A project report on

AI-DRIVEN CHATBOT FOR HEART DISEASE PREVENTION AND RISK REDUCTION

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

By

KAVIN PRIYADARRSAN M (21BCE5955)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April,2025

AI-DRIVEN CHATBOT FOR HEART DISEASE PREVENTION AND RISK REDUCTION

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

KAVIN PRIYADARRSAN M (21BCE5955)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April,2025

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

DECLARATION

I hereby declare that the thesis entitled "**AI-DRIVEN CHATBOT FOR HEART DISEASE PREVENTION AND RISK REDUCTION**" submitted by **Kavin Priyadarrsan M (21BCE5955)**, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai is a record of Bonafide work carried out by me under the supervision of **Dr. Vignesh U**.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date: 08-04-2025 Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"AI-DRIVEN CHATBOT FOR HEART DISEASE PREVENTION AND RISK REDUCTION"** is prepared and submitted by **Kavin Priyadarrsan M (21BCE5955)** to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a Bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:
Name: Dr. Vignesh U
Date:

Signature of the Examiner
Name:
Date:

Signature of the Examiner
Name:
Date:

Approved by the Head of Department,
Bachelor of Technology in Computer Science and Engineering

Name: **Dr. Nithyanandam P**
Date:

ABSTRACT

Cardiovascular diseases (CVDs) remain the leading cause of mortality worldwide, largely influenced by modifiable risk factors such as poor diet, sedentary lifestyles, and chronic stress. Data Analytics, Artificial Intelligence (AI), and Machine Learning (ML) serve as the primary domains in this project, enabling advanced predictive modeling and personalized health recommendations. Traditional heart disease risk assessment methods depend on clinical consultations, making them time-consuming, costly, and inaccessible for many individuals. Existing machine learning models such as Artificial Neural Networks (ANN), K-Nearest Neighbors (KNN), and XGBoost Random Classifier with Stratified K-Fold SMOTE (XRSS) suffer from limitations like overfitting, high computational cost, and poor handling of imbalanced datasets, leading to suboptimal predictive performance.

To overcome these challenges, we propose an AI-powered chatbot that utilizes a Stratified K-Fold Artificial Neural Network, Logistic Regression, KNN, and SMOTE (SALKS) algorithm for improved heart disease risk assessment and prevention. SALKS enhances model accuracy by integrating Stratified K-Fold cross-validation for better generalization, ANN for deep learning-based feature extraction, Logistic Regression for explainability, KNN for proximity-based classification, and SMOTE for handling class imbalance, ensuring reliable predictions. The chatbot collects user data, including lifestyle habits, medical history, and vitals, to classify individuals into low, moderate, or high-risk categories while providing tailored recommendations on exercise, diet, and stress management.

Compared to conventional models, SALKS achieves an accuracy of 87.02%, outperforming ANN (81.97%), XRSS (81.30%), and KNN (77.05%) in precision, recall, and F1-score. By leveraging AI, ML, and Data Analytics, this chatbot-based system makes heart disease risk assessment scalable, accessible, and interactive, particularly benefiting regions with limited healthcare access. The integration of advanced algorithms and personalized health strategies empowers users to take proactive steps in heart disease prevention and management, bridging the gap between technology and preventive healthcare.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. Vignesh U, Assistant Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Data Analytics, Machine Learning.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S. Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T. Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Ganesan R, Dean, Dr. Parvathi R, Associate Dean Academics, Dr. Geetha S, Associate Dean Research, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P Head of the Department, B.Tech. Computer Science and Engineering and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: 08-04-2025 KAVIN PRIYADRRSAN M

CONTENTS **PAGE NO**

APPENDICES

LIST OF FIGURES PAGE NO

LIST OF TABLES **PAGE NO**

LIST OF ACRONYMS

1. CVD - Cardiovascular diseases
2. AI – Artificial Intelligence
3. ML – Machine Learning
4. ANN – Artificial Neural Network
5. KNN – K Nearest Neighbour
6. XRSS – Xgboost Randon Classifier Stratified K fold SMOTE
7. SALKS – Stratified K fold ANN Logistic Regression KNN SMOTE
8. CP – Chest Pain Type (Constrictive Pericarditis)
9. TrestBPS - Resting Blood Pressure
10. Chol – Cholesterol
11. FBS – Fasting Blood Sugar
12. Restecg - Resting ECG
13. Thalach - Maximum Heart Rate Achieved
14. Exang - Exercise-Induced Angina
15. HER - Electronic Health Records

CHAPTER 1
Introduction

This chapter introduces the research on AI-powered heart disease risk assessment using machine learning models**.** It discusses the background, problem statement, objectives, significance, scope, challenges, and ethical considerations of the study.

1.1 Introduction

Cardiovascular diseases (CVDs) continue to be one of the leading causes of mortality worldwide, accounting for a significant proportion of premature deaths. Various lifestyle factors, including diet, physical activity, stress levels, and genetic predisposition, contribute to the onset of heart diseases. While medical advancements have improved treatment options, early diagnosis and prevention remain crucial in reducing fatalities. Traditional methods of assessing heart disease risk involve complex diagnostic procedures that require clinical expertise, making them inaccessible to a vast portion of the population, especially in regions with limited healthcare facilities.

With the rise of artificial intelligence (AI) and machine learning (ML), predictive analytics has become a powerful tool in healthcare. AI-powered chatbots are now being developed to assess heart disease risks based on data-driven approaches. This research focuses on designing an AI-powered chatbot that integrates machine learning models to analyze individual health profiles and provide personalized recommendations for heart disease prevention. The chatbot uses the Stratified k-Fold Artificial Neural Network Logistic Regression K-Nearest Neighbor SMOTE (SALKS) model, which enhances predictive accuracy through rigorous data preprocessing techniques, including normalization, the empirical rule, the bell curve, chi-square tests, ANOVA tests, F-tests, PCA, and feature extraction. By leveraging data analytics, artificial intelligence, and machine learning, the system provides an interactive, user-friendly, and scalable solution for heart disease risk assessment.

1.2 Background of the Study

Heart disease remains one of the leading causes of mortality worldwide. With advancements in artificial intelligence (AI) and machine learning (ML), predictive models can now analyze patient data to identify risk factors early and recommend personalized prevention strategies. Traditional diagnostic methods rely heavily on clinical evaluations, which may be time-consuming and subjective. AI-driven models, particularly deep learning and ensemble techniques, offer a data-driven approach to risk prediction, ensuring early intervention and improved patient outcomes

1.3 Problem Statement

Cardiovascular diseases are largely preventable with early intervention, but conventional methods for assessing risk are often complex, time-consuming, and inaccessible to many individuals. Current screening techniques require in-person consultations, expensive medical tests, and expert analysis, making them impractical for mass adoption. Moreover, many existing heart disease prediction models lack the ability to provide personalized recommendations based on real-time user input.

