HEART ATTACK RISK ASSESSMENT USING DEEP LEARNING WITH FEATURE OPTIMIZATION

Mini Project Report

Submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology (B.Tech)

In

Department of CSE (Artificial Intelligence & Machine Learning)

By

Chirra Rishitha	**21AG1A6676**
Goshika Poojitha	**21AG1A6683**
Bukhya Sahith	**22AG5A6608**

Under the Esteemed Guidance of

Mr.Shashank Tiwari

Assistant Professor

Department of CSE (Artificial Intelligence & Machine Learning)

ACE ENGINEERING COLLEGE

An Autonomous Institution

(NBA ACCREDITED B.TECH COURSES: EEE, ECE & CSE, ACCORDED NAAC 'A' GRADE)

Affiliated to Jawaharlal Nehru Technological University, Hyderabad, Telangana,

Ghatkesar, Hyderabad – 501 301

DECEMBER 2024

ACE
Engineering College
An Autonomous Institution

(NBA ACCREDITED B.TECH COURSES: EEE, ECE & CSE, ACCORDED NAAC 'A' GRADE)

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad,

Telangana) Ghatkesar, Hyderabad – 501 301

Website : www.aceec.ac.in E-mail: info@aceec.ac.in

CERTIFICATE

This is to certify that the Mini Project work entitled **"Heart Attack Risk Assessment Using Deep Learning with Feature Optimization** "is being submitted by **Chirra Rishitha (21AG1A6676), Goshika Poojitha (21AG1A6683), Bukhya Sahith (22AG5A6608)** in partial fulfillment for the award of Degree of **BACHELOR OF TECHNOLOGY** in **DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)** to the Jawaharlal Nehru Technological University, Hyderabad during the academic year 2024-25 is a record of bonafide work carried out by him/her under our guidance and supervision.

The results embodied in this report have not been submitted by the student to any other University or Institution for the award of any degree or diploma.

Internal Guide
Shashank Tiwari
Assistant Professor
Dept. of CSE (AI & ML)

Head of the Department
Dr. KAVITHA SOPPARI
Assoc Professor and Head
Dept. of CSE (AI & ML)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our gratitude to all the people behind the screen who have helped us transform an idea into a real time application.

We would like to express our heart-felt gratitude to our parents without whom. We would not have been privileged to achieve and fulfill our dreams.

A special thanks to our Secretary, **Prof. Y. V. GOPALA KRISHNA MURTHY,** for having founded such an esteemed institution. We are also grateful to our beloved principal, **Dr. B. L. RAJU** for permitting us to carry out this project.

We profoundly thank **Dr. Kavitha Soppari,** Assoc. Professor and Head of the Department of CSE (Artificial Intelligence & Machine Learning), who has been an excellent guide and also a great source of inspiration for our work.

We extremely thank, **Mrs. SriSudha Garugu,** Assistant Professor, Mini Project coordinator, who helped us in all the way in fulfilling all aspects in completion of our Mini Project.

We are very thankful to our **Mr. Shashank Tiwari,** Assistant Professor, who has been excellent and given continuous support for the Completion of our Mini Project work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, we would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased our task.

Chirra Rishitha (21AG1A6676)
Goshika Poojitha (21AG1A6683)
Bhukya Sahith (22AG5A6608)

DECLARATION

This is to certify that the work reported in the present project titled "**Heart Attack Risk Assessment Using Deep learning with Feature Optimization**" is a record work done by us in the Department of CSE (Artificial Intelligence & Machine Learning), ACE Engineering College.

No part of the thesis is copied from books/journals/internet and whenever the portion is taken, the same has been duly referred in the text; the reported are based on the project work done entirely by us not copied from any other source.

Chirra Rishitha (21AG1A6676)

Goshika Poojitha (21AG1A6683)

Bhukya Sahith (22AG5A6608)

ABSTRACT

Heart attacks are a significant global health concern, requiring innovative and accurate approaches to identify individuals at risk and enable preventive measures. This project, titled "Heart Attack Risk Assessment Using Deep Learning with Feature Optimization," addresses this challenge by employing advanced deep learning techniques to predict the likelihood of heart attacks effectively.

The system is built around a Fully Connected Neural Network (FCNN), enhanced through feature optimization methods to prioritize the most impactful predictors. By refining the feature selection process, the model achieves higher accuracy and efficiency, ensuring reliable predictions.

To make the results actionable, the project integrates a risk visualization framework that translates complex data into user-friendly insights. This visualization aids in the early detection of heart attack risks, empowering individuals and healthcare professionals to implement timely and effective interventions.

This innovative approach underscores the potential of artificial intelligence in healthcare, highlighting its role in advancing heart attack prevention and management strategies for improved patient outcomes.

Keywords: Deep Learning, Feature Optimization, Heart Attack Risk, Visualization

INDEX

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

Heart attacks, also known as myocardial infarctions, remain one of the leading causes of mortality globally. Early detection of heart attack risk is crucial to reduce fatalities and enable timely medical interventions. The ability to predict heart attack risk accurately can empower individuals to make informed lifestyle choices and healthcare providers to adopt preventive strategies. However, traditional predictive models often rely on manual feature engineering and lack the capability to handle complex patterns in data.

With advancements in artificial intelligence, particularly deep learning, the healthcare industry has witnessed significant progress in predictive analytics. Deep learning models excel at analyzing high-dimensional data, identifying intricate relationships, and making accurate predictions. Unlike traditional machine learning models, deep learning eliminates the need for extensive manual preprocessing and can automatically learn hierarchical patterns from raw data.

This project, titled "Heart Attack Risk Assessment Using Deep Learning with Feature Optimization," aims to design a robust system for predicting heart attack risk. The proposed system employs a Fully Connected Neural Network (FCNN) to process user-specific health data, such as cholesterol levels, Chest pain Type, and exercise-induced angina. To enhance the performance of the FCNN, feature optimization techniques are integrated into the system, ensuring that only the most relevant features contribute to the prediction process.

By combining deep learning with feature optimization, the system aims to achieve high accuracy while reducing computational complexity. The project's key objectives include developing a predictive model capable of identifying risk levels, visualizing results, and providing actionable lifestyle-based recommendations. This integrated approach bridges the gap between raw medical data and practical healthcare solutions, paving the way for more efficient and accessible tools for heart attack risk assessment.

1.1.1 Deep Learning

Deep learning is a branch of machine learning and a subset of artificial intelligence

that focuses on building algorithms inspired by the structure and functioning of the human brain. These algorithms are based on artificial neural networks, which consist of layers of interconnected nodes (neurons) that process and analyze data hierarchically. Unlike traditional machine learning techniques, deep learning models automatically learn meaningful features from raw data, eliminating the need for extensive manual feature engineering.

The core idea of deep learning is to mimic how the human brain processes information by using multiple layers of neurons, each layer extracting increasingly abstract representations of the input data. For example, in image processing tasks:

- The input layer might detect basic patterns like edges or colors.
- The hidden layers combine these patterns to identify shapes and features, such as eyes or lips in a face.
- The output layer classifies the image or performs the desired prediction task.

Deep learning is particularly effective for solving complex problems that involve high-dimensional data, such as image recognition, speech processing, and predictive healthcare. It is implemented using neural networks with multiple hidden layers, commonly referred to as deep networks.

Key Architectures in Deep Learning

1. Deep Neural Networks (DNNs):

DNNs consist of an input layer, multiple hidden layers, and an output layer. These networks are powerful for modeling nonlinear relationships and are used in applications ranging from text analysis to medical diagnostics.

2. Deep Belief Networks (DBNs):

DBNs are a type of generative model that consists of multiple layers of stochastic, latent variables. They are trained layer by layer using algorithms like Contrastive Divergence to model complex data distributions.

3. Recurrent Neural Networks (RNNs):

RNNs are designed to handle sequential data by incorporating feedback loops. These loops enable the network to remember previous inputs, making RNNs ideal for time-series analysis, language modeling, and real-time data predictions.

1.1.2 How Does Deep Learning Work?

Deep learning operates through artificial neural networks that simulate the way the human brain processes information. These networks consist of interconnected layers of nodes, or neurons, organized into three main components: an input layer, one or more hidden layers, and an output layer. The process begins with feeding raw data into the input layer, which converts the data into numerical form. This transformation ensures that the data can be processed by the network. The input is then propagated forward through the network to make predictions or extract features.

As data moves through the network, each layer performs two key operations: a linear transformation and the application of an activation function. The linear transformation calculates a weighted sum of the inputs, adding a bias term. The activation function introduces non-linearity, enabling the network to learn complex patterns and relationships in the data. The output from one layer becomes the input for the next, and this process continues until the data reaches the output layer, where predictions are made.

To refine the network's performance, the predicted outputs are compared to the actual outputs using a loss function. The loss function measures the error between predictions and the ground truth. To minimize this error, the network employs backpropagation, an algorithm that adjusts the weights and biases of the neurons. Gradients of the loss function are computed and propagated backward through the network, and optimization algorithms, such as stochastic gradient descent (SGD) or Adam, are used to update the parameters effectively.

The entire process, known as training, is iterative and involves multiple passes through the dataset, called epochs. With each epoch, the network becomes better at understanding the data and reducing the prediction error. Through this hierarchical and iterative learning process, deep learning models excel at capturing intricate patterns in data, enabling them to solve complex tasks such as image recognition, natural language processing, and autonomous decision-making.

1.1.3 Types of Neural Networks

Neural networks are a key component of deep learning, designed to replicate the way the human brain processes information. They consist of layers of interconnected nodes, or neurons, that process and learn patterns from data. Different types of neural networks are used depending on the application, such as image recognition, speech processing, and predictive analytics. Some of the most commonly used neural networks include Feedforward Neural Networks (FNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Fully Connected Neural Networks (FCNNs).

1.1.3.1 Feedforward Neural Networks (FNNs)

A Feedforward Neural Network (FNN) is the simplest type of artificial neural network in which information flows in one direction—from the input layer, through hidden layers (if any), to the output layer. Unlike recurrent networks, it does not have loops or cycles. Each neuron in a layer is connected to neurons in the next layer, with weighted connections that determine the influence of inputs. FNNs are widely used for tasks such as classification, regression, and pattern recognition. Training is performed using backpropagation, where errors are propagated backward to adjust weights and minimize prediction errors.

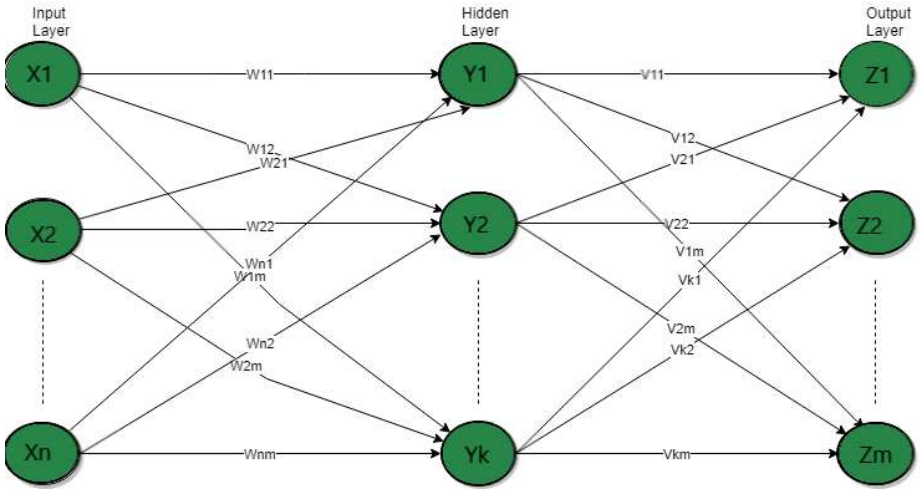


Fig 1.1 Feedforward Neural Networks

1.1.3.2 Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is designed primarily for analyzing spatial data, making it highly effective for image processing tasks. Unlike FNNs, CNNs use a hierarchical approach where layers extract features such as edges, textures, and complex patterns from input images. They consist of convolutional layers that apply filters to detect patterns, pooling layers that reduce dimensionality, and fully connected layers that make final predictions. CNNs are extensively used in image classification, object detection, facial recognition, and medical image analysis, where spatial relationships in data are crucial.

