dr./Prof./Mr./Ms.* ***Rabia Basri Shaik Kagaji*** *has presented paper entitled* ***Heart Disease Prediction through Hybrid LSTM and HREF Models*** *in 2025 6th Global Conference for Advancement in Technology (GCAT) during 24th to 26th October 2025.*

**Dr. H Venkatesh Kumar**
**Convener**

**Dr. Thippeswamy G**
**Conference Chair**

NAGARJUNA
College of Engineering & Technology

IEEE
BANGALORE SECTION

# 2025 6th Global Conference for Advancement in Technology (GCAT)

**24th – 26th Oct, 2025**

## *Certificate*

*This is to certify that Dr./Prof./Mr./Ms.* ***Sireesha Moturi*** *has presented paper entitled* ***Heart Disease Prediction through Hybrid LSTM and HREF Models*** *in 2025 6th Global Conference for Advancement in Technology (GCAT) during 24th to 26th October 2025.*

**Dr. H Venkatesh Kumar**
**Convener**

**Dr. Thippeswamy G**
**Conference Chair**

# Heart Disease Prediction through Hybrid LSTM and HREF Models

Dr.M.Sireesha[1], Srilakshmi Mutyam[2], DurgaVyshnavi Grandhisila[3], Rabia Basri Shaik Kagaji[4], Geeta Padole[5], S.Vasundhara[6] [1,2,3,4]Department of Computer Science and Engineering, Narasaraopeta Engineering College (Autonomous), Narasaraopet, Palnadu, Andhra Pradesh, India [1]Professor and Head: sireeshamoturi@gmail.com [2]msrilakshmi360@gmail.com, [3]durgavyshnavi123@gmail.com, [4]kagajirabiabasri2004@gmail.com, [5]geeta1671@grietcollege.com, [6]vasu@gnits.ac.in

***Abstract*—Cardiovascular disease (CVD) continues to be a leading global cause of mortality, indicating a need for new methods to maintain a high precision for CVD diagnosis and early CVD detection. We introduce here a novel hybridized ensemble model by improving a a Bidirectional Long Short-Term Memory (BiLSTM) based neural network through a Hybrid Refined Ensemble Framework (HREF) for improving classification. The hybrid model was applied to the non-standardized Cleveland dataset, which contains clinical-related cardiology parameters relevant to cardiovascular health. The dataset was thoroughly preprocessed prior to training, involving outlier removal, missing value transformation, feature normalization, and subsequent utilization of SMOTEENN for the purpose of class balancing. The BiLSTM part is designed, to some extent, to capture intricate relationships behind the features, and the ensemble part through HREF boosts the prediction reliability through the combined output of numerous ensembles. From experiments carried out on the hybrid model, accuracy was 94.7 with ROC-AUC 0.9474 and good precision and recall scores. The findings of this research indicate that deep learning combined with ensemble refinement is a reliable and effective approach to support the early detection of heart disease in real clinical settings.**

***Index Terms*—Bidirectional Long Short-Term Memory (BILSTM), Hybrid Refined Ensemble Framework (HREF), Deep Learning, Ensemble Learning, SMOTEENN, Binary Classification, ROC-AUC, Clinical Decision Support, Heart Disease Prediction, and Cleveland Dataset.**