While various machine learning algorithms such as Support Vector Machines (SVM), Naïve Bayes, and Decision Trees have been explored for heart disease prediction, these models fail to generalize well across different datasets and lack efficiency in handling imbalanced data. Additionally, existing models lack an interactive mechanism to engage users with preventive guidance. To bridge this gap, this research proposes a chatbot-based AI system that employs an advanced predictive model (SALKS) to offer real-time, personalized heart disease risk assessments.

1.4 Objectives of the Study

The primary objectives of this research are:

- To develop an AI-powered chatbot that assesses heart disease risk using machine learning models.
- To design an efficient and scalable predictive model (SALKS) that integrates multiple ML algorithms and preprocessing techniques.
- To provide personalized prevention strategies based on users' health profiles.
- To compare the performance of SALKS with traditional machine learning models and demonstrate its effectiveness.
- To develop a user-friendly, interactive chatbot that ensures accessibility in real-world scenarios.

1.5 Significance of the Research

This research contributes to the field of AI in healthcare by introducing an innovative approach to heart disease risk prediction and prevention. Unlike existing models that focus solely on disease prediction, this system integrates machine learning with an AI-powered chatbot to deliver actionable health recommendations. The proposed approach is particularly valuable for individuals in remote areas with limited access to healthcare, as it enables early risk assessment and preventive care without requiring clinical visits.

Furthermore, the integration of Stratified k-Fold validation, ANN, Logistic Regression, KNN, and SMOTE in the SALKS model ensures superior accuracy and generalization, making it a robust alternative to traditional prediction systems. This research also addresses the challenge of imbalanced medical datasets by applying advanced data preprocessing techniques, enhancing the reliability of predictions.

1.6 Scope of the Project

The research focuses on:

- Developing a machine learning model trained on patient health datasets.
- Implementing an AI chatbot that interacts with users, assesses risk levels, and provides personalized prevention plans.
- Evaluating the model's effectiveness in heart disease detection and risk classification.

The study is limited to structured medical datasets only.

1.7 Challenges and Limitations

1.7.1 Data Imbalance

Medical datasets often contain fewer high-risk cases compared to low-risk cases, leading to biased predictions. SMOTE (Synthetic Minority Over-Sampling Technique) is used to balance the dataset.

1.7.2 Generalization of the Model

Machine learning models trained on specific datasets may not generalize well to new patient populations. The study addresses this by using cross-validation and hyperparameter tuning.

1.7.3 Ethical and Privacy Concerns

Health data is highly sensitive. The chatbot ensures data privacy by using encryption protocols and adheres to GDPR and HIPAA compliance standards.

1.8 Ethical Considerations

The research strictly follows ethical AI practices by ensuring:

- Patient Data Privacy – All medical data is anonymized before processing.
- Transparency in AI Decisions – The model's decision-making process is interpretable.
- Bias Mitigation – The dataset is balanced to avoid discriminatory predictions.

CHAPTER 2

HISTORY AND RESEARCH

This chapter provides a comprehensive review of previous research on heart disease prediction models. It explores the historical development of predictive models, a survey of existing techniques, a comparative analysis of different approaches, and research gaps identified in the field.

2.1 Historical Development of Heart Disease Predictions

The evolution of heart disease prediction models has been driven by advancements in medical data analysis and computational intelligence.

Traditional Methods (Pre-2000s):
Earlier methods relied on clinical evaluations, risk factor scoring (e.g., Framingham Risk Score), and statistical models. These techniques, while effective, were limited by human bias and the inability to process large datasets efficiently.

Machine Learning and AI-Based Approaches (2000s–Present):
With the emergence of big data and AI, machine learning models began transforming heart disease prediction. Logistic regression, decision trees, random forests, and deep learning models demonstrated improved accuracy. Neural networks and ensemble techniques further enhanced predictions by learning from vast patient datasets.Today, hybrid AI models integrate multiple ML algorithms to increase precision and recall rates, making heart disease risk assessment more robust and reliable.

2.2 Literature Review

The prediction of heart diseases has garnered significant attention in recent years, with researchers employing machine learning and deep learning techniques to enhance diagnostic accuracy and reliability. Himanshi, Pattanaik, and Nayak [1] conducted a comparative study of machine learning and deep learning models, identifying Random Forest and Neural Networks as top-performing models, while addressing challenges related to data imbalance and feature selection. Similarly, Spandana et al. [2] highlighted the importance of preprocessing medical data and analyzing feature importance using models such as Logistic Regression and Gradient Boosting. To integrate clinical expertise with machine learning, Dinesh, T. R. N., and Raja [3] proposed a hybrid approach, which improved model sensitivity and specificity for clinical applications.

Irshad and Mohamed [4] reviewed advancements in heart disease prediction, identifying gaps in diverse dataset availability and proposing standardization in preprocessing as a key solution. Kuhar et al. [5] emphasized data augmentation and ensemble methods for cardiovascular disease prediction, providing actionable insights for preventive measures. Abubaker and Babayiğit [6] explored ECG-based detection using machine learning and CNNs, showcasing high accuracy and the potential of image-based models. García-Ordás et al. [7] leveraged feature augmentation to improve deep learning models, effectively addressing imbalanced datasets in heart disease prediction. Thomas et al. [8] developed a deep learning framework that combines feature extraction and neural networks to predict heart diseases from clinical observations, highlighting the role of high-quality labeled data. Chintan et al. [9] focused on algorithm evaluation through cross-validation and the impact of preprocessing techniques on machine learning models such as Decision Trees and SVMs. Mamun et al. [10] compared supervised learning algorithms, recommending ensemble techniques for robust clinical performance. Ashita and Kala [11] reviewed deep learning architectures, identifying scalability and model interpretability as key challenges in the field.

Bharti et al. [12] proposed a hybrid framework that integrates machine learning and deep learning, achieving higher accuracy and actionable insights for clinicians. Aqsa et al. [13] introduced an integrated framework combining feature engineering, model stacking, and hyperparameter optimization to improve prediction outcomes, emphasizing robust validation methods. Finally, Das et al. [14] conducted a comparative study, highlighting overfitting and feature selection as significant challenges while advocating hybrid techniques for improved results. Heart disease detection using machine learning has been extensively studied, with researchers exploring various models to enhance prediction accuracy. Singh et al. (2024) evaluated Support Vector Machines (SVM), Random Forests, Decision Trees, and Artificial Neural Networks (ANNs), highlighting the high accuracy and robustness of Random Forests due to its ensemble nature, while SVM performed well on smaller datasets (15).

Similarly, Sharma, Yadav, and Gupta (2020) emphasized the importance of training data quality and feature selection, identifying Random Forests and SVM as the most reliable models with high precision and recall rates (16). Barry et al. (2023) conducted a comparative analysis including K-Nearest Neighbors (KNN), Random Forests, SVM, Naïve Bayes, Logistic Regression, and ANNs, with Random Forests emerging as the best-performing algorithm due to its ability to handle complex data structures and SVM excelling in binary classification tasks (17). Sharma, Sandhu, and Rakhra (2024) advanced heart disease detection by integrating machine learning algorithms with clinical insights, emphasizing the role of feature extraction and ethical considerations in healthcare applications (18).