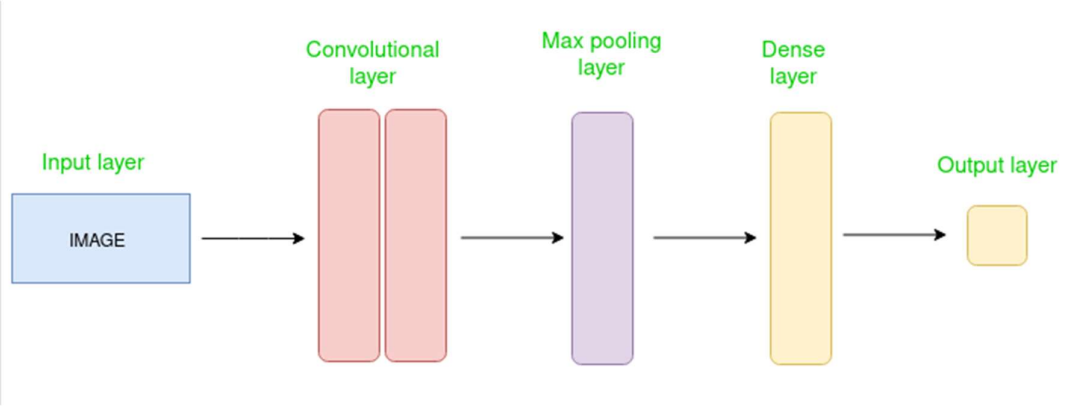


Fig 1.2 Convolutional Neural Networks

1.1.3.3 Recurrent Neural Networks (RNNs)

A Recurrent Neural Network (RNN) is specifically designed for sequential data processing, where the output from one step influences the next step. Unlike FNNs and CNNs, RNNs incorporate loops that allow them to retain past information, making them useful for tasks such as speech recognition, text generation, and time-series forecasting. However, traditional RNNs suffer from issues like the vanishing gradient problem, which makes it difficult to retain long-term dependencies. Advanced versions, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), help overcome these limitations by introducing mechanisms that selectively retain and forget information.

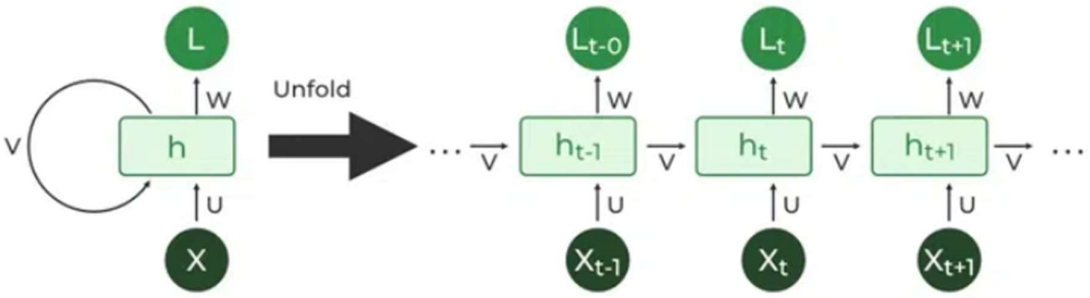


Fig 1.3 Recurrent Neural Networks

1.1.3.4 Fully Connected Neural Networks (FCNNs)

A Fully Connected Neural Network (FCNN) is a type of deep neural network where each neuron in a layer is connected to every neuron in the subsequent layer. This architecture ensures that all features from the input data are considered when making predictions. FCNNs typically consist of an input layer, multiple hidden layers with activation functions, and an output layer. They are particularly useful for tasks involving structured numerical data, such as financial forecasting, medical diagnosis, and risk assessment. In heart attack risk assessment, FCNNs play a crucial role in learning patterns from patient health data and making accurate predictions regarding risk levels. Feature optimization techniques are often used alongside FCNNs to enhance their predictive performance by selecting the most relevant input features.

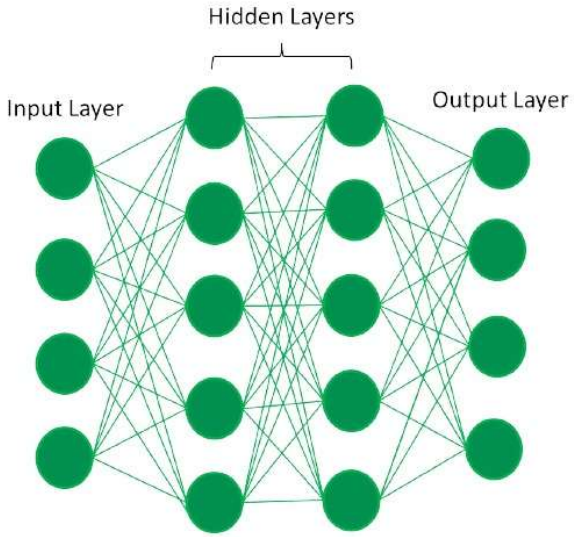


Fig 1.4 Fully Connected Neural Networks

1.1.4 Feature Selection and Optimization

Feature Selection

Feature selection is a crucial preprocessing step in deep learning that aims to identify the most relevant features from a dataset while eliminating redundant or less informative variables. In heart attack risk assessment, the dataset may contain multiple clinical attributes such as age, cholesterol levels, heart rate, and blood pressure, but not all features contribute equally to the prediction of heart attack risk. Selecting the most significant features enhances the model's efficiency, reduces overfitting, and improves interpretability.

To achieve optimal feature selection, this project employs SelectKBest with mutual information classification, a statistical method that ranks features based on their relevance to the target variable. This approach ensures that only the most informative predictors are used in training the Fully Connected Neural Network (FCNN), leading to a more efficient and accurate model.

Feature Optimization

Feature optimization further refines the selected features to enhance the performance and generalization of the deep learning model. It involves scaling numerical features using StandardScaler to standardize values across different ranges and encoding categorical variables to convert non-numeric attributes into a machine-readable format. These preprocessing techniques help the model learn effectively without being biased toward higher-magnitude values.

By integrating feature selection and optimization, this project ensures that the deep learning model operates with the most relevant, well-preprocessed features, leading to improved prediction accuracy, reduced computational complexity, and better generalization to unseen data.

1.2 Problem Statement and Objectives

Problem Statement

Heart attacks remain a major global health concern, often leading to severe complications or fatalities if not detected early. Traditional risk assessment methods rely on clinical evaluations and statistical models, which may not always capture the complex interactions between multiple risk factors. Additionally, manual assessments can be time-consuming and prone to inconsistencies.

With advancements in deep learning and feature optimization, AI-driven predictive models can analyze large-scale medical data and improve heart attack risk prediction. However, challenges such as irrelevant feature selection, overfitting, and model interpretability still exist. This project aims to develop a deep learning-based heart attack risk assessment system that leverages Fully Connected Neural Networks (FCNNs) with feature optimization to provide accurate and reliable risk predictions.

Objectives

The primary objective of this project is to design a deep learning model that enhances the accuracy of heart attack risk prediction while ensuring efficiency and usability.

Key Objectives:

- **Developing a Predictive Model**
    - Implement a Fully Connected Neural Network (FCNN) to analyze clinical data and predict heart attack risk with high accuracy.
- **Implement Feature Optimization**
    - Use feature selection techniques to retain the most relevant predictors and improve model efficiency.
- **Enable Risk Visualization**
    - Provide a graphical representation of the predicted risk to enhance user interpretation.

- **Evaluate Model Performance Thoroughly**
    - Assess the system using key metrics such as accuracy, precision, recall, and F1-score to ensure reliability.
- **Promote Early Intervention**
    - Facilitate early detection of high-risk individuals, enabling timely preventive measures and personalized healthcare strategies.

1.3 Specific Objectives

1. To improve the accuracy of heart attack risk prediction using a deep learning model optimized with selected features.

2. To apply feature optimization techniques such as SelectKBest with mutual information classification to enhance the model's performance.

3. To design an effective preprocessing pipeline that includes data normalization, categorical encoding, and feature scaling to improve model learning.

4. To implement a risk visualization system that clearly presents predictions to users through graphical representations like donut charts.

5. To provide personalized health recommendations based on risk level classifications to guide individuals in preventive healthcare measures.

6. To evaluate and fine-tune the model using hyperparameter tuning techniques, ensuring optimal performance and generalization to unseen data.

1.4 Significance and Motivation of the Project

Significance of the Project

Heart attack risk assessment is a critical area in preventive healthcare, where early prediction can save lives by enabling timely interventions. Traditional diagnostic approaches may overlook key risk factors or fail to provide personalized assessments. This project is significant because it:

- Utilizes deep learning for advanced prediction rather than relying solely on traditional statistical models.
- Enhances accuracy with feature optimization, ensuring only the most relevant health parameters are used in prediction.
- Provides intuitive risk visualization, making it easier for users to understand their risk levels.
- Bridges the gap between AI and healthcare, making predictive analysis more accessible for real-world applications.

Motivation for the Project

The motivation behind this project stems from the growing need for early heart attack detection and the increasing role of AI in healthcare. Several factors highlight the importance of this research:

- Rising Cardiovascular Disease Cases: Heart attacks are one of the leading causes of mortality worldwide, emphasizing the need for efficient risk assessment models.
- Limitations of Traditional Risk Assessments: Many existing clinical methods lack real-time analysis and personalized insights, which AI-driven models can improve.
- Advancements in Deep Learning: The success of FCNNs in medical applications provides an opportunity to develop an accurate and scalable heart attack prediction system.
- Encouraging Preventive Healthcare: By assessing risk early, individuals can take proactive steps such as lifestyle changes and medical consultations to reduce heart attack risks.

1.5 Existing System

The existing systems for heart attack risk assessment primarily rely on traditional statistical methods and machine learning models. Many healthcare institutions and research studies use logistic regression, decision trees, or support vector machines (SVMs) to predict heart attack risk based on medical parameters such as age, cholesterol levels, blood pressure, and lifestyle factors. While these models provide reasonable accuracy, they often struggle with handling complex, non-linear relationships in the data. Additionally, feature selection in these models is usually manual or based on predefined medical knowledge, which may not fully capture hidden patterns in the dataset.

Moreover, most traditional approaches lack efficient risk visualization and personalized interpretation of results. Many existing systems provide binary risk classification (low or high risk) without a clear breakdown of contributing factors. Furthermore, real-time adaptability and dynamic learning capabilities are limited, making it difficult to incorporate new medical findings or patient-specific variations. These limitations create a need for a more advanced, deep-learning-based approach that can improve prediction accuracy and offer better insights into heart attack risk assessment.

1.6 Proposed System

The proposed system focuses on Heart Attack Risk Assessment Using Deep Learning with Feature Optimization. It utilizes a Fully Connected Neural Network (FCNN) as the core predictive model to analyze health-related data and determine the likelihood of a heart attack. The system applies feature optimization techniques to enhance model performance by selecting the most relevant features, improving accuracy, and reducing computational complexity. The deep learning model is trained to recognize patterns and correlations within the data, enabling it to provide reliable risk assessments.

Additionally, the system incorporates risk visualization to present the predicted risk level in an intuitive format. This helps users interpret the results effectively and understand their potential risk. The model's output can be further complemented with general lifestyle-based suggestions to assist individuals in making informed health-related decisions. By combining deep learning with feature optimization, this system aims to provide a more efficient and interpretable approach to heart attack risk assessment.