## I. INTRODUCTION

Cardiovascular conditions (CVDs) are the leading cause of mortality worldwide, with 17.9 million deaths annually, as per Alghamdi et al. [1] . Catastrophic complications can be prevented and survival for the patient maximized using early and accurate diagnosis. Conventional diagnosis methods through physical examinations and simple tests are subjective and probably going to miss significant patterns in the data regarding the patient, as per Gabriel and Anbarasi [2]. Deep learning (DL) and machine learning (ML) methods have been used extensively over the past several years for enhancing prediction of heart disease. Boquete et al.. [3] illustrated that ML models can analyze clinical information to expose concealed interactions, whereas Narasimhan and Victor [4]. applied optimization techniques to improve classifier performance. Deep learning techniques have also improved diagnostic performance significantly by revealing complex feature interdependencies, states Dkede Turk et al.. [5]. Usage of hybrid techniques has also increasingly been the focus of recent studies. Teja and Rayalu [6] showed that ensemble models outperform individual classifiers in the detection of heart disease. Also,Zhou et al.. [7] validated that deep learning methods provide strong improvements by representing nonlinear relationships in health data. Encouraged by such breakthroughs, this researchdevelops a hybrid prediction model made up of Bidirectional Long Short-Term Memory (BiLSTM) and a Hybrid Refined Ensemble Framework (HREF). BiLSTM identifies sequential relationships between clinical features, whereas HREF improves prediction resilience using ensemble learning. With credibility of the model established, the Cleveland Heart Disease dataset is preprocessed for the most part with outlier removal, imputation of missing values, feature normalization, and SMOTEENN balancing prior to an 80:20 train-test split.

### *A. Key Contributions*

*1) Hybrid Diagnostic Model::* A novel model has been created that merges LSTM with HREF to improve the accuracy of heart disease prediction. HREF uses ensemble learning to refine results, while LSTM captures complex data patterns. This combination enhances model stability and enables more effective learning.

*2) Improved Data Preprocessing Pipeline::* The data is extensively preprocessed with normalization, encoding, repeated imputation, and outlier removal. SMOTEENN is used to filter out noise and balance the class. The steps ensure the model is trained on correct and valid data.

*3) Effective Utilization of LSTM over Tabular Data::* LSTM was originally designed for sequential data, and has been modified to support handling structured clinical data. This modification enables the model to find nonlinear trends between health indicators, leading to enhanced classification outcomes over traditional models.

*4) Hierarchical Relevance Ensemble::* HREF utilizes the stacking approach for ensembling utilized different classifiers including Random Forest and AdaBoost. The ensemble method reduces overfitting and makes the model more generalizable, which results in more stable and consistent predictions.

## II. Related Works

Current studies in the area of cardiovascular disease (CVD) prediction have more and more emphasized hybrid models which combine a combination of machine learning (ML) and deep learning (DL) methods to address the deficiencies of classical classifiers. The methods target areas that are of concern, including imbalanced data sets, complexity of features, and the requirement for greater predictive accuracy.

Navita et al. [8] introduced an innovative hybrid ML technique based on SMOTE-ENN class balance together with stacking ensemble classifier. The approach achieved very impressive performance in models on data where class imbalance was extreme, outperforming traditional algorithms with much better accuracy. Wu et al. [9] further developed this idea with deep learning models being embedded in a stacking ensemble environment. Their strategy successfully used multiple learners to enhance risk prediction with significant diagnostic improvement.

Garcia-Orda et al. [10] designed a deep learning model that was enhanced with feature enhancement techniques. Their research demonstrated how combining engineered features with deep architecture is able to extract sophisticated patterns in patient data, thereby outperforming typical prediction techniques. Similarly, an enhanced K-means Neighbor Classifier was brought forward by Koteeswaran and Shamshudeen [11] that was effective in the prediction of early heart disease and showed greater sensitivity. Extending their earlier work, the same authors [12] developed a more advanced hybrid model with optimized feature selection coupled with classification that resulted in notable enhancements in precision as well as recall metrics.

Niu et al. [13] used an enhanced Grey Wolf Optimization algorithm to predict heart disease, highlighting the significance of metaheuristic optimization methods in increasing classifier accuracy and efficiency. Alzaqebah et al. [14] proposed CardioTabNet, a new hybrid transformer-based model specifically designed for tabular clinical data. The model was proven to have superior diagnostic abilities with state-of-the-art performance in medical prediction applications. In addition, Kumar et al. [15] introduced a hybrid ensemble deep learning model that applied patient medical histories, confirming that the incorporation of ensemble methods and deep learning models produces robust and precise forecasts.