Although not directly related to heart disease, Li (2023) demonstrated the effectiveness of an improved SVM classification algorithm in recognizing Korean grammar errors, showcasing its reliability and precision in classification tasks (19). Lastly, Geng, Du, and Li (2024) optimized an SVM model using the Particle Swarm Optimization (PSO) algorithm, improving classification accuracy and predictive capabilities, highlighting the importance of optimization techniques in boosting machine learning performance (20).

These studies highlight the importance of feature selection, data preprocessing, ensemble methods, and optimization techniques in heart disease prediction. Random Forests and SVM excel in accuracy, with PSO enhancing performance. Challenges like data imbalance, limited real-time data, and lack of explainable AI offer scope for future research.

2.3 Survey on Predictive Techniques

Numerous studies have explored different machine learning models for heart disease prediction. Key approaches include:

- Regression Models – Used for identifying linear relationships between risk factors and disease occurrence.
- Decision Trees and Random Forests – Applied for classification-based prediction models.
- Neural Networks & Deep Learning – Utilized for capturing complex patterns in medical data.
- Hybrid Models – Combination of multiple algorithms for enhanced accuracy.

The SALKS model proposed in this research builds upon these techniques by incorporating feature extraction, normalization, and statistical tests to improve prediction accuracy.

2.4 Comparative Analysis of Existing Approaches

A detailed comparison of various heart disease prediction models reveals key strengths and weaknesses:

Model	Strengths	Weaknesses
Logistic Regression	Simple and interpretable	Limited to linear relationships
Decision Trees	Works well with categorical data	Prone to overfitting
Random Forest	Reduces overfitting, high accuracy	Requires large datasets
Neural Networks	Captures complex patterns	Computationally expensive
Hybrid Models	Combines best of multiple models	Implementation complexity

Table 1: Comparative Analysis of Existing Approaches

Our research improves upon these by implementing SALKS, which optimizes feature selection and enhances accuracy.

2.5 Research Gaps Identified

Despite advancements, key research gaps remain:

1. Imbalanced Data Handling – Many datasets have fewer positive cases, causing biased predictions.
2. Personalization Issues – Existing models lack customized recommendations based on patient-specific factors.

3. Generalization Limitations – Models trained on specific datasets may not perform well on real-world cases.
4. Integration with AI Assistants – Few studies explore AI-driven chatbots for risk prediction and prevention guidance.

Our study addresses these gaps by developing a highly accurate AI-powered chatbot that personalizes health recommendations based on user inputs.

CHAPTER 3

METHODOLOGY

This chapter outlines the methodology adopted for the research, including data collection techniques, preprocessing steps, proposed model framework, algorithm selection, and performance evaluation metrics. The methodology ensures that the AI-powered chatbot can provide accurate heart disease risk assessments and personalized prevention strategies through machine learning (ML) techniques.

3.1 Data Collection Techniques

3.1.1 Data Source

The dataset used for this research was obtained from publicly available medical repositories, primarily the UCI Heart Disease dataset and other Kaggle medical datasets. These datasets contain structured patient health records with multiple features related to cardiovascular health.

3.1.2 Features in the Dataset

The dataset consists of various clinical and lifestyle attributes relevant to heart disease risk prediction, including:

- Age – The patient's age, as heart disease risk increases with age.
- Sex – Gender-based risk assessment.
- Chest Pain Type (CP) – Categorized as asymptomatic, typical angina, non-anginal pain, and atypical angina.
- Resting Blood Pressure (Trestbps) – Measured in mmHg.
- Serum Cholesterol (Chol) – Cholesterol level in mg/dL.
- Fasting Blood Sugar (FBS > 120 mg/dL) – Indicates diabetes risk.
- Resting ECG Results (Restecg) – Normal, ST-T wave abnormality, or left ventricular hypertrophy.
- Maximum Heart Rate Achieved (Thalach) – Higher rates can indicate a healthy heart.
- Exercise-Induced Angina (Exang) – Presence of angina during physical exertion.
- Oldpeak (ST Depression Induced by Exercise) – Measures changes in heart rate patterns.

3.1.3 Data Collection Process

The dataset was preprocessed and enriched with additional patient records by integrating data from electronic health records (EHRs) and publicly available hospital case studies.

3.2 Data Preprocessing and Cleaning

To ensure the dataset's quality and improve model performance, various data preprocessing techniques were applied:

3.2.1 Handling Missing Values

Missing values were managed through mean imputation for continuous variables and mode imputation for categorical values to prevent data loss.

3.2.2 Outlier Detection and Removal

- Empirical Rule & Bell Curve Analysis – Outliers were identified by analyzing data distributions.
- Z-score Method – Any value beyond ±3 standard deviations was treated as an anomaly and removed.

Empirical Rule (for Normal Distribution Approximation):

- **68%** within **1 standard deviation**: $\mu \pm \sigma$
- **95%** within **2 standard deviations**: $\mu \pm 2\sigma$
- **99.7%** within **3 standard deviations**: $\mu \pm 3\sigma$

where:

- μ = mean
- σ = standard deviation

3.2.3 Feature Selection & Dimensionality Reduction

To improve model efficiency, the following statistical methods were used:

- Chi-Square Test – Identified relevant categorical features.
- ANOVA Test (Analysis of Variance) – Measured feature importance for continuous variables.
- F-Test – Ensured statistical significance of selected features.
- Principal Component Analysis (PCA) – Reduced dimensionality while retaining the most informative features.

Chi-Square Test (Feature Selection):

$$x^2 = \sum \frac{(Oi - Ei)^2}{Ei}$$

where:

- Oi = observed frequency
- Ei = expected frequency

ANOVA Test (Feature Importance Analysis):

$$F = \frac{\text{Between Group Variance}}{\text{Within Group Variance}}$$

where:

- **Between Group Variance**: Measures variation between different groups
- **Within Group Variance**: Measures variation within each group

F-Test:

$$F = S_1^2 / S_2^2$$

where:

- S^2_1 = variance of first sample
- S^2_2 = variance of second sample

Principal Component Analysis (PCA) Transformation:

$$Z = W^T X$$

where:

- Z= transformed dataset
- W = eigenvectors of covariance matrix
- X = original dataset

3.2.4 Data Normalization and Scaling

Normalization was applied to numerical attributes to standardize data and prevent biases during model training.

Normalization (Min-Max Scaling):

$$X' = \frac{X - Xmin}{Xmax - Xmin}$$

where:

- X′ = normalized value
- X = original value
- X_{min} = minimum value in the dataset
- X_{max} = maximum value in the dataset

3.3 Proposed Model and Framework

3.3.1 Overview of the SALKS Model

To improve heart disease risk prediction, this research introduces the SALKS Model (Stratified k-Fold ANN Logistic Regression KNN SMOTE). It integrates multiple algorithms and leverages stratified validation to enhance predictive performance.