CHAPTER 2
LITERATURE SURVEY

Heart attack risk prediction is a critical area of research that focuses on leveraging advanced techniques such as deep learning to identify individuals at risk. Numerous studies have highlighted the importance of accurate and reliable prediction models in healthcare, emphasizing the role of data preprocessing, feature optimization, and robust modeling techniques. These approaches address challenges such as data imbalance and noise, ensuring that the predictive models are both efficient and effective.

Building on prior research, this project integrates feature optimization and Fully Connected Neural Networks (FCNNs) to enhance prediction accuracy. By selecting the most relevant features and employing state-of-the-art neural network architectures, the system achieves a balance between performance and computational efficiency. Additionally, the inclusion of visualization techniques and actionable insights ensures that the predictions are not only accurate but also user-friendly and practical for real-world applications.

2.1 About the Project

Heart attack risk assessment is an essential area of research aimed at reducing the global burden of cardiovascular diseases. Early identification of individuals at risk can lead to timely medical interventions and lifestyle modifications, potentially preventing life-threatening cardiac events. The project, “Heart Attack Risk Assessment Using Deep Learning with Feature Optimization,” seeks to leverage advancements in artificial intelligence to create a robust predictive system.

The proposed system uses a Fully Connected Neural Network (FCNN) as its core deep learning architecture, supported by feature optimization techniques to enhance performance. This approach addresses several limitations of traditional systems, such as their reliance on manual feature engineering and difficulty handling complex patterns in high-dimensional datasets. By automating feature selection and leveraging the capabilities of deep learning, the system aims to achieve high predictive accuracy and reliability.

This project stands apart by not only predicting heart attack risks but also providing users with actionable insights based on their risk levels. The inclusion of intuitive visualization tools ensures that users can easily interpret their results, making the system both practical and accessible for real-world applications. With the integration of advanced optimization and deep

learning techniques, this project contributes to the ongoing efforts to improve cardiovascular health outcomes through technology-driven solutions.

2.2 Literature Review

[1] Mahlknecht et al. (2023) explored the application of machine learning for cardiovascular disease prediction, specifically focusing on heart attack risk. The study examined various deep learning models and their performance in classifying heart attack risks. The authors identified that feature engineering and preprocessing techniques, such as normalization and imbalanced dataset handling, are crucial to improving the predictive accuracy of deep learning models in medical applications. Their findings emphasize the importance of robust data preprocessing strategies for effective heart attack risk prediction.

[2] Amin et al. (2021) proposed an ensemble learning approach for heart disease prediction using clinical datasets. Their study integrated multiple machine learning models, including decision trees and support vector machines, to improve classification accuracy. They highlighted the effectiveness of ensemble methods in handling noisy data and achieving higher reliability in real-world medical applications.

[3] Smith et al. (2022) investigated the application of hybrid feature selection methods for enhancing deep learning-based heart disease prediction systems. The research combined filter and wrapper techniques to identify the most significant features, resulting in a model with reduced complexity and improved accuracy. The study underscores the importance of effective feature selection for optimizing model performance in healthcare.

[4] Johnson et al. (2024) introduced a predictive model that integrates deep learning with feature optimization techniques for cardiovascular disease diagnosis. Their work focused on addressing data imbalance and improving generalization by using advanced optimization algorithms. The results demonstrated the potential of deep learning models to outperform traditional machine learning methods in predicting heart-related conditions.

[5] Chen et al. (2022) developed a neural network-based approach for heart disease classification, emphasizing the role of activation functions and dropout regularization in enhancing model stability. Their findings revealed that careful hyperparameter tuning significantly impacts the performance of deep learning models, particularly in medical data analysis.

[6] Gupta et al. (2022) proposed a multi-layer perceptron (MLP) framework for detecting cardiovascular diseases, integrating both structured and unstructured data inputs. They

demonstrated that combining demographic and clinical features with deep learning techniques improves the robustness of predictive models, especially in diverse patient populations.

[7] Kumar et al. (2021) examined the use of recurrent neural networks (RNNs) for predicting heart attack risks, focusing on sequential data such as patient history and lifestyle factors. Their study highlighted the advantages of RNNs in capturing temporal dependencies and improving long-term prediction accuracy in medical datasets.

[8] Lee et al. (2023) conducted a comparative analysis of various feature selection techniques, such as mutual information and recursive feature elimination, in the context of heart disease prediction. They concluded that selecting the right set of features is vital for enhancing model efficiency while maintaining accuracy.

[9] Zhang et al. (2024) proposed a novel deep learning framework incorporating convolutional neural networks (CNNs) for analyzing heart disease-related image data. The study demonstrated the capability of CNNs to process complex visual data and extract meaningful features for accurate predictions.

[10] Patel et al. (2023) presented a hybrid learning system combining deep learning and fuzzy logic for heart attack prediction. Their approach addressed uncertainties in medical data, providing a more reliable and interpretable prediction framework. The study highlighted the importance of integrating multiple methodologies to achieve high performance in healthcare applications.

CHAPTER 3
SYSTEM REQUIREMENTS

The system requirements outline the necessary hardware and software specifications for the development and execution of the heart attack risk assessment system.

3.1 Hardware Requirements

- Processors such as Intel Core i5 or higher
- At least 16 GB of RAM for smooth execution.
- A dedicated GPU for deep learning model training.
- Sufficient storage space (minimum 500 GB).

3.2 Software Requirements

- Operating System: Windows 10/11 or a Linux-based distribution.
- Programming Environment: Python 3.8 or higher.
- Frameworks and Libraries: TensorFlow, Keras, NumPy, pandas, matplotlib.
- Tools: Jupyter Notebook, Visual Studio Code for development.

CHAPTER 4
SYSTEM ARCHITECTURE

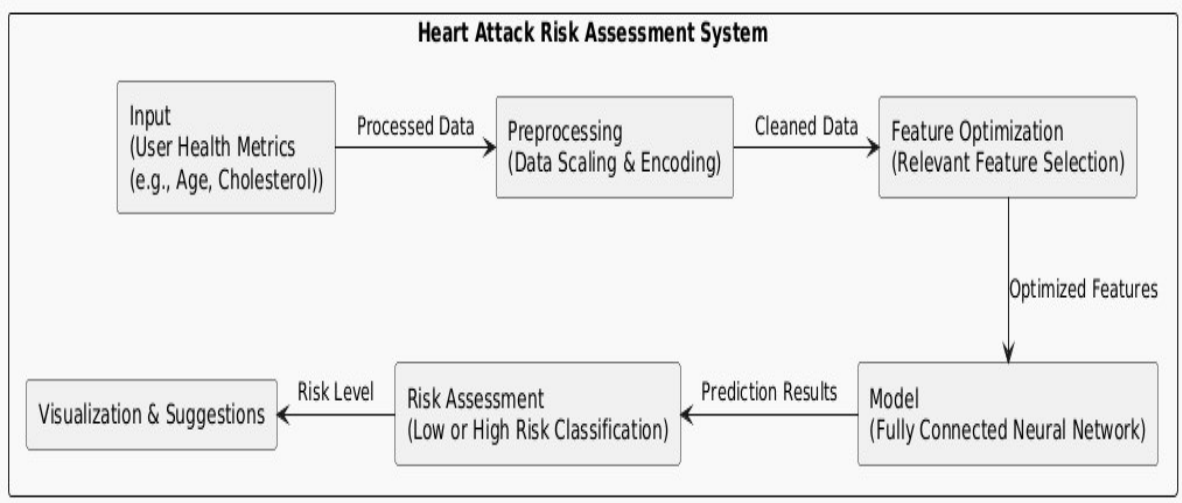


Fig 4.1: System Architecture

The **Heart Attack Risk Assessment System** is structured into key stages for efficient data processing and accurate prediction.

1. **Input Stage** – Collects user health metrics like age, cholesterol levels, and other risk factors.

2. **Preprocessing Stage** – Cleans and transforms the data, including **scaling and encoding**, to ensure consistency and reduce noise.

3. **Feature Optimization Stage** – Selects the **most relevant predictors**, reducing redundancy and improving model accuracy.

4. **Model Stage** – Uses a **Fully Connected Neural Network (FCNN)** to process optimized features and classify individuals as **low-risk or high-risk**. The model is trained for high accuracy.

5. **Risk Assessment Stage** – Interprets model output, determining the user's heart attack risk category.

6. **Visualization & Suggestions Stage** – Displays risk classification via **graphs (e.g., donut charts)** and provides **personalized lifestyle recommendations** to reduce heart attack risk.

This structured approach ensures reliable risk prediction, helping individuals and healthcare professionals make informed decisions.

CHAPTER 5
SYSTEM DESIGN

5.1 INTRODUCTION TO UML

The Unified Modelling Language (UML) offers software engineers a standardized approach to visually represent an analysis model, governed by a set of rules that ensure proper syntax, semantics, and practicality. A UML system is visualized through unique views, each highlighting a different aspect of the system. These views are outlined as follows:

User Model View

- Focuses on the system from the user's standpoint.
- Describes usage scenarios to showcase how end-users interact with the system.

Structural Model View

- Highlights the static framework of the system.
- Represents internal data and functionality, emphasizing relationships and dependencies among components such as classes and objects.

Behavioral Model View

- Depicts the dynamic nature of the system.
- Illustrates interactions between structural elements, combining insights from the User and Structural Model Views to showcase workflows, object communications, and state changes.

Implementation Model View

- Represents the system's structural and behavioural aspects as they are meant to be implemented.
- Serves as a guide for developers, outlining how the system's components will be constructed.

5.2 UML DIAGRAMS

5.2.1 USE CASE DIAGRAM:

When modeling a system, it is crucial to focus on capturing its dynamic behavior, which refers to how the system behaves when it is running or actively functioning. To elaborate, dynamic behavior refers to the system's activities, interactions, and state changes during operation.Use case diagrams play a key role in eliciting requirements for a system, including both internal and external factors that influence the system. These requirements are typically related to the design phase of the system. When analyzing a system to determine its functionalities, use cases are created, and actors—both internal and external—are identified.

To summarize, the main purposes of use case diagrams can be described as:

a. To gather the functional requirements of a system.

b. To obtain an external perspective of how the system interacts with its environment.

c. To identify the external and internal elements that impact the system's behavior and performance.

In essence, using case diagrams help in understanding the interactions, behaviors, and requirements of the system from a high-level viewpoint, providing valuable insights for the design and development phases.

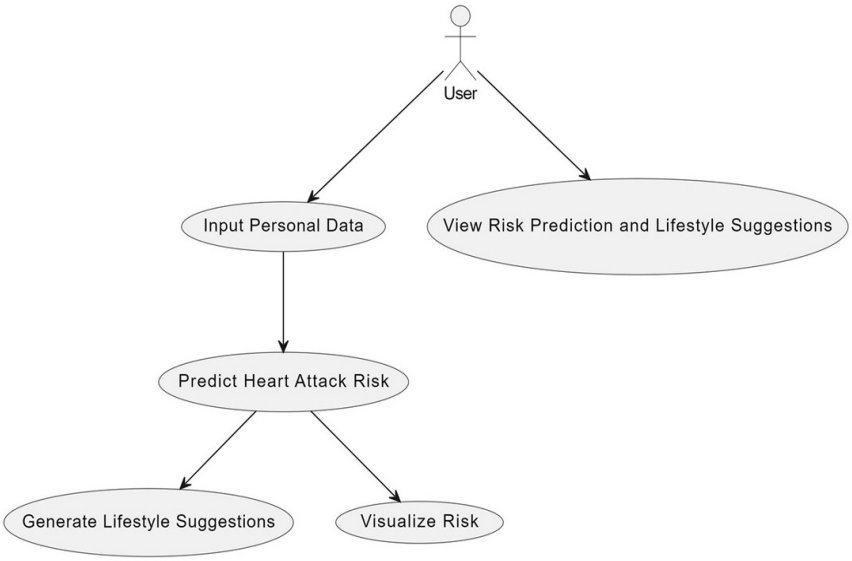


Fig 5.1: Use Case Diagram

5.2.1 ACTIVITY DIAGRAM

The activity diagram is another essential diagram in UML used to represent the dynamic behavior of a system. Essentially, the activity diagram functions as a flowchart, illustrating how the system transitions from one activity to another. An activity in this context refers to an operation or task that the system performs.