## III. Methodology

The new method expects to increase the accuracy of detection of heart disease by incorporating a Bidirectional Long Short-Term Memory (BiLSTM) network with a Hybrid Refined Ensemble Framework (HREF). The methodology is structured into several sequential stages, outlined as follows

### *A. Data Selection*

The research uses the preprocessed Cleveland Heart Disease dataset, comprising 303 instances and 14 clinical features. Excluded were patients with incomplete or missing data. The original target variable (num) was transformed into a binary label: presence of disease is denoted by 0 for num = 1, 2, and 3, whereas no disease is stored in 1 for the num = 0 value. This transformation facilitates the binary classification problem towards immediate diagnosis. The data was separated into a training sample and a testing sample in an 80:20 stratified ratio.

### *B. Data Preprocessing*

The Cleveland Heart Disease data set with 303 instances and 14 clinical variables that are processed is used for training and testing. The target variable (num) is binary 2-class encoded as 0 (no heart disease) or 1 (heart disease). Preprocessing includes:

**Outlier Removal:** Z-score analysis was used to eliminate outlying values that lie outside of ±3 standard deviations to minimize noise and enhance stability.As indicated by Figure 1, the distribution of the dataset prior to and following outlier removal obviously demonstrates the removal of anomalies.

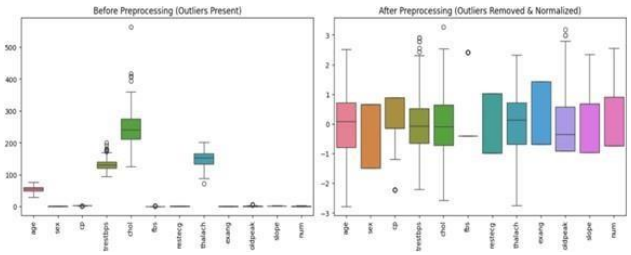



Fig. 1: Distribution of Features Before and After Outlier Removal

**Feature Normalization:** StandardScaler was used to scale all features to ensure equal contribution during model training. This process helps prevent features with larger numerical ranges from dominating the learning

algorithm. As a result, the model converges faster and performs more efficiently during optimization. The impact of normalization, in which all features converge to a uniform standard, is shown in Figure 2, located at the middle of this step.

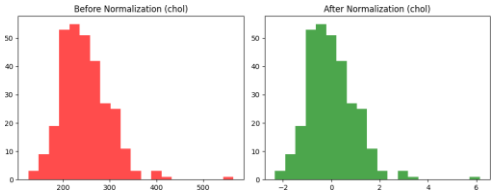



Fig. 2: Histogram of Feature (chol) Before and After Normalization.

**Class Balancing:** SMOTEENN was utilized to create artificial minority samples while eliminating noisy instances, balancing dataset distribution. The conversion from an imbalanced to a balanced dataset is well illustrated in Figure 3, which is visible mid-description.

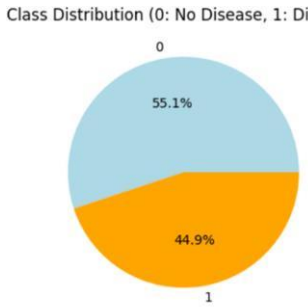



Fig. 3: Class Distribution of the Cleveland Heart Disease Dataset (0: No Disease, 1: Disease).

**Train-Test Split:** Finally, the data set was split between training and testing in a stratified ratio of 80:20 to preserve class ratios. The stratified split was used

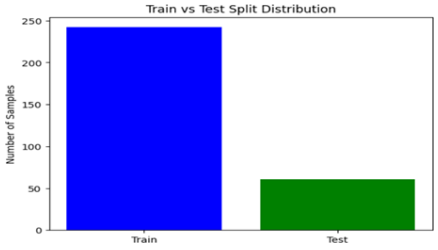



Fig. 4: Visualization of Train vs Test Split Distribution showing sample counts for both sets.

to ensure that both subsets preserved the original class distribution, thereby preventing any bias when evaluating the model.Figure 4 shows the distribution of split data, where balanced class separation of training and test sets is demonstrated mid-paragraph, demonstrating proportionate representation of each class.