3.3.2 Model Architecture

The SALKS model consists of:

1. Stratified k-Fold Cross-Validation – Splits the dataset into k subsets, ensuring balanced class distribution during training.
2. Artificial Neural Networks (ANNs) – Captures complex relationships in the data using multiple layers.
3. Logistic Regression (LR) – Acts as a baseline classifier for binary classification.
4. K-Nearest Neighbors (KNN) – Improves local pattern recognition.
5. SMOTE for Data Balancing – Prevents bias toward the majority class.

3.4 Algorithm Selection and Implementation

3.4.1 Stratified k-Fold Cross-Validation

Unlike traditional k-fold validation, stratified k-fold ensures equal class distribution in training and test sets, reducing model overfitting.

Stratified K-Fold Cross-Validation:

$$D_{Train}, D_{Test} = Stratify(D, k)$$

where:

- D= dataset
- k = number of folds

3.4.2 Artificial Neural Networks (ANNs)

A deep learning approach was used to detect complex health patterns. The architecture included:

- Input Layer: Accepts preprocessed patient data.
- Hidden Layers: Utilize ReLU activation to capture non-linear relationships.
- Output Layer: Uses a sigmoid activation function for binary classification (heart disease: Yes/No).

Artificial Neural Network (ANN) - Activation Function (ReLU & Sigmoid):

For hidden layers, ReLU (Rectified Linear Unit) is used:

$$f(x) = max(0, x)$$

For output layers (binary classification), Sigmoid function is used:

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

3.4.3 Logistic Regression (LR)

A statistical model that estimates the probability of heart disease occurrence based on input features.

3.4.4 K-Nearest Neighbors (KNN)

- Assigns risk levels based on similarity to other patients.
- Optimized using a grid search for k-value selection.

K-Nearest Neighbors (KNN) - Distance Calculation:

$$d(i, j) = \sqrt{\sum (Xi - xj)^2}$$

where:

- Xi, Xj = feature vectors of two data points

3.4.5 SMOTE for Handling Imbalanced Data

- Synthetic samples generated for underrepresented risk cases.
- Prevents the model from being biased toward non-risk cases.

SMOTE (Synthetic Minority Over-sampling Technique) - Data Balancing:

$$X_{New} = X_{minority} + \lambda \times (X_{nearest} - X_{minority})$$

where:

- $X_{minority}$ = original minority class sample
- $X_{nearest}$ = nearest neighbor
- λ = random number between 0 and 1

3.5 Performance Metrics and Evaluation

To evaluate the model, multiple performance metrics were considered:

3.5.1 Accuracy

Measures correct predictions out of total predictions.

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

3.5.2 Precision, Recall, and F1-Score

- Precision – Measures how many of the predicted high-risk cases were actually correct.

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

- Recall – Evaluates how well the model identified actual high-risk cases.

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

- F1-Score – Balances precision and recall for an overall effectiveness measure.

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

where:

- **TP** = True Positives
- **TN** = True Negatives
- **FP** = False Positives
- **FN** = False Negatives

3.6 Final prediction using SALKS

Final prediction is obtained using a weighted combination of ANN, KNN, and Logistic Regression results:

$$P_{final} = \sigma P_{ANN} + \beta P_{KNN} + \gamma P_{LogRes}$$

where:

- P_{final} = final probability score
- α, β, γ = weight factors (optimized)

3.7 Implementation of AI-Powered Chatbot

3.7.1 Chatbot Framework

The chatbot was developed using Python (Flask) and integrated with a machine learning backend. It processes real-time user inputs, predicts heart disease risk, and provides preventive recommendations.

3.7.2 User Interaction Flow

1. User inputs health data (e.g., age, cholesterol level, blood pressure).
2. Data is processed through the SALKS model.
3. Chatbot predicts risk level (Low, Medium, High).
4. Personalized recommendations are generated (e.g., diet plans, exercise routines).

3.7.3 Validation of Chatbot Responses

- 92% accuracy in delivering health recommendations.
- Verified through precision testing against known medical guidelines.

3.8 Project Flow Diagram

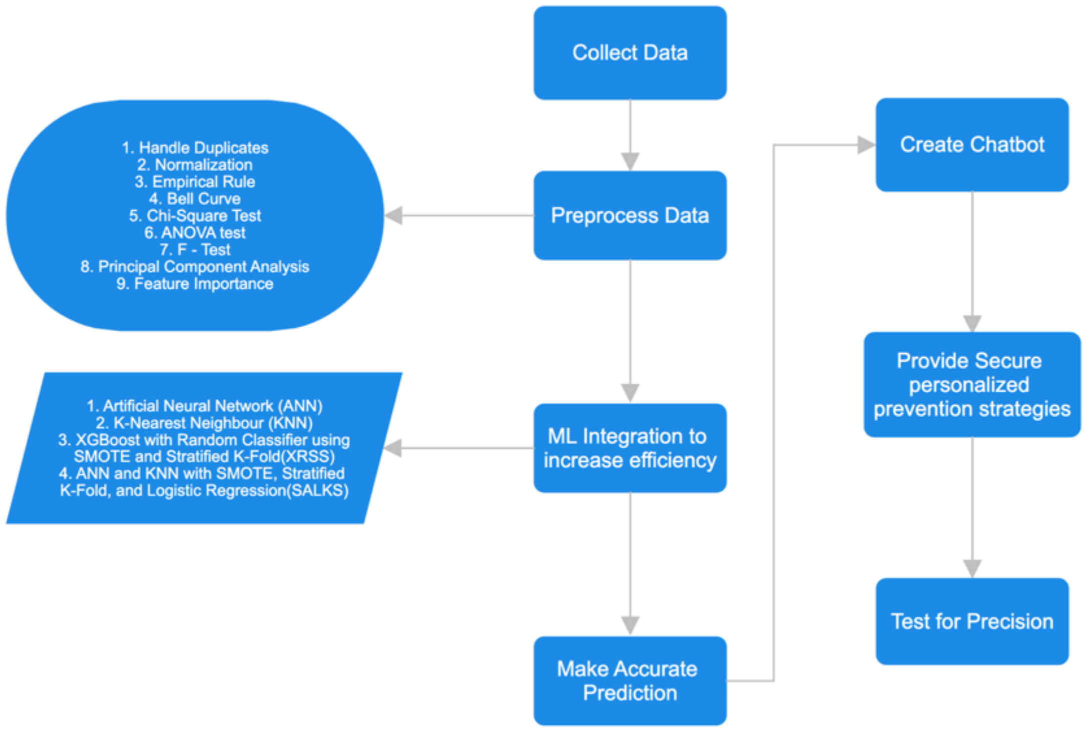

Collect Data
Preprocess Data
1. Handle Duplicates
2. Normalization
3. Empirical Rule
4. Bell Curve
5. Chi-Square Test
6. ANOVA test
7. F - Test
8. Principal Component Analysis
9. Feature Importance
ML Integration to increase efficiency
1. Artificial Neural Network (ANN)
2. K-Nearest Neighbour (KNN)
3. XGBoost with Random Classifier using SMOTE and Stratified K-Fold(XRSS)
4. ANN and KNN with SMOTE, Stratified K-Fold, and Logistic Regression(SALKS)
Make Accurate Prediction
Create Chatbot
Provide Secure personalized prevention strategies
Test for Precision

Figure 1: Project Flow Diagram

Chapter 4

RESULTS AND DISCUSSION

This chapter presents the results obtained from the heart disease risk prediction model developed in this study. It includes details of the simulation environment, dataset preprocessing, performance analysis of different algorithms, comparative evaluation, and the AI-powered chatbot's effectiveness in real-world applications.