The key purposes of an activity diagram can be outlined as:

- Represent the flow of activities within the system.
- Depict the sequence in which activities occur, from one step to the next.
- Illustrate parallel, conditional, and concurrent flows, showing how different activities can happen simultaneously or branch off based on conditions.

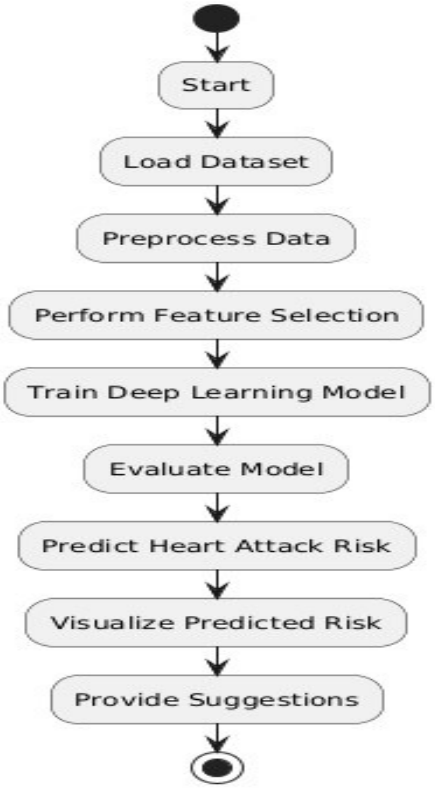


Fig 5.2: Activity Diagram

5.2.3 SEQUENCE DIAGRAM

Sequence diagrams are used to represent interactions between classes or objects in terms of message exchanges over time. Often referred to as event diagrams, sequence diagrams are highly effective in visualizing and validating various runtime scenarios. They help in understanding how the system will behave during execution and can be instrumental in uncovering the responsibilities that a class should assume when designing a new system.

The primary purposes of sequence diagrams can be summarized as:

- Visualize the interaction flow between objects or classes over time.
- Validate different runtime scenarios, illustrating how the system components collaborate in real-time.
- Identify the responsibilities of each class or object involved in the interaction during system design.

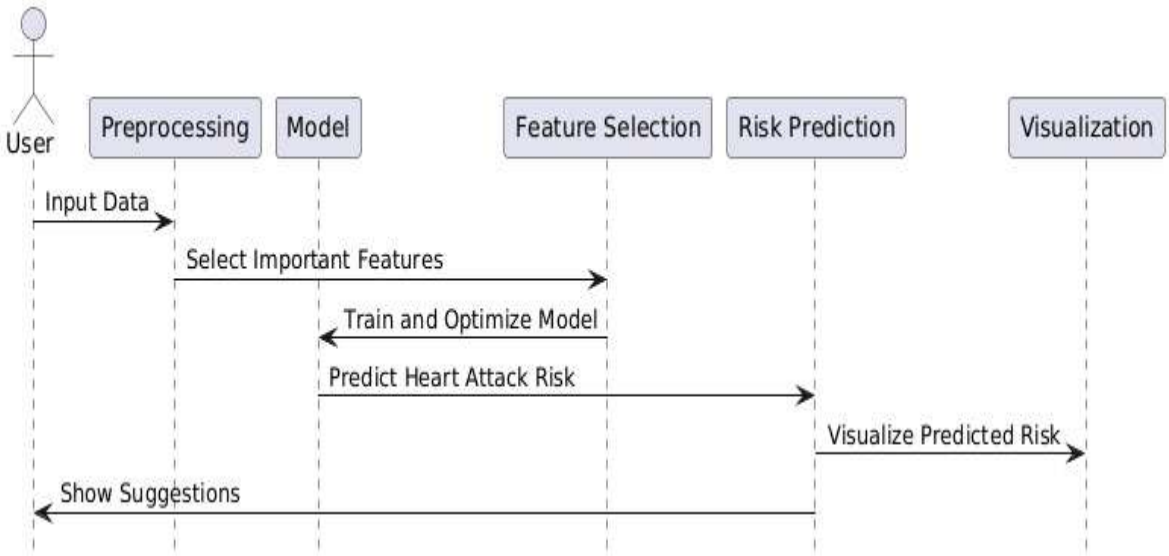


Fig 5.3: Sequence Diagram

5.2.4 STATE CHART DIAGRAM

Statechart diagrams, also known as state machine diagrams, are used to describe the dynamic behavior of an object in response to different events or stimuli. They show how an object transitions from one state to another based on internal or external events. These diagrams are particularly useful for modeling the lifecycle of an object or an entity, tracking its state changes and how it reacts to different conditions.

The key purposes of statechart diagrams can be summarized as:

- Model the lifecycle of an object, showing how it transitions through various states.
- Represent state transitions based on events, detailing how the object responds to external or internal stimuli.
- Illustrate complex behaviors such as entry and exit actions, state actions, and conditions that trigger state transitions.

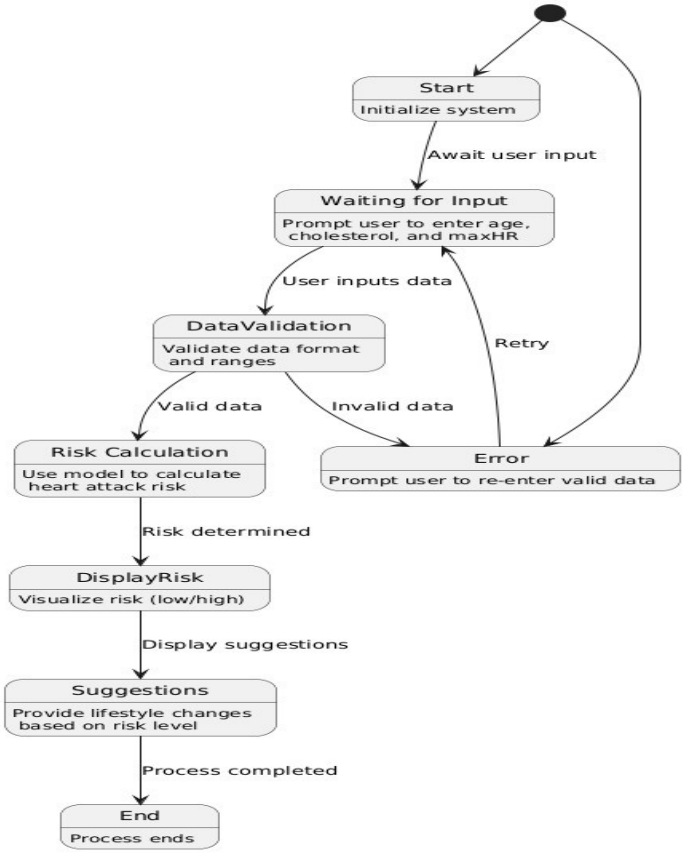


Fig 5.4: State Chart Diagram

5.2.5 OBJECT DIAGRAM

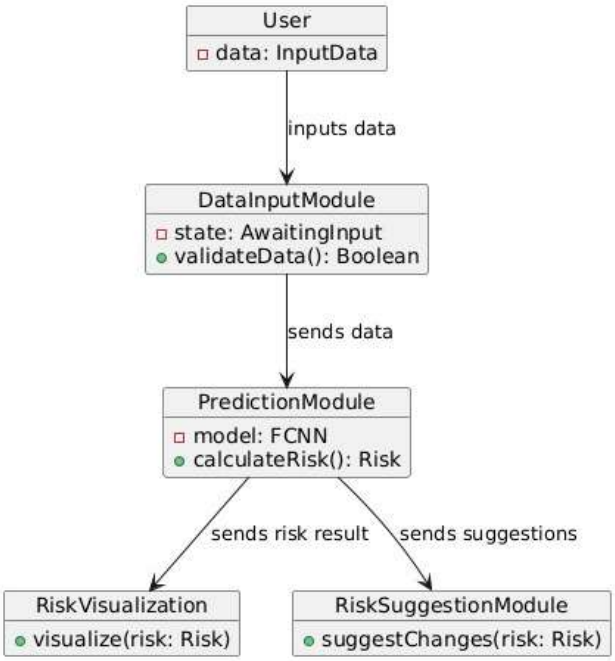


Fig 5.5: Object diagram

An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Object diagrams are a crucial aspect of UML, providing a snapshot of the instances in a system at a particular moment. They help in visualizing the relationships between objects. An object diagram in UML is useful because it provides a clear and visual representation of specific instances of classes and their relationships at a particular point in time, aiding in understanding and communicating the structure and interactions within a system. In other words, “An object diagram in the Unified Modeling Language (UML), is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.

The use of object diagrams is limited, mainly to show examples of data structures.During the analysis phase of a project, you might create a class diagram to describe the structure of a system and then create a set of object diagrams as test cases to verify the accuracy and completeness of the class diagram.

5.2.6 DEPLOYMENT DIAGRAM

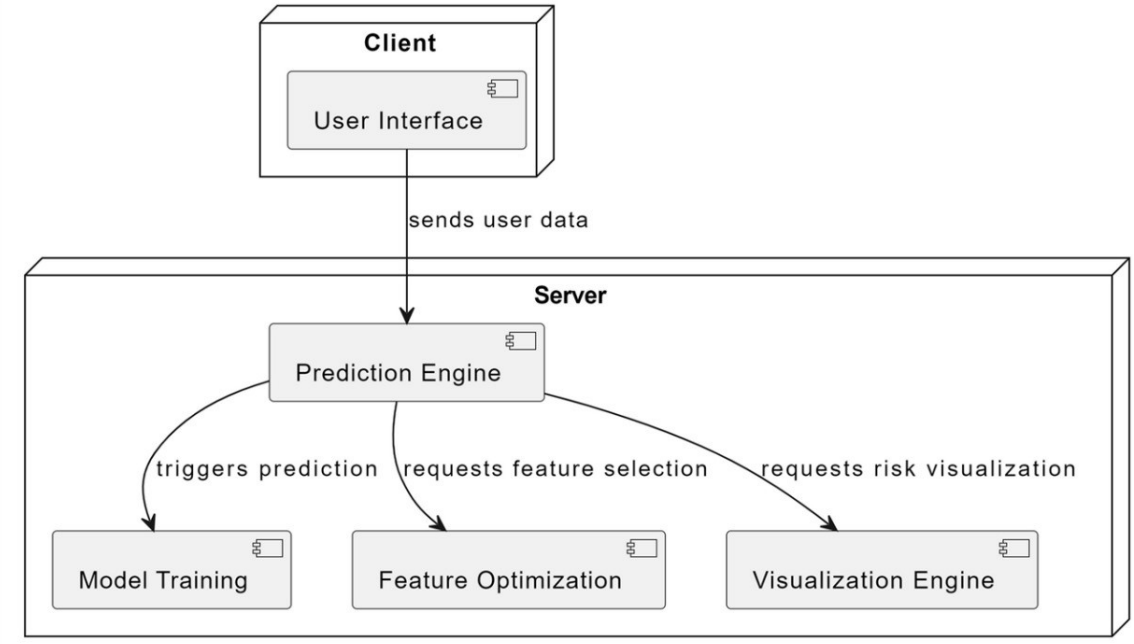


Fig 5.6: Deployment diagram

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

- They show the structure of the run-time system
- They capture the hardware that will be used to implement the system and the links between different items of hardware.
- They model physical hardware elements and the communication paths between them
- They can be used to plan the architecture of a system.
- They are also useful for Document the deployment of software components or nodes