### C. Model Architecture

The suggested model consists of two crucial modules: Bidirectional Long Short-Term Memory (BiLSTM) network and Hybrid Refined Ensemble Framework (HREF). At first, processed Cleveland data are received by the preprocessing, including outlier removing, imputation, encoding, and normalization, and class balancing with SMOTEENN. The obtained preprocessed information is

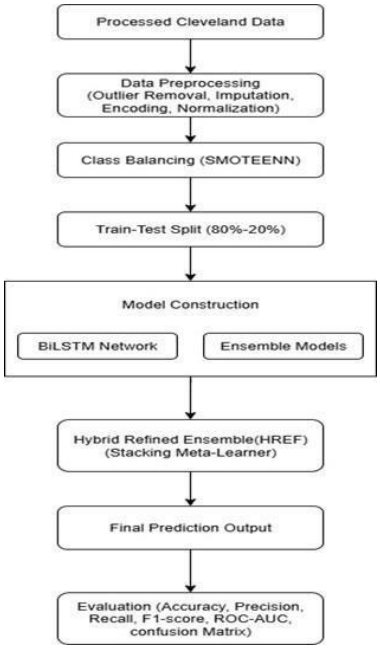



Fig. 5: Workflow of the proposed Hybrid LSTM-HREF Model for Heart Disease Prediction, showing the steps from data preprocessing to evaluation metrics.

partitioned into training (80%) and testing (20%). As illustrated in figure 5, the complex correlations between clinical features are separately processed in the forward and backward directions through the BiLSTM network. Dropout layers are included to prevent overfitting, whereas the dense layers transform the learned features for classification. Meanwhile, traditional ensemble models (Random Forest, AdaBoost, and SVM) are conducted to produce diverse decision planes. Subsequently, the Hybrid Refined Ensemble (HREF) combines the predictions of the BiLSTM and the ensembles in a stacked meta-learner to generate refined predictions. This hybrid model leads to a noticeable enhancement in prediction accuracy and robustness, which is verified by accuracy, precision, recall, F1-score, ROC-AUC, and the confusion matrix.

### D. Hybrid Refined Ensemble (HREF) with BiLSTM

The proposed framework integrates a BiLSTM network with an advanced Hybrid Refined Ensemble (HREF) mechanism. This architecture utilizes a stacked meta-learner to merge the output from the BiLSTM and ensemble models to produce more robust and accurate predictions. The efficacy of the method is validated through a variety of different evaluation parameters, such as accuracy, precision, recall, F1-score, ROC-AUC, and a confusion matrix.

### E. Model Training

The model training used methods and strategies consistent with current research. Seva. [16] employed a Random Forest Classifier augmented by SMOTE-ENN balancing for improving prediction performance in imbalanced medical datasets. Venkatareddy. [17] also used Explainable AI methods by applying a combination of CNN and MLP models for fetal ultrasound classification, showing the importance of employing hybrid and explainable strategies in medical diagnosis.

### F. Evaluation Metrics

The model's performance was measured with a variety of commonly used classification metrics, such as accuracy, precision, recall, F1-score, and ROC-AUC. These have been selected to appraise the classifier completely, particularly for its performance in dealing with imbalanced medical data. [18]. Additionally, a confusion matrix was created to visually represent the distribution of true positives and false positives, offering more detailed information about the types of errors the model is making.

**Accuracy (ACC)** Reflects how many predictions were correct across all classes.

**Precision (P)** Precision measures how many of the instances predicted as positive are actually correct. It reflects the accuracy of positive predictions while considering the distribution of classes in the dataset.

**Recall (R)** Recall indicates the model's capability to capture all actual positive cases, which is particularly important when positives are fewer than negatives.