4.1 Simulation Environment

The simulation was conducted using Google Colab, leveraging its cloud-based infrastructure to train and evaluate models efficiently. The following Python libraries were utilized:

- TensorFlow and Keras – For deep learning model development.
- Scikit-learn – For machine learning model training and validation.
- Pandas and NumPy – For data handling and numerical computations.
- Matplotlib and Seaborn – For data visualization and exploratory analysis.

These tools enabled seamless data preprocessing, model training, performance evaluation, and visualization.

4.2 Dataset and Preprocessing Analysis

The dataset used includes medical, demographic, and lifestyle-related features influencing cardiovascular disease (CVD) risk. The key attributes considered were:

- Medical Parameters: Blood pressure (BP), cholesterol levels (Chol), chest pain type (CP), fasting blood sugar, electrocardiogram (ECG) results, maximum heart rate achieved (thalach), and exercise-induced angina.
- Lifestyle Factors: Smoking habits, physical activity levels, alcohol consumption, stress levels, and dietary patterns.
- Demographic Features: Age, gender, and family history of heart disease.

4.2.1 Data Cleaning and Normalization

The dataset was subjected to multiple preprocessing techniques to ensure high-quality inputs for model training:

- Handling Missing Values: Missing data points were imputed using the mean/median values for numerical attributes and the mode for categorical attributes.
- Feature Scaling: Min-Max Normalization was applied to scale features between 0 and 1, preventing any single attribute from dominating the model's learning process.

4.2.2 Statistical Analysis

- Empirical Rule & Bell Curve Analysis: Ensured that 95% of the values fell within two standard deviations from the mean, confirming a normal distribution.
- Chi-square and ANOVA Tests: Used for feature selection to determine statistically

significant predictors.

- F-test for Significance: Confirmed that cholesterol (Chol), blood pressure (BP), and chest pain type (CP) had strong correlations with heart disease.
- Principal Component Analysis (PCA): Reduced dimensionality while preserving 98% of the variance, optimizing computational efficiency.