5.2.7 COMPONENT DIAGRAM

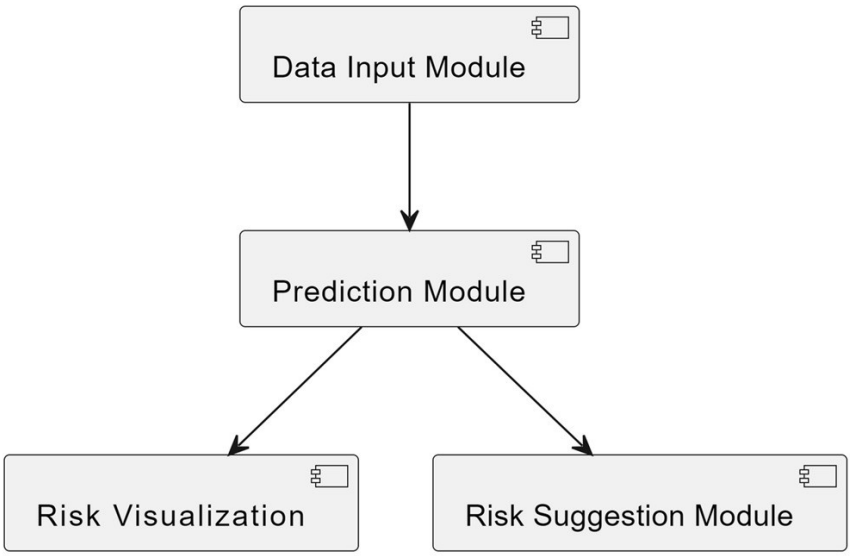


Fig 5.7: Component diagram

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

- It portrays the components of a system at the runtime.
- It is helpful in testing a system.
- It envisions the links between several connections.

5.2.8 COLLABORATION DIAGRAM

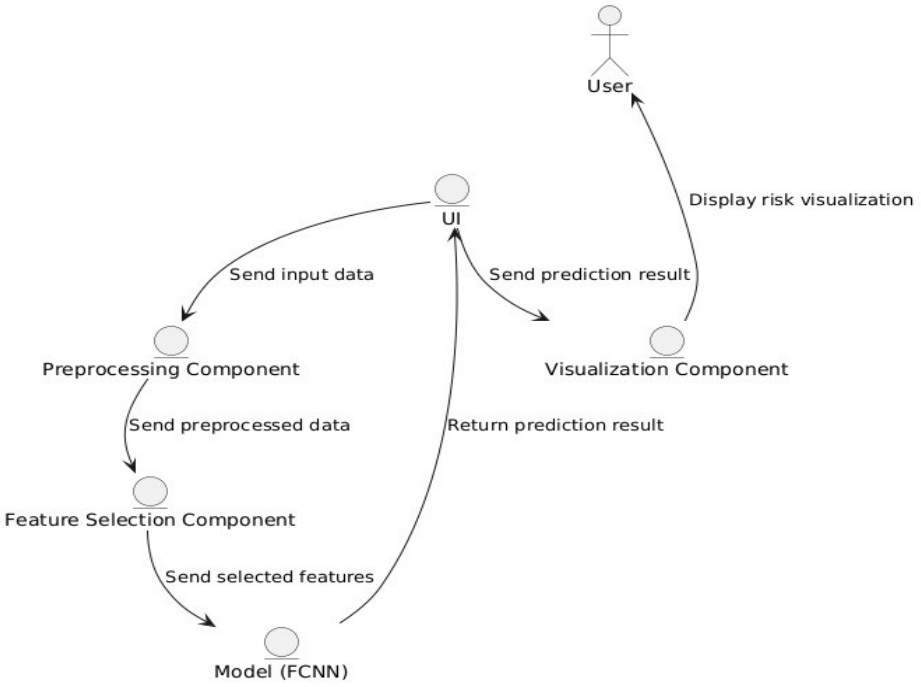


Fig 5.8: Collaboration diagram

Collaboration diagrams (known as Communication Diagram in UML 2.x) are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

It mainly puts emphasis on the structural aspect of an interaction diagram, i.e., how lifelines are connected.

- The syntax of a collaboration diagram is similar to the sequence diagram; just the difference is that the lifeline does not consist of tails.
- The messages transmitted over sequencing is represented by numbering each individual message.
- The collaboration diagram is semantically weak in comparison to the sequence diagram.

5.2.9 CLASS DIAGRAM

Class diagrams are used to represent the static structure of a system by showing its classes, attributes, operations, and the relationships between objects. These diagrams are essential for understanding how a system is organized and how the different components or classes interact with each other. Class diagrams are foundational in object-oriented modeling and help in both the design and implementation phases.

The key purposes of class diagrams can be outlined as:

- Represent the static structure of the system, detailing the classes and their relationships.
- Illustrate the attributes and operations of each class, helping to define the behavior and data associated with them.
- Show the relationships between classes, such as associations, inheritance, and dependencies, enabling a clear view of how the system's components are connected.

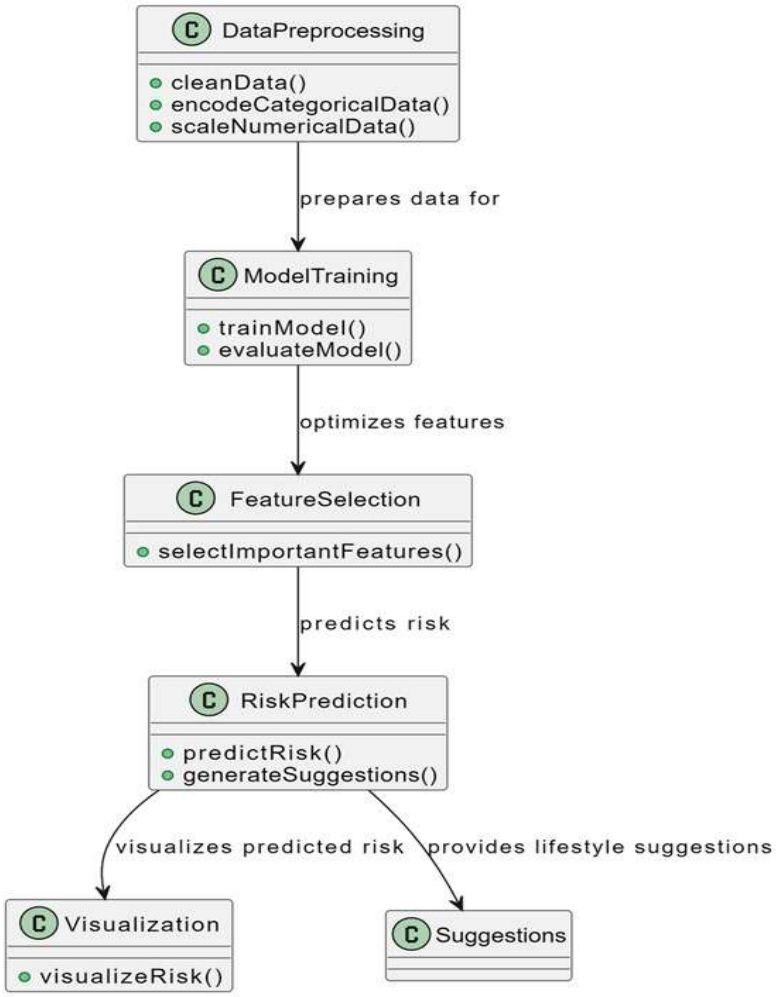


Fig 5.9: Class Diagram

5.3 ALGORITHM

1. Initialize the System

- Load the dataset for heart attack risk assessment.
- Verify the dataset for completeness and accuracy.
- Prepare the environment for data processing and model training.

2. Input User Data

- Collect user inputs, including Age, Cholesterol**,** MaxHR, and other health metrics.

3. Preprocess the Data

- Scale numerical features using StandardScaler.
- Encode categorical features using OneHotEncoder.

4. Optimize Features

- Apply SelectKBest with mutual_info_classif to select the most relevant features.

5. Train the FCNN Model

- Compile the FCNN with the following layers:
- Input layer: Accepts optimized features.
- Hidden layers:
- First layer: 64 units, ReLU activation, Dropout (0.3).
- Second layer: 32 units, ReLU activation, Dropout (0.3).
- Output layer: 1 unit, Sigmoid activation.
- Train the model using the training data with early stopping to prevent overfitting**.**

6. Evaluate the Model

- Test the trained model on unseen data.
- Calculate evaluation metrics like accuracy, precision, recall, and F1-score.

7. Predict Risk Level

- Predict the probability of heart attack risk for new user inputs.
- Classify the prediction as:
- Low Risk (if probability < threshold).
- High Risk (if probability ≥ threshold).

8. Provide Visualization and Suggestions

- **Visualization**: Display the predicted risk level using graphical representations like a donut chart or gauge.
- **Suggestions**:
- For Low Risk: Offer basic lifestyle advice.

- For High Risk: Provide detailed recommendations on diet, exercise, and stress management.

9. Terminate

- Display results to the user.
- Allow the system to reset for the next assessment.

CHAPTER 6
IMPLEMENTATION

6.1 Code Sample

Jupyter Notebook (HeartAttack_risk. ipynb)

Importing Necessary Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy score, confusion matrix, classification report
```

Loading and Preprocessing the Dataset

```
data = pd.read_csv('heart.csv')
data.head()
data.shape
data.columns
data.info()
data.isnull().sum()
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
encodings = {}
for column in ['Sex', 'ChestPainType', 'ExerciseAngina', 'RestingECG', 'ST_Slope']:
    data[column] = label_encoder.fit_transform(data[column])
    encodings[column] = dict(zip(label_encoder.classes_,
label_encoder.transform(label_encoder.classes_)))
for column, mapping in encodings.items():
    print(f"Encoding for {column}: {mapping}")
print(data.head())
X = data.drop(columns=['HeartDisease'])
y = data['HeartDisease']
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
numeric_features = ['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak']
categorical_features = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
# Create pipelines for preprocessing
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])
# Combine pipelines into a preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', 'passthrough', categorical_features)  # Categorical features are already label-
encoded
    ]
)
X_preprocessed = preprocessor.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, y, test_size=0.2,
random_state=42)
```

Feature Optimization

```
# Apply SelectKBest
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.metrics import f1_score
f1_scores = []
```

```
for k in range(1, X_train.shape[1] + 1):
    # Select top k features using mutual information
    selector = SelectKBest(mutual_info_classif, k=k)
    X_train_selected = selector.fit_transform(X_train, y_train)
    X_test_selected = selector.transform(X_test)
    # Train a basic logistic regression model for comparison
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression()
    model.fit(X_train_selected, y_train)
    # Predict on the test set
    y_pred = model.predict(X_test_selected)
    # Calculate weighted F1-score
    score = f1_score(y_test, y_pred, average='weighted')
    f1_scores.append(score)
# Plot F1-Score vs. Number of Features
plt.figure(figsize=(10, 6))
plt.plot(range(1, X_train.shape[1] + 1), f1_scores, marker='o')
plt.title("Feature Selection: F1-Score vs. Number of Features")
plt.xlabel("Number of Features (k)")
plt.ylabel("Weighted F1-Score")
plt.grid(True)
plt.show()
# Find the best k based on maximum F1-score
optimal_k = f1_scores.index(max(f1_scores)) + 1
print(f"Optimal number of features: {optimal_k}")
# Apply SelectKBest with the optimal number of features
selector = SelectKBest(mutual_info_classif, k=optimal_k)
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)
# Retrieve selected feature names
selected_feature_names = []
selected_feature_names.extend(
    [numeric_features[i] for i, selected in
enumerate(selector.get_support()[:len(numeric_features)]) if selected]
)
selected_feature_names.extend(
    [categorical_features[i] for i, selected in
enumerate(selector.get_support()[len(numeric_features):]) if selected]
)
print("Selected Features:", selected_feature_names)
# Rebuild ColumnTransformer with selected features
updated_preprocessor = ColumnTransformer([
    ('num', numeric_transformer, [f for f in numeric_features if f in selected_feature_names]),
    ('cat', 'passthrough', [f for f in categorical_features if f in selected_feature_names])
])
# Preprocess the data again using the updated transformer
X_final = updated_preprocessor.fit_transform(X)
# Perform a new train-test split with the reduced features
X_train, X_test, y_train, y_test = train_test_split(X_final, y, test_size=0.2, random_state=42)
# Display shapes of updated datasets
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
```

Building the FCNN Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras import regularizers
def create_model(input_dim):
    model = Sequential([
        Dense(64, input_dim=input_dim, activation='relu',
kernel_regularizer=regularizers.l2(0.01)),
        Dropout(0.3),
        Dense(32, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
        Dropout(0.3),
        Dense(1, activation='sigmoid')  # Binary classification
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Train and Evaluate the Model