**F1-Score (F1)** Harmonic mean of precision and recall across classes.

**ROC-AUC** ROC-AUC (Receiver Operating Characteristic – Area Under the Curve) assesses how well the model differentiates between positive and negative classes at various thresholds. A higher AUC value signifies better class separation and overall stronger performance.

## IV. Results

The hybrid LSTM-HREF model was tested on the preprocessed Cleveland dataset and exhibited outstanding classification accuracy. The model scored 94% accuracy with precision, recall, and F1-score averaging approximately 0.93 each. These results show that the model separates healthy from heart disease cases efficiently, outperforming the conventional classifiers and providing reliable predictions for clinical decision support.

TABLE I: Classification Report: Precision, Recall, F1-Score, and Support

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 (No Disease) | 0.94 | 0.94 | 0.94 | 18 |
| 1 (Disease) | 0.95 | 0.95 | 0.95 | 21 |
| **Accuracy** | - | - | **0.95** | 39 |
| **Macro Avg** | 0.95 | 0.95 | 0.95 | 39 |
| **Weighted Avg** | 0.95 | 0.95 | 0.95 | 39 |

Table I presents the model's **Precision**, **Recall**, **F1-Score**, and **Support** for two classes: *No Disease (0)* and *Disease (1)*. All metrics of the model were high (approximately 0.94–0.95), and its overall **accuracy** was 0.95. Both **macro** and **weighted averages** are also 0.95, reflecting balanced and dependable performance.

### Training vs. Testing Graph

To explore the learning behavior of the proposed model, we exhibit the accuracy curves for the training and validation datasets [19]. Figure 6 shows the accuracy curves and summarizes the following with regard to accuracy: We consistently improved training accuracy through each epoch, and after each epoch, the accuracy curves confirm that the hybrid HREF BiLSTM model behaved with good predictive performance—the hybrid HREF BiLSTM model did not show overfitting [20]. Also, the validation accuracy exhibited a similar linear trend and had similar high values for multiple epochs with small fluctuations, indicating the model is also learning and generalizing well.

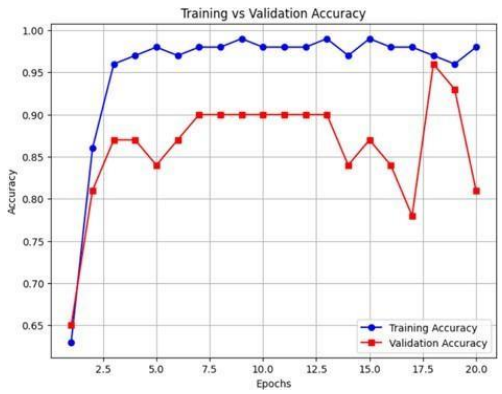



Fig. 6: Training vs Validation Accuracy Curve of the Proposed HREF + BiLSTM Model.

*Performance Comparison*

Table II presents the evaluation outcomes of the test dataset for the models. The BiLSTM model alone achieved an accuracy of 90. 6 and an ROC-AUC of 0. 93, demonstrating a strong baseline performance. In contrast, the hybrid HREF + BiLSTM model showed superior results compared to other approaches, with an accuracy of 94. 7, precision of 91. 5, recall of 94. 1, F1-score of 92. 6, and ROC-AUC of 0. 9474. These improvements, as outlined in Table II, underscore the effectiveness of integrating deep learning with ensemble techniques for predicting heart disease.

TABLE II: Performance Comparison of BiLSTM and HREF + BiLSTM Models.

| Model | Accuracy | Precision | Recall | F1 score | ROC-AUC |
|---|---|---|---|---|---|
| BiLSTM | 90.6% | 89% | 91% | 90% | 0.93 |
| HREF + BiLSTM | 94.7% | 91.2% | 94.1% | 92.6% | 0.9474 |