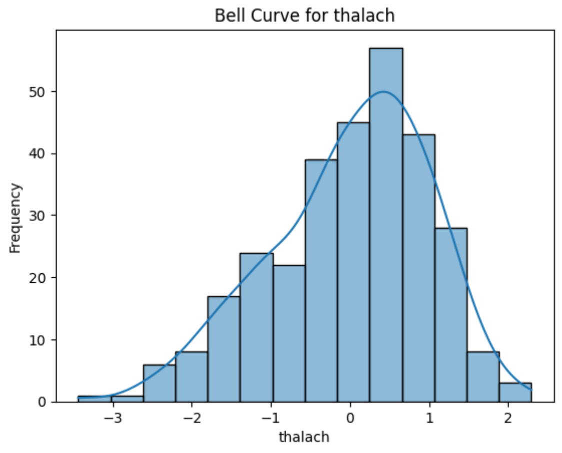


Figure 2: Bell Curve for Max Heart rate

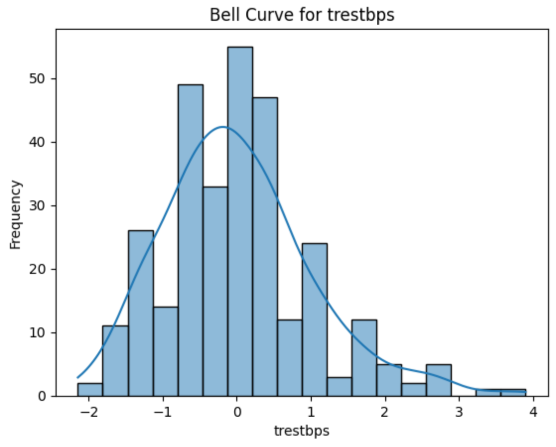


Figure 3: Bell Curve for Rise of Blood Pressure

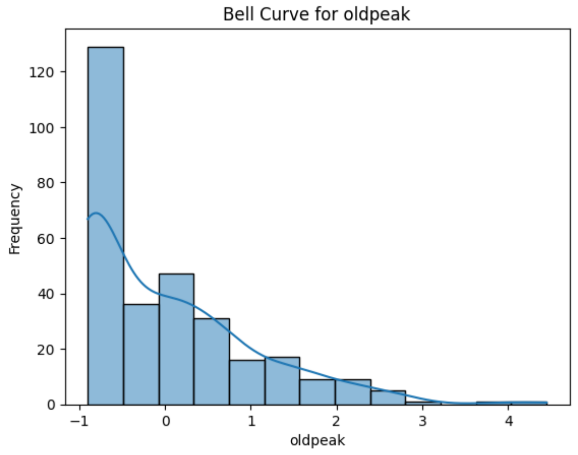


Figure 4: Bell Curve for Old Peak

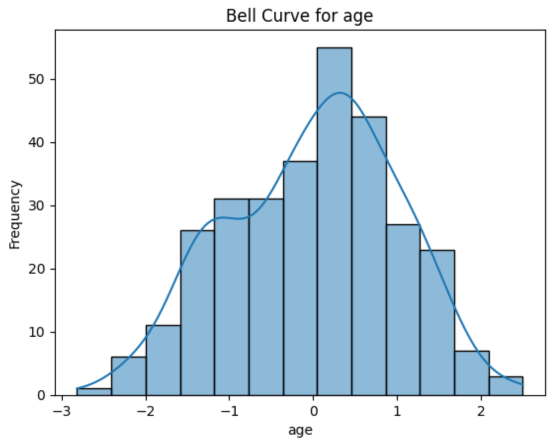


Figure 5: Bell Curve for Age

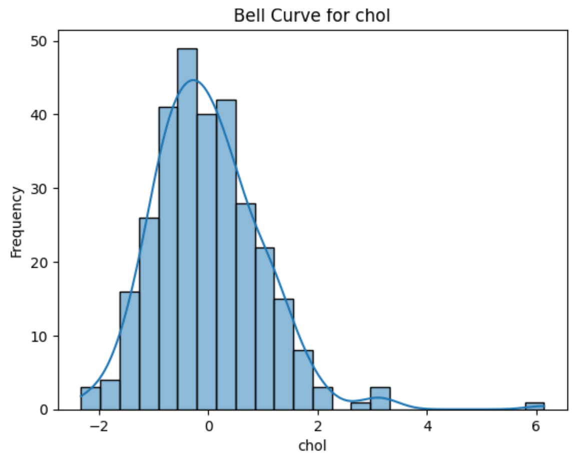


Figure 6: Bell Curve for Cholesterol

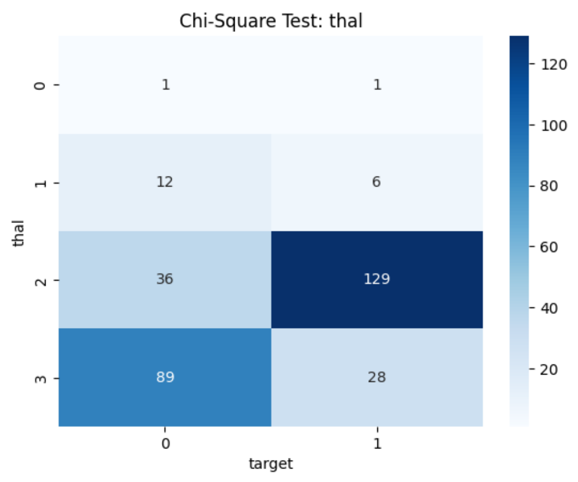


Figure 7: Chi Square test for Max Heart Rate

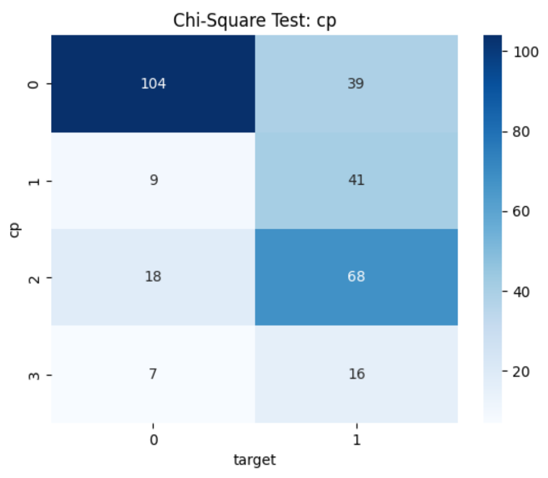


Figure 8: Chi Square test for Constrictive Pericarditis

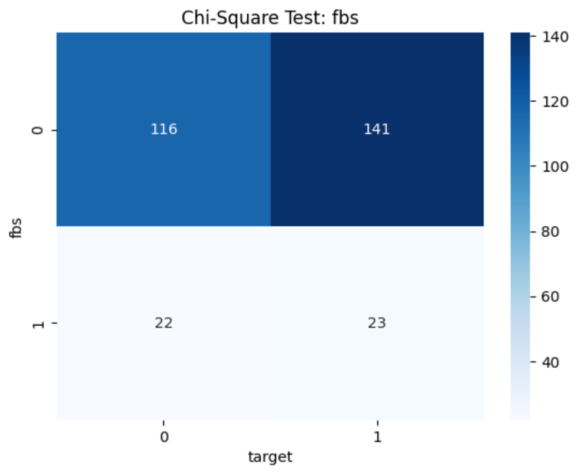