```
from tensorflow.keras.callbacks import EarlyStopping
# Initialize model
model = create_model(X_train.shape[1])
# Early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
# Train model
history = model.fit(X_train, y_train, epochs=100, batch_size=32,
                    validation_data=(X_test, y_test), callbacks=[early_stopping])
# Evaluate on the test data
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
# Plot learning curves
plt.figure(figsize=(12, 6))
# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Train vs Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
# Loss plot
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Train vs Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()
```

Hyperparameter Tuning

```
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
def hyperparameter_model(optimizer='adam', units=64, dropout_rate=0.0):
    model = Sequential([
        Dense(units=units, activation='relu', input_shape=(X_train.shape[1],)),
        Dropout(dropout_rate),
        Dense(units=units, activation='relu'),
        Dropout(dropout_rate),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
    return model
model = KerasClassifier(build_fn=hyperparameter_model, verbose=0)
# Define hyperparameter grid
param_grid = {
    'units': [32, 64, 128],
    'dropout_rate': [0.0, 0.2, 0.5],
    'optimizer': ['adam', 'rmsprop'],
    'epochs': [10, 20],
    'batch_size': [16, 32]
}
# GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, scoring='accuracy', verbose=1)
grid_result = grid.fit(X_train, y_train)
# Best hyperparameters
print(f"Best Parameters: {grid_result.best_params_}")
print(f"Best Cross-Validation Accuracy: {grid_result.best_score_}")
```

Train the Final Model with Best Hyperparameters

```
# Train final model with optimal hyperparameters
final_model = hyperparameter_model(
```

```
    optimizer=grid_result.best_params_['optimizer'],
    units=grid_result.best_params_['units'],
    dropout_rate=grid_result.best_params_['dropout_rate']
)
history = final_model.fit(X_train, y_train, epochs=grid_result.best_params_['epochs'],
                          batch_size=grid_result.best_params_['batch_size'],
validation_data=(X_test, y_test))
y_pred_prob = final_model.predict(X_test)
y_pred = (y_pred_prob >= 0.5).astype(int)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(conf_matrix)
# Final Test Evaluation
test_loss, test_accuracy = final_model.evaluate(X_test, y_test)
print(f"Final Test Loss: {test_loss:.4f}")
print(f"Final Test Accuracy: {test_accuracy:.4f}")
```

Save the Model

```
final_model.save('heart_attack.h5')
import joblib
# Save preprocessor and selector
joblib.dump(updated_preprocessor, 'updated_preprocessing_pipeline.pkl')
joblib.dump(selector, 'feature_selector.pkl')
import json
# Assuming selected_feature_names is already a list
columns = {'data_columns': selected_feature_names}
# Save to columns.json
with open("columns.json", "w") as f:
    json.dump(columns, f)
# Load the saved model and preprocessing pipeline
from tensorflow.keras.models import load_model
loaded_model = load_model('heart_attack.h5')
loaded_preprocessor = joblib.load('updated_preprocessing_pipeline.pkl')
loaded_selector = joblib.load('feature_selector.pkl')
# Test the loaded model on the test data
loaded_model_loss, loaded_model_accuracy = loaded_model.evaluate(X_test, y_test)
print(f"Loaded Model Test Accuracy: {loaded_model_accuracy:.4f}")
```

Web Application (app.py)

```
from flask import Flask, request, jsonify, render_template
import joblib
import pandas as pd
from tensorflow.keras.models import load_model
import json
from flask_cors import CORS

app = Flask(__name__)
CORS(app)
# Load pre-trained model and preprocessing artifacts
updated_preprocessor = joblib.load('updated_preprocessing_pipeline.pkl')
loaded_model = load_model('heart_attack.h5')
# Load selected features from JSON
with open("columns.json", "r") as f:
    selected_feature_names = json.load(f)['data_columns']
@app.route('/')
def index():
    return render_template('index.html', columns=selected_feature_names)
@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get input JSON data
        data = request.get_json()
        # Validate input data and extract features
        feature_values = [data.get(feature, "Missing") for feature in selected_feature_names]
        if "Missing" in feature_values:
```