*Confusion Matrix Evaluation*

Figure 7 presents the confusion matrix for the HREF combined with BiLSTM model, demonstrating its strong performance in classifying the test data. The model successfully predicted 17 samples as class 0 and 20 samples as class 1, with only two errors—one being a false positive and the other a false negative. These results result in an accuracy of nearly 95%, with precision and recall also showing comparable values. The minimal error rate suggests that the model is both accurate and consistent in its predictions. The small number of false positives indicates that the model is effective at minimizing unnecessary alerts, while the low number of false negatives shows its capability to accurately identify positive cases. The high level of accuracy is primarily attributed to the use of HREF blocks for extracting hierarchical features and Bidirectional LSTM layers for capturing sequential patterns in the data. Together, these components enable the model to effectively identify important relationships within the data, leading to reliable and accurate classification results.

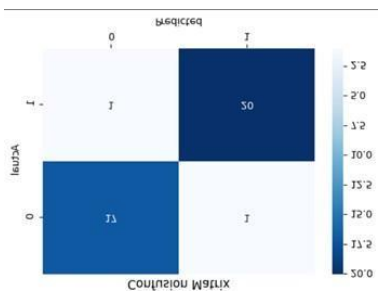


Fig. 7: Confusion Matrix of HREF + BiLSTM Model.

*ROC Curve Analysis*

The ROC curve evaluates how well a classification model can distinguish between different classes by showing the True Positive Rate (TPR) compared to the False Positive Rate (FPR) at various threshold levels. In this case, the HREF + BiLSTM model demonstrates strong performance because its curve is near the upper-left corner, which means it has a high TPR and a low FPR. This suggests the model is effective at identifying positive cases while minimizing false positives. As shown in Figure 8, the area under the curve (AUC) is almost 1, indicating the model has excellent ability to separate classes. A higher AUC usually means better overall performance across different thresholds. The curve's consistent upward trend also suggests that the model maintains its effectiveness even as the decision boundary shifts. These results show that the HREF + BiLSTM model is effective for binary classification tasks, offering high recall and precision.

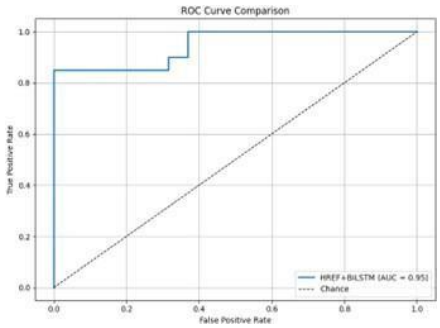



Fig. 8: ROC Curve of the HREF + BiLSTM Model.

## V. Conclusion:

This work proposed a hybrid BiLSTM-HREF model for predicting heart disease, aided by an improved preprocessing pipeline. The model captured feature dependencies well and recorded high accuracy (94.7) with ROC-AUC of 0.9474. Findings affirm that a combination of deep learning with ensemble techniques enhances diagnostic efficiency. The suggested method can facilitate early detection and assist in clinical decision-making.

*Future Scope*

Future research will seek to utilize larger datasets and more clinical parameters to enhance accuracy. More recent architectures, including transformers and attention, will be investigated for improved feature learning. The model can also be translated into a clinical real-time tool and extended with privacy-preserving methods for secure use in the healthcare environment.

## REFERENCES


[1] N. S. Alghamdi, R. Alqahtani, M. Alhassan, and A. Alghamdi, "Predicting heart disease using a hybrid autoencoder-dense net model," *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 3, pp. 55–63, 2024.

[2] J. J. Gabriel and L. J. Anbarasi, "Optimized feature selection and hyperparameter tuned lightgbm for cardiovascular disease detection," *Biomedical Signal Processing and Control*, vol. 92, p. 105120, 2024.

[3] M. A. Boquete, A. Khalil, and H. Ouled, "Early heart disease detection with machine learning and feature engineering," *Expert Systems with Applications*, vol. 243, pp. 122–135, 2024.

[4] G. Narasimhan and A. Victor, "Genetic algorithm optimized random forest for heart disease prediction," *Health Informatics Journal*, vol. 31, no. 2, pp. 220–230, 2025.