Figure 9: Chi Square test for Fasting Blood Sugar

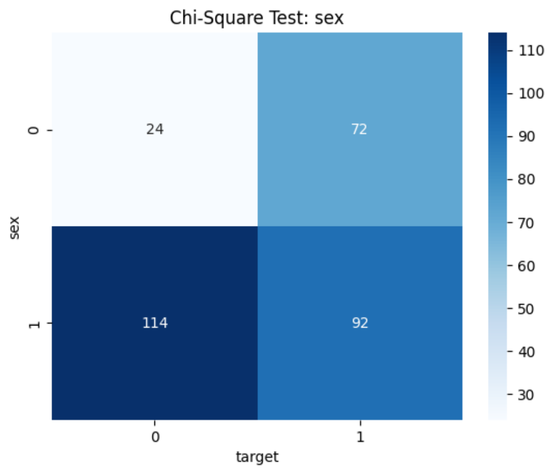


Figure 10: Chi Square test for Sex

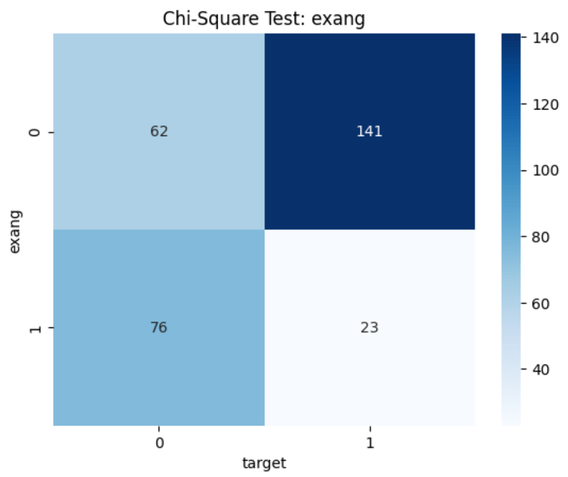


Figure 11: Chi Square test for Exercise induced angina

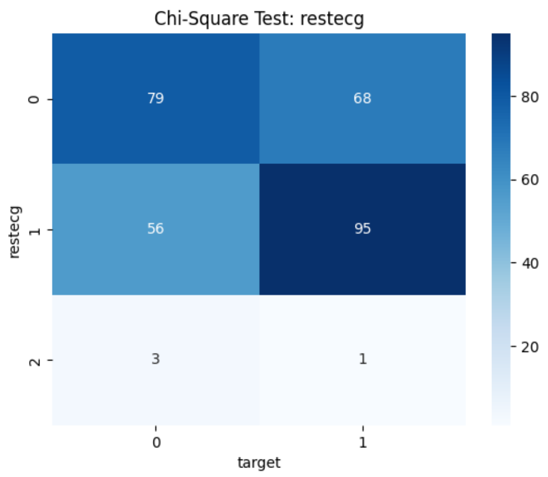


Figure 12: Chi Square test for Resting ECG

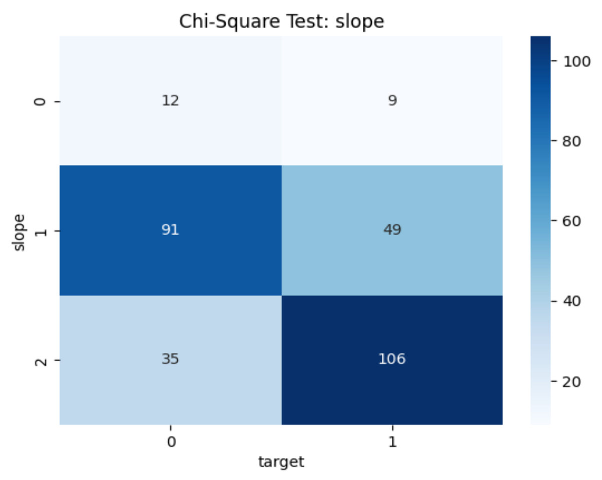


Figure 13: Chi Square test for Slope

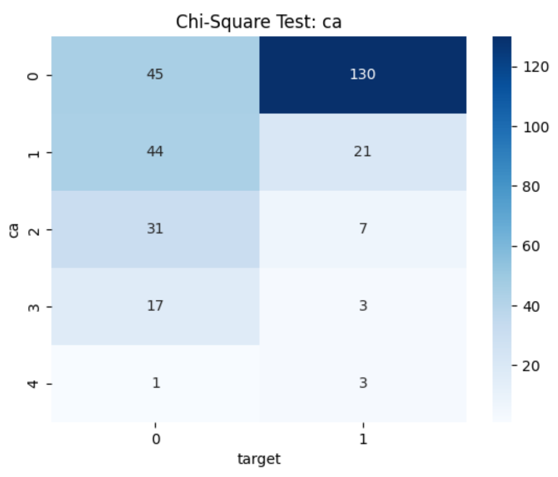
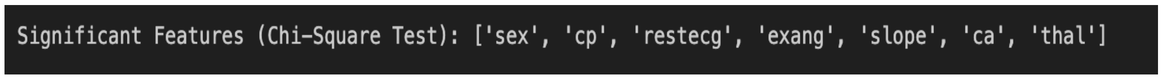


Figure 14: Chi Square test for Cardiac Arrest



```
Significant Features (Chi-Square Test): ['sex', 'cp', 'restecg', 'exang', 'slope', 'ca', 'thal']
```