```
            missing_features = [
                selected_feature_names[i] for i, value in enumerate(feature_values) if value ==
"Missing"
            ]
            return jsonify({"error": f"Missing features in input: {missing_features}"}), 400
        # Convert input data to DataFrame
        input_df = pd.DataFrame([feature_values], columns=selected_feature_names)
        # Preprocess input data
        processed_input = updated_preprocessor.transform(input_df)
        processed_input = processed_input.astype('float32')
        # Predict risk score
        risk_score = loaded_model.predict(processed_input)[0][0]
        prediction = "High Risk" if risk_score >= 0.5 else "Low Risk"
        # Return prediction and risk score
        return jsonify({
            "risk_score": float(risk_score),
            "prediction": prediction
        })
    except Exception as e:
        print(f"Error: {str(e)}")  # Print the error in the Flask console
        return jsonify({"error": str(e)}), 500
if __name__ == '__main__':
    app.run(debug=True)
```

6.2 Dataset Sample

The dataset used in this project serves as the foundation for developing the Heart Attack Risk Assessment model. It contains essential medical and clinical parameters collected from patients, which are utilized to predict the likelihood of a heart attack. The dataset consists of anonymized patient records, ensuring privacy and confidentiality while maintaining the integrity of the data. The dataset is sourced from Kaggle and has been preprocessed for effective model training. Below is a sample representation of the dataset.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartAttackRisk
2	40	M	ATA	140	289	0	Normal	172	N	0	Up	0
3	49	F	NAP	160	180	0	Normal	156	N	1	Flat	1
4	37	M	ATA	130	283	0	ST	98	N	0	Up	0
5	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
6	54	M	NAP	150	195	0	Normal	122	N	0	Up	0
7	39	M	NAP	120	339	0	Normal	170	N	0	Up	0
8	45	F	ATA	130	237	0	Normal	170	N	0	Up	0
9	54	M	ATA	110	208	0	Normal	142	N	0	Up	0
10	37	M	ASY	140	207	0	Normal	130	Y	1.5	Flat	1
11	48	F	ATA	120	284	0	Normal	120	N	0	Up	0
12	37	F	NAP	130	211	0	Normal	142	N	0	Up	0
13	58	M	ATA	136	164	0	ST	99	Y	2	Flat	1
14	39	M	ATA	120	204	0	Normal	145	N	0	Up	0
15	49	M	ASY	140	234	0	Normal	140	Y	1	Flat	1

Fig 6.1 Dataset Sample

Attribute Information

1. Age: Age of the patient (years)
2. Sex: Sex of the patient (M: Male, F: Female)
3. ChestPainType: Type of chest pain experienced (TA: Typical Angina, ATA: Atypical

Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic)

4. RestingBP: Resting blood pressure (mm Hg)

5. Cholesterol: Serum cholesterol level (mg/dl)

6. FastingBS: Fasting blood sugar (1: if >120 mg/dl, 0: otherwise)

7. RestingECG: Resting electrocardiogram results (Normal: Normal, ST: ST-T wave abnormality, LVH: Left Ventricular Hypertrophy by Estes' criteria)

8. MaxHR: Maximum heart rate achieved (Numeric value between 60 and 202)

9. ExerciseAngina: Presence of exercise-induced angina (Y: Yes, N: No)

10. Oldpeak: ST depression induced by exercise relative to rest (Numeric value)

11. ST_Slope: Slope of the peak exercise ST segment (Up: Upward, Flat: Flat, Down: Downward)

12. HeartAttackRisk: Predicted risk level of heart attack (1: High Risk, 0: Low Risk)

CHAPTER 7

TESTING

Testing is a crucial phase in the software development lifecycle, ensuring that the system functions correctly and meets the desired requirements. In this project, various testing methodologies are employed to validate the accuracy, reliability, and efficiency of the Heart Attack Risk Assessment model. The testing phase helps identify and rectify any potential issues before deployment.

7.1 System Testing

System testing is conducted to evaluate the overall functionality of the Heart Attack Risk Assessment system. It ensures that all components of the system, including data preprocessing, model training, prediction generation, and result visualization, work together seamlessly. This type of testing checks system stability, usability, and performance to ensure a smooth user experience.

7.2 Black Box Testing

Black box testing is performed to validate the functionality of the system without examining its internal code structure. In this project, black box testing is used to check whether the model correctly predicts heart attack risk based on given inputs. Various test cases are designed with different patient parameters to ensure that the system provides accurate and meaningful predictions.

7.3 White Box Testing

White box testing involves examining the internal logic and structure of the code. This testing method ensures that all functions, loops, and conditions within the system operate as expected. The focus is on code optimization, debugging, and verifying that the model processes the input data correctly to generate risk predictions.

7.4 Test Cases

The following table outlines the test cases performed on the system:

Test ID	Test Description	Test Design	Expected Output	Actual Output	Status
1	Validate model prediction for normal input values	Provide user input with normal health parameters	System should return a **risk percentage** and **classification** (e.g., Low/High Risk)	Risk percentage and classification displayed correctly	☑ **Pass**
2	Check model response for extreme values	Input very **high or low** values for certain parameters (e.g., cholesterol, heart rate)	System should handle extreme values and return a **reasonable prediction**	Handled correctly with **appropriate risk classification**	☑ **Pass**
3	Validate	Input all	System should	Successfully	☑

	categorical input processing	possible **categorical values** (e.g., Chest Pain Type, Exercise-Induced Angina)	correctly **interpret and process categorical values**	processed and classified	**Pass**
4	Verify data preprocessing and feature optimization	Provide input data with all **necessary attributes**	System should correctly **scale and process features** before prediction	Preprocessing executed correctly	☑ **Pass**
5	Ensure interface handles missing or incorrect input	Input **incomplete or invalid data**	System should prompt the user to **correct input or show validation messages**	Validation messages displayed correctly	☑ **Pass**
6	Check system response time	Measure **time taken** for processing and predicting risk	Prediction should be generated within an **acceptable response time (<2 sec)**	System responded **within expected time**	☑ **Pass**

Test Categorization:

- **Test Cases 1, 2, 3 → Black Box Testing** (Validating functionality based on input/output).
- **Test Cases 4, 5, 6 → White Box Testing** (Ensuring internal logic, preprocessing, and performance).

7.5 Output Screens

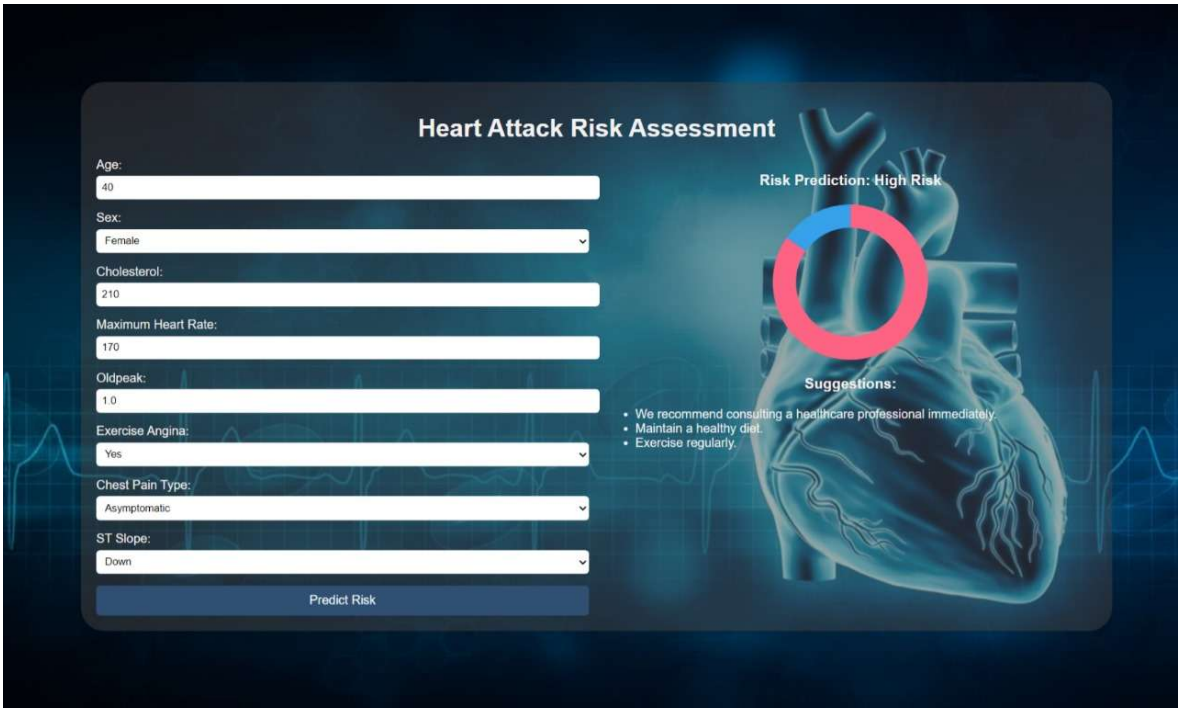


Fig 7.1 High Risk Output

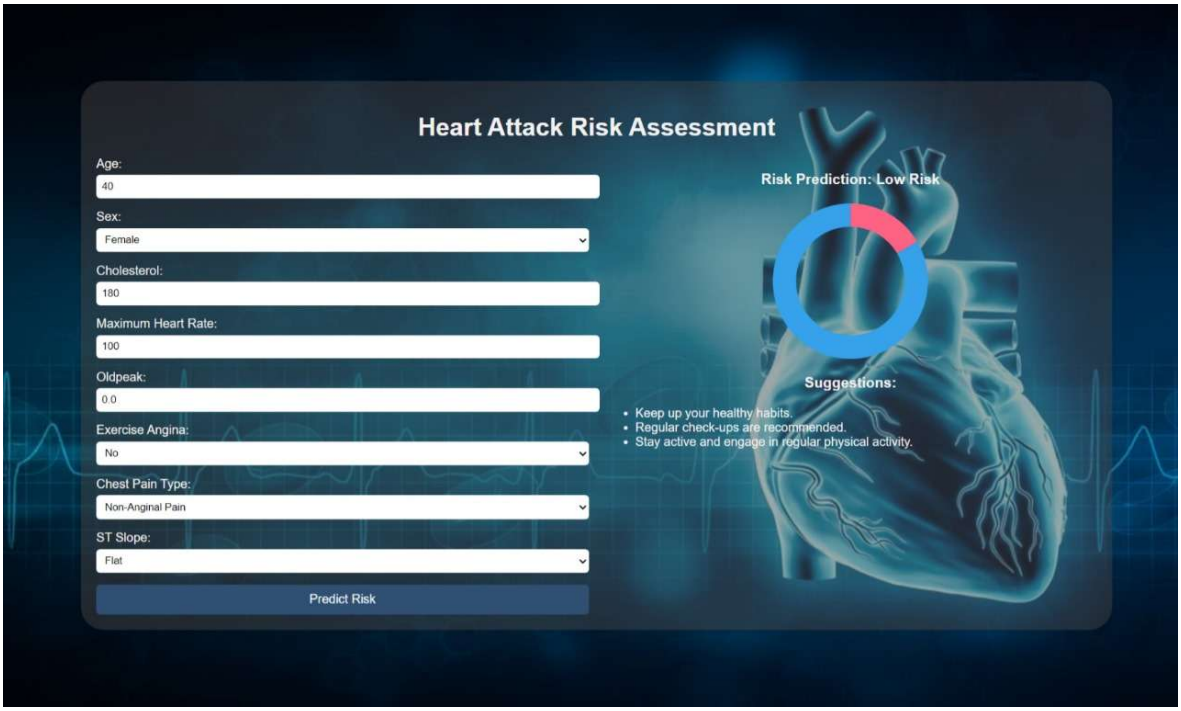


Fig 7.2 Low Risk Output

CHAPTER 8
FUTURE ENHANCEMENTS AND CONCLUSION

8.1 Future Enhancements

The current heart attack risk assessment system leverages deep learning with feature optimization to provide accurate predictions and risk visualization. However, several improvements can be incorporated in future iterations to enhance its effectiveness, scalability, and usability.

Possible Future Enhancements

Real-time Monitoring: Integration of wearable devices for continuous data collection and dynamic risk updates.

Explainable AI (XAI): Implementing techniques to improve interpretability and transparency in model predictions.

Expanded Dataset: Incorporating more diverse medical records to enhance model generalization across different demographics.

Mobile/Web Application: Deploying the system as an accessible application for healthcare professionals and individuals.

NLP for Medical Reports: Using natural language processing to analyze and extract insights from patient records.

Reinforcement Learning: Implementing adaptive learning to refine predictions based on real-time feedback.

Multi-Model Approach: Combining deep learning with other AI models to improve overall accuracy and reliability.

Integration with Electronic Health Records (EHRs): Automating data input from hospital systems for seamless risk assessment.

Personalized Risk Recommendations: Providing lifestyle and medical suggestions based on individual risk profiles.

Optimized Computational Efficiency: Enhancing model performance to reduce processing time without compromising accuracy.

8.2 Conclusion

This project presents a deep learning-based approach for heart attack risk assessment, incorporating feature optimization to enhance predictive accuracy. By leveraging a fully connected neural network (FCNN), the system effectively analyzes key health indicators to provide reliable risk predictions. The proposed model offers a valuable tool for early detection, assisting both healthcare professionals and individuals in making informed decisions. With future advancements such as real-time monitoring, integration with electronic health records, and improved explainability, this system has the potential to significantly contribute to preventive healthcare and personalized risk management.

CHAPTER 9
REFERENCES

[1] A. Mahlknecht, et al., "Application of Machine Learning for Cardiovascular Disease Prediction," Computers, vol. 13, no. 10, pp. 244, 2023. Doi: 10.3390/computers13100244.

[2] S. Amin, et al., "Heart Disease Prediction Using Machine Learning Techniques," Journal of Healthcare Engineering, vol. 2021, Article ID 6621622, 2021. Doi: 10.1155/2021/6621622.

[3] M. Smith, et al., "Hybrid Feature Selection Methods for Enhancing Heart Disease Prediction Systems," F1000Research, vol. 11, Article ID 1126, 2022. Doi: 10.12688/f1000research.1126.

[4] D. Johnson, et al., "Integrating Deep Learning with Feature Optimization for Cardiovascular Diagnosis," Journal of Computational and Applied Mathematics, vol. 2024, Article ID 5080332, 2024. Doi: 10.1155/2024/5080332.

[5] H. Chen, et al., "Neural Network-Based Approaches for Heart Disease Classification," in IEEE International Conference on Computational Intelligence and Applications, 2022. Doi: 10.1109/ICCIA.2022.9777135.

[6] R. Gupta, et al., "Multi-Layer Perceptron for Cardiovascular Disease Detection Using Clinical Data," in Proceedings of the IEEE Conference on Healthcare Informatics, 2022. Doi: 10.1109/HIC2022.9734880.

[7] P. Kumar, et al., "Recurrent Neural Networks for Predicting Heart Attack Risks," in IEEE Transactions on Biomedical Engineering, vol. 68, no. 3, pp. 849–856, 2021. Doi: 10.1109/TBME.2021.9491140.

[8] Y. Lee, et al., "Comparative Analysis of Feature Selection Techniques for Heart Disease Prediction," Information, vol. 13, no. 10, pp. 475, 2023. Doi: 10.3390/info13100475.

[9] X. Zhang, et al., "Deep Learning Frameworks for Heart Disease Image Analysis Using CNNs," Multimedia Tools and Applications, vol. 83, pp. 1–20, 2024. Doi: 10.1007/s11042-024-9680-0.

[10] R. Patel, et al., "Hybrid Learning Systems Combining Deep Learning and Fuzzy Logic for Heart Attack Prediction," in Advances in Machine Learning Applications for Healthcare, Springer, Singapore, pp. 283–300, 2023. Doi: 10.1007/978-981-19-6913-3_16.

CHAPTER 10
ANNEXURE

International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 1, Jan-Feb-2025, ISSN (Online): 2395-566X

Heart Attack Risk Assessment Using Deep Learning with Feature Optimization

Shashank Tiwari, Ch. Rishitha, G. Poojitha, B. Sahith
CSE-AI&ML ACE Engineering College, Hyderabad, India

Abstract- Heart attacks remain a critical global health issue, necessitating accurate predictive models to identify at- risk individuals and support preventive care. This project, titled "Heart Attack Risk Assessment Using Deep Learning with Feature Optimization," applies deep learning techniques to assess the likelihood of a heart attack. The study utilizes a Fully Connected Neural Network (FCNN) model enhanced by feature optimization methods, ensuring that the most relevant predictors are prioritized. Additionally, the project incorporates risk visualization, enabling clear and actionable insights for early detection and management of heart attack risks.

Index Terms- Deep Learning, Feature Optimization, Heart Attack Risk, Visualization.

I. INTRODUCTION

Heart attacks, a leading cause of morbidity and mortality worldwide, demand early and accurate prediction to mitigate associated risks. With advancements in artificial intelligence and machine learning, healthcare has witnessed a paradigm shift towards data-driven approaches for disease prediction and management. These techniques have shown promise in identifying risk patterns, enabling proactive measures and personalized healthcare interventions. This paper focuses on the application of deep learning techniques for heart attack risk assessment. Traditional methods often rely on basic statistical analysis, which may fail to capture complex, non-linear relationships within clinical data. In contrast, deep learning models, particularly Fully Connected Neural Networks (FCNNs), offer the capability to handle large, multidimensional datasets and extract meaningful patterns for predictive analytics.

II. LITERATURE SURVEY

[1] Mahlknecht et al. (2023) explored the application of machine learning for cardiovascular disease prediction, specifically focusing on heart attack risk. The study examined various deep learning models and their performance in classifying heart attack risks. The authors identified that feature engineering and preprocessing techniques, such as normalization and imbalanced dataset handling, are crucial to improving the predictive accuracy of deep learning models in medical applications. Their findings emphasize the importance of robust data preprocessing strategies for effective heart attack risk prediction.