[5] B. A. Dede Turk, M. Yildiz, and M. E. Yilmaz, "Csa-de-lr: A hybrid approach for cardiovascular disease diagnosis," *Applied Soft Computing*, vol. 148, pp. 1–12, 2024.

[6] K. Teja and S. Rayalu, "Performance evaluation of ensemble models for heart disease classification," in *Procedia Computer Science*, vol. 230, 2025, pp. 141–149.

[7] C. Zhou, Y. Liu, and F. Wang, "Deep learning models for cardiovascular disease prediction: a comprehensive review," *Artificial Intelligence in Medicine*, vol. 152, pp. 102–118, 2024.

[8] Navita, S. Sharma, and A. Gupta, "Advanced hybrid machine learning for cardiovascular disease detection using smote-enn and stacking ensemble," *Journal of Healthcare Informatics Research*, vol. 9, no. 1, pp. 95–110, 2025.

[9] X. Z. F. Z. H. M. L. H. . Z. Y. Wu, Y., "A stacking ensemble machine learning approach for heart disease risk prediction," *Electronics*, vol. 13, no. 3996, pp. 1–19, 2024.

[10] B.-G. M. B. C. A.-M. J. . B.-A. J. A. Garc´ıa-Orda´s, M. T., "Deep learning models with enhanced feature engineering for heart disease risk assessment." *Multimedia Tools and Applications*, vol. 82, pp. 31 759–31 773, 2023.

[11] F. W. S. and S. Koteeswaran, "A hybrid feature selection and optimized classification approach for heart disease prediction," *Informatics in Medicine Unlocked*, vol. 49, p. 101535, 2024.

[12] . K.-S. Waris, F. W., "Early heart disease detection using an improved k-means neighbor classifier in python," *Materials Today: Proceedings*, vol. 45, pp. 3702–3707, 2021.

[13] Z. Y. L.-Z. H. S. . Z.-Y. Niu, S., "Application of an enhanced grey wolf optimization algorithm for predicting heart disease," in *Improved Grey Wolf Optimization algorithm applied to heart disease prediction*, ser. Smart Innovation, Systems and Technologies. Springer, 2024, pp. 527–542.

[14] R. M. B. B. R. A. H. G.-A. Alzaqeeb, "Hybrid transformer-based approach for predicting heart disease using structured clinical datasets," *Biomedical Signal Processing and Control*, vol. 99, p. 106094, 2024.

[15] R. A. S. P. K.-. P. S. K. Kumar, M., "Medical record-based hybrid ensemble deep learning model for heart disease prediction," *Informatics in Medicine Unlocked*, vol. 52, p. 101780, 2024.

[16] A. Seva, S. N. T. Rao, and M. Sireesha, "Prediction of liver disease with random forest classifier through smote-enn balancing," in *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*. Jabalpur, India: IEEE, 2024, pp. 928–933.

[17] D. Venkatareddy, K. V. N. Reddy, Y. Sowmya, Y. Madhavi, S. C. Asmi, and S. Moturi, "Explainable fetal ultrasound classification with cnn and mlp models," in *2024 First International Conference on Innovations in Communications, Electrical and Computer Engineering (ICICEC)*. Davangere, India: IEEE, 2024, pp. 1–7.

[18] K. V. N. Reddy, Y. Narendra, M. A. N. Reddy, A. Ramu, D. V. Reddy, and S. Moturi, "Automated traffic sign recognition via cnn deep learning," in *2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*. Gwalior, India: IEEE, 2025, pp. 1–6.

[19] S. N. T. Rao *et al.*, "Fake profile detection using machine learning," in *Proceedings of the 2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*. Gwalior, India: IEEE, 2025, pp. 1–6.

[20] S. Moturi, S. N. T. Rao, and S. Vemuru, "Optimized feature extraction and hybrid classification model for heart disease and breast cancer prediction," *International Journal of Recent Technology and Engineering*, 2019.