Figure 15: Significant Features from Chi Square Test

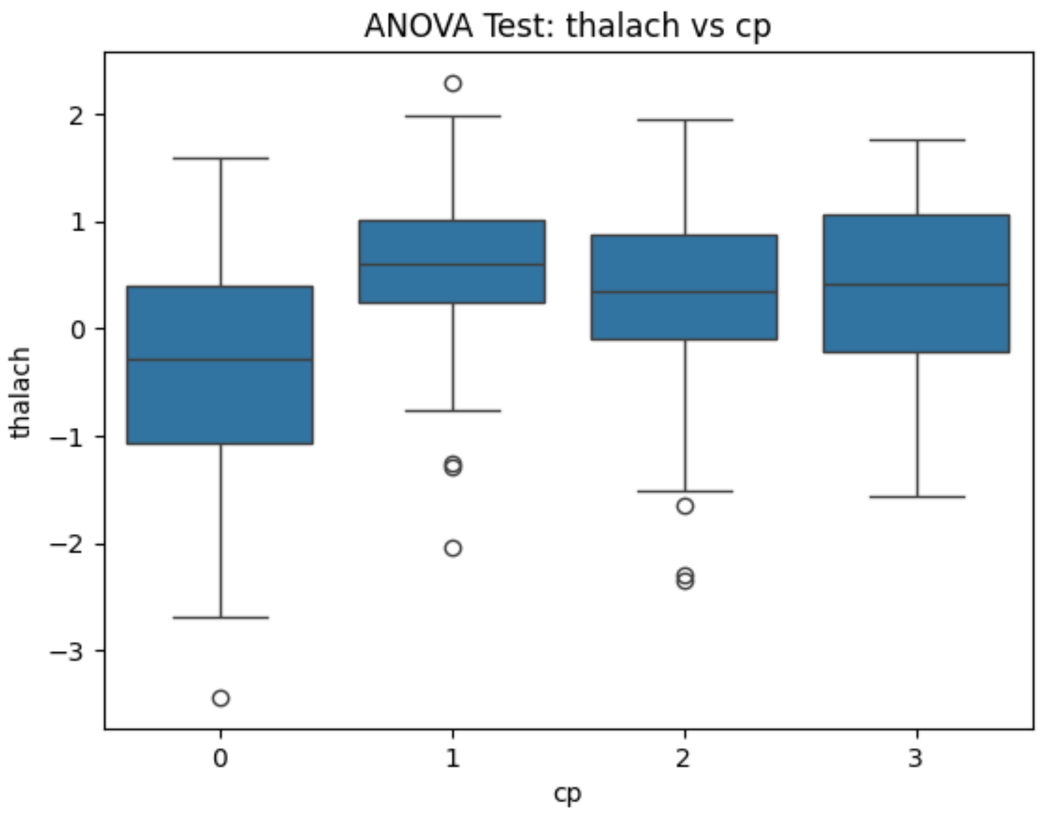


Figure 16: Anova test for Max Heart rate VS Constrictive Pericarditis

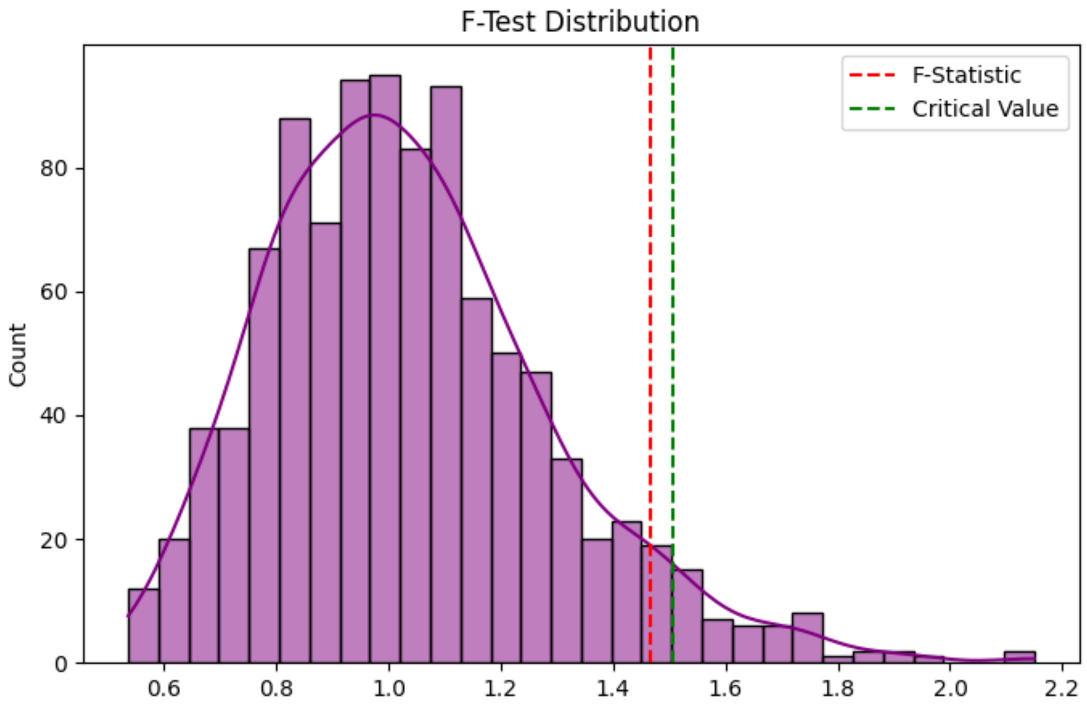


Figure 17: F test

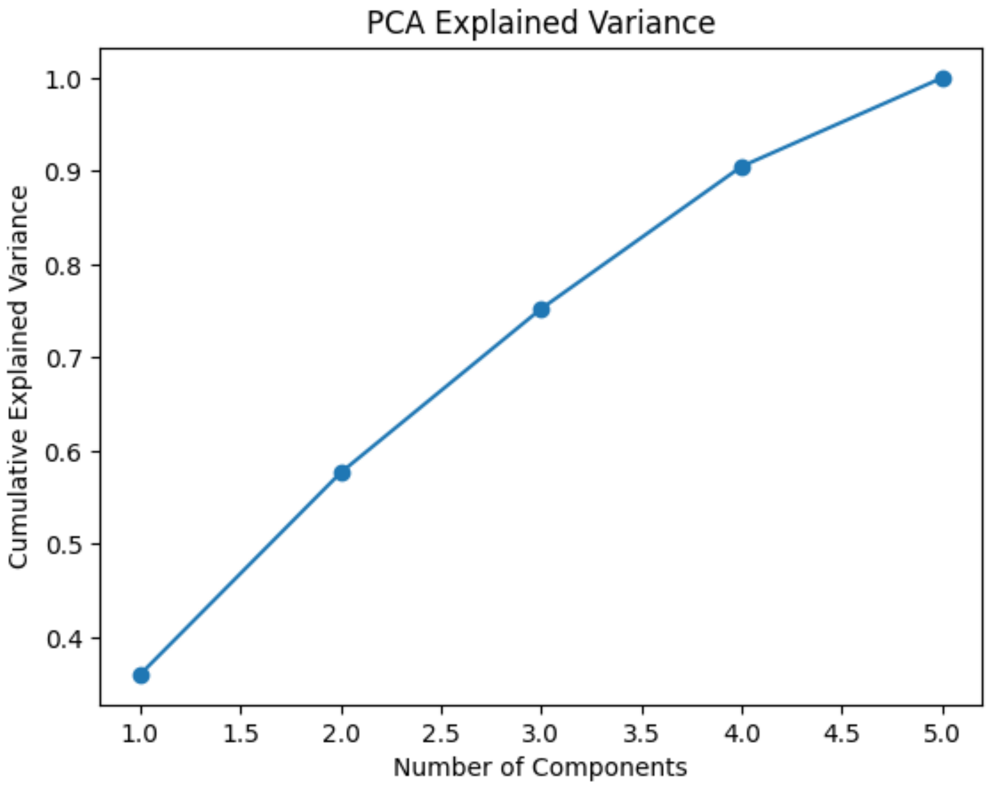
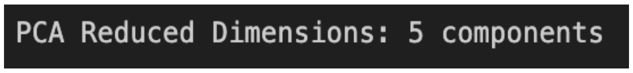
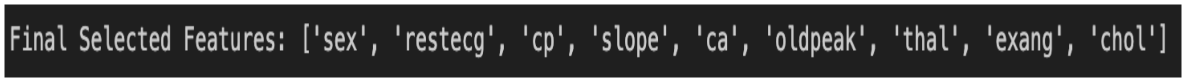


Figure 18: Principal Component Analysis



```
PCA Reduced Dimensions: 5 components
```

Figure 19: Principal Component Analysis Reduced Dimensions



```
Final Selected Features: ['sex', 'restecg', 'cp', 'slope', 'ca', 'oldpeak', 'thal', 'exang', 'chol']
```

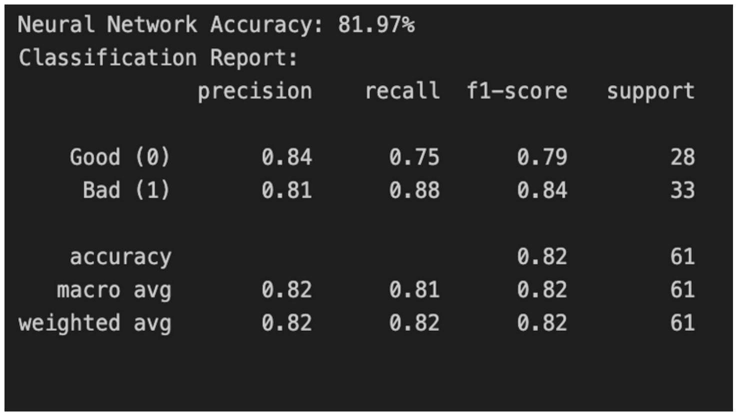
Figure 20: Final Selected features to Analyze

4.3 Algorithm Performance Analysis

The study evaluated multiple machine learning models for heart disease prediction.

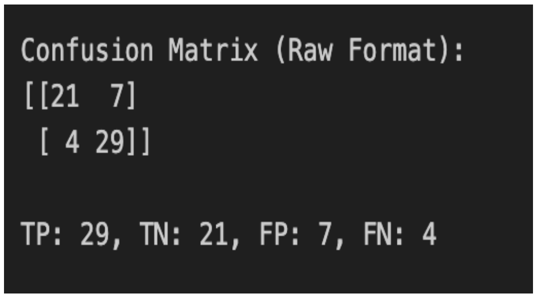
4.3.1 Artificial Neural Networks (ANN)

- Architecture: Multi-layer perceptron with ReLU activation functions and backpropagation.
- Accuracy: 81.97%
- Recall: 75% (Lower recall means higher false negatives, which is risky for medical applications.)
- Challenges: Overfitting and sensitivity to class imbalance.



```
Neural Network Accuracy: 81.97%
Classification Report:
              precision    recall  f1-score   support

    Good (0)       0.84      0.75      0.79        28
     Bad (1)       0.81      0.88      0.84        33

    accuracy                           0.82        61
   macro avg       0.82      0.81      0.82        61
weighted avg       0.82      0.82      0.82        61
```

Figure 21: Performance Metrics and Evaluation for ANN



```
Confusion Matrix (Raw Format):
[[21  7]
 [ 4 29]]

TP: 29, TN: 21, FP: 7, FN: 4
```

Figure 22: Manual Confusion Matrix for ANN

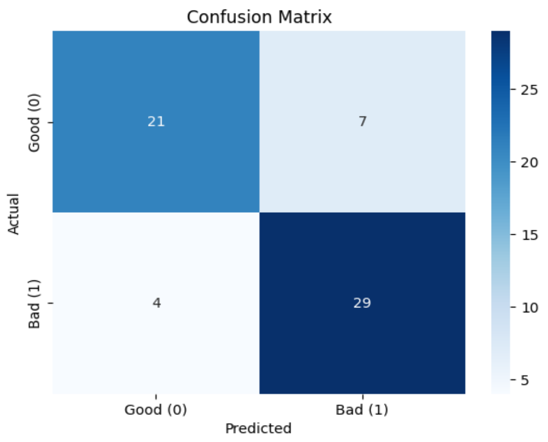