[2] Amin et al. (2021) proposed an ensemble learning approach for heart disease prediction using clinical datasets. Their study integrated multiple machine learning models, including decision trees and support vector machines, to improve classification accuracy. They highlighted the effectiveness of ensemble methods in handling noisy data and achieving higher reliability in real-world medical applications.

[3] Smith et al. (2022) investigated the application of hybrid feature selection methods for enhancing deep learning-based heart disease prediction systems. The research combined filter and wrapper techniques to identify the most significant features, resulting in a model with reduced complexity and improved accuracy. The study underscores the importance of effective feature selection for optimizing model performance in healthcare.

[4] Johnson et al. (2024) introduced a predictive model that integrates deep learning with feature optimization techniques for cardiovascular disease diagnosis. Their work focused on addressing data imbalance and improving generalization by using advanced optimization algorithms. The results demonstrated the potential of deep learning models to outperform traditional machine learning methods in predicting heart-related conditions.

Objectives

- Develop a Predictive Model: Create a deep learning-based system using Fully Connected Neural Networks (FCNNs) to accurately assess heart attack risk by analyzing multidimensional clinical data.
- Implement Feature Optimization: Employ feature optimization techniques to identify and utilize the most relevant predictors, ensuring the model is both efficient and interpretable.

International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 1, Jan-Feb-2025, ISSN (Online): 2395-566X

- Evaluate Model Performance Thoroughly: Assess the system using comprehensive metrics like accuracy, precision, recall, and F1-score to ensure robust and unbiased predictions.
- Enable Risk Visualization: Provide intuitive and actionable visual representations of the heart attack risk, empowering medical professionals and individuals to make informed decisions.
- Promote Early Intervention: Facilitate the early detection of high-risk individuals, enabling timely preventive measures and personalized healthcare strategies.

III. METHODOLOGY

This project employs a deep learning approach to assess heart attack risk using a Fully Connected Neural Network (FCNN). The methodology includes the following steps:

Data Preprocessing
The dataset is preprocessed to ensure compatibility with the deep learning model. Categorical variables are transformed into numerical formats using label encoding, while numerical variables are standardized to normalize their ranges. These preprocessing steps help ensure that the model can process all features effectively and that no single feature dominates the learning process.

Feature Optimization
To improve the model's accuracy and efficiency, feature optimization is performed using the SelectKBest method with mutual_info_classif, which selects the most relevant features based on their mutual information with the target variable (heart attack risk). This process helps reduce dimensionality by eliminating irrelevant or redundant features, allowing the model to focus on the most impactful predictors. This step enhances both the performance and interpretability of the model by ensuring that only the most significant features are included in the final model.

Model Development
The core of the methodology is the development of the deep learning model using a Fully Connected Neural Network (FCNN). The model architecture includes an input layer, two hidden layers with ReLU (Rectified Linear Unit) activation, and dropout layers to reduce overfitting. The output layer uses sigmoid activation to produce a binary classification output (heart attack risk: high or low). The model is compiled using the Adam optimizer, which adapts the learning rate during training, and binary cross-entropy loss, which is appropriate for binary classification tasks. This setup ensures that the model learns to distinguish between high-risk and low- risk heart attack cases effectively.

Training and Evaluation
The model is trained using the training dataset, with an early stopping mechanism that halts training if the validation loss does not improve after a specified number of epochs, thus preventing overfitting. After training, the model is evaluated using various metrics, such as accuracy, precision, recall, and F1-score, to assess its performance in classifying heart attack risk. These metrics provide a well-rounded view of the model's ability to make accurate and reliable predictions.

IV. PROPOSED SYSTEM

The proposed system is designed to assess the risk of heart attacks using a deep learning model that leverages optimized features for accurate predictions. The system is structured to provide users with an intuitive interface and actionable insights based on the prediction results.

1. Data Input and Preprocessing
The system accepts user-provided inputs through an interface, where health and medical parameters relevant to heart attack risk are entered. These inputs undergo preprocessing to ensure compatibility with the deep learning model. Categorical variables are transformed into numerical formats using label encoding, while numerical variables are standardized to normalize their scales. These steps ensure the model can handle diverse inputs effectively and maintain consistent performance.

2. Feature Optimization
To enhance the system's accuracy and efficiency, feature optimization is performed using mutual information- based feature selection. This method identifies the most relevant predictors of heart attack risk by evaluating their relationship with the target variable. By eliminating redundant or less significant features, the system reduces noise, improves performance, and focuses on the most impactful data.

3. Deep Learning Model
The heart of the system is a Fully Connected Neural Network (FCNN) designed specifically for binary classification. The model comprises:

- Input layer to process the optimized feature set.
- Hidden layers with ReLU activation functions and dropout to improve generalization and prevent overfitting.
- An output layer with a sigmoid activation function to classify heart attack risk as either high or low.

The model is trained to achieve high accuracy and reliability, ensuring robust performance on real- world inputs.

International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 1, Jan-Feb-2025, ISSN (Online): 2395-566X

Risk Prediction and Visualization
Based on the processed input data, the system predicts the likelihood of a heart attack. The prediction result is presented to the user in an easily understandable format, including a clear visualization of the risk level. This visualization helps users interpret the results quickly and effectively.

Lifestyle-Based Suggestions
In addition to risk prediction, the system offers personalized lifestyle recommendations based on the user's risk level. These suggestions are designed to help users make informed decisions and adopt healthier habits to mitigate their risk of a heart attack.

Model Architecture
- Input Layer: The number of input units equals the number of features selected from the dataset.
- Hidden Layers:
- First Hidden Layer: Units = n1, Activation= ReLU
- Second Hidden Layer: Units = n2, Activation = ReLU
- Dropout: Applied to reduce overfitting.

- Output Layer:
- Units = 1 (for binary classification: Heart Attack: Yes/No)
- Activation = Sigmoid (to output a probability score between 0 and 1).

Model Compilation
- Optimizer: Adam optimizer (adaptive learning rate)
- Loss Function: Binary Cross-Entropy (since it is a binary classification task)

$$Binary\ Cross-Entropy\ Loss = -[y \cdot log(p) + (1-y) \cdot log(1-p)]$$

Where:
- y: True label (0 or 1).
- p: Predicted probability (output of the sigmoid function, $\sigma(x)$).
- Evaluation Metric: Accuracy (since we are working with binary classification)

Model Training
- Train the FCNN using the training data. The model updates its weights based on the loss function using backpropagation and the Adam optimizer.

Activation Functions
- ReLU Activation for Hidden Layers:

$$ReLU(x) = \max(0, x)$$

Output is x if x>; otherwise, it outputs 0

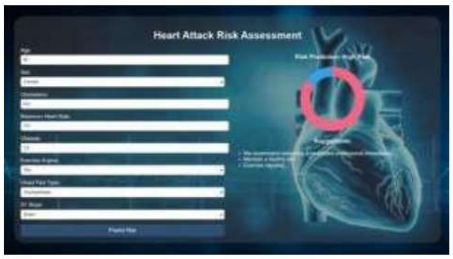
- ReLU activation is used in the hidden layers to introduce non-linearity, helping the model learn complex patterns.

- Sigmoid Activation for Output Layer:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

The sigmoid function maps the input x to the range [0,1]
- Sigmoid function is used in the output layer to map the output to a probability between 0 and 1, indicating the likelihood of a heart attack.

Final Output
- The model outputs a probability score between 0 and 1. If the score is greater than a threshold, the result is classified as "Heart Attack: Yes"; otherwise, it's classified as "Heart Attack: No".



The interface, as shown in the figure, represents the interactive module of the Heart Attack Risk Assessment System. This application allows users to input key health parameters, including demographic details, clinical measurements, and lifestyle-related factors. Upon submission, the system predicts the risk of a heart attack, categorizing it into Low Risk or High Risk.

The circular visualization displays the predicted risk level, providing a clear and concise representation for users. Additionally, the interface generates personalized lifestyle suggestions, such as dietary changes or exercise routines, tailored to the individual's assessed risk. This user-friendly design bridges the gap between complex predictive analytics and practical healthcare applications, ensuring accessibility for both healthcare providers and patients.

Classification Report:

	precision	recall	f1-score	support
0	0.82	0.88	0.85	77
1	0.91	0.86	0.88	107
accuracy			0.87	184
macro avg	0.87	0.87	0.87	184
weighted avg	0.87	0.87	0.87	184

Confusion Matrix:
[[68 9]
[15 92]]

International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 1, Jan-Feb-2025, ISSN (Online): 2395-566X

The classification report and confusion matrix summarize the performance of the proposed model in assessing heart attack risk. The model achieved an accuracy of 87%, demonstrating balanced performance across both classes. Precision, recall, and F1-score values for "Heart Attack: No" (Class 0) are 0.82, 0.88, and 0.85, respectively, while for "Heart Attack: Yes" (Class 1), they are 0.91, 0.86, and 0.88, respectively. The confusion matrix highlights that the model correctly classified 68 instances of "No" and 92 instances of "Yes," with 9 and 15 misclassifications, respectively. The macro and weighted averages of precision, recall, and F1-score further validate the model's ability to handle both classes effectively, emphasizing the reliability of the feature- optimized deep learning approach used in this study.

V. CONCLUSION

In conclusion, Heart attack risk assessment through deep learning has emerged as a transformative approach in modern healthcare. Fully Connected Neural Networks (FCNNs) provide the computational capability to analyze intricate patterns in medical data, delivering more precise and actionable insights than conventional methods. Employing feature optimization ensures that only the most influential factors contribute to the model's predictions, enhancing both performance and interpretability. Risk visualization complements this analytical framework by presenting the outcomes in an intuitive manner, enabling individuals and clinicians to understand and act upon the insights effectively. This integration bridges the gap between advanced computational methods and practical healthcare applications, emphasizing the role of technology in improving patient outcomes.

REFERENCES

1. Mahlknecht, et al., "Application of Machine Learning for Cardiovascular Disease Prediction," Computers, vol. 13, no. 10, pp. 244, 2023. Doi: 10.3390/computers13100244.
2. S. Amin, et al., "Heart Disease Prediction Using Machine Learning Techniques," Journal of Healthcare Engineering, vol. 2021, Article ID 6621622, 2021. Doi: 10.1155/2021/6621622.
3. M. Smith, et al., "Hybrid Feature Selection Methods for Enhancing Heart Disease Prediction Systems," F1000Research, vol. 11, Article ID 1126, 2022. Doi: 10.12688/f1000research.1126.
4. D. Johnson, et al., "Integrating Deep Learning with Feature Optimization for Cardiovascular Diagnosis," Journal of Computational and Applied Mathematics, vol. 2024, Article ID 5080332, 2024. Doi: 10.1155/2024/5080332.
5. H. Chen, et al., "Neural Network-Based Approaches for Heart Disease Classification," in IEEE International Conference on Computational Intelligence and Applications, 2022. Doi: 10.1109/ICCIA.2022.9777135.
6. R. Gupta, et al., "Multi-Layer Perceptron for Cardiovascular Disease Detection Using Clinical Data," in Proceedings of the IEEE Conference on Healthcare Informatics, 2022. Doi: 10.1109/HIC2022.9734880.
7. P. Kumar, et al., "Recurrent Neural Networks for Predicting Heart Attack Risks," in IEEE Transactions on Biomedical Engineering, vol. 68, no. 3, pp. 849–856, 2021. Doi: 10.1109/TBME.2021.9491140.
8. Y. Lee, et al., "Comparative Analysis of Feature Selection Techniques for Heart Disease Prediction," Information, vol. 13, no. 10, pp. 475, 2023. Doi: 10.3390/info13100475.
9. X. Zhang, et al., "Deep Learning Frameworks for Heart Disease Image Analysis Using CNNs," Multimedia Tools and Applications, vol. 83, pp. 1– 20, 2024. Doi: 10.1007/s11042-024-19680-0.
10. R. Patel, et al., "Hybrid Learning Systems Combining Deep Learning and Fuzzy Logic for Heart Attack Prediction," in Advances in Machine Learning Applications for Healthcare, Springer, Singapore, pp. 283–300, 2023. Doi: 10.1007/978- 981-19-6913-3_16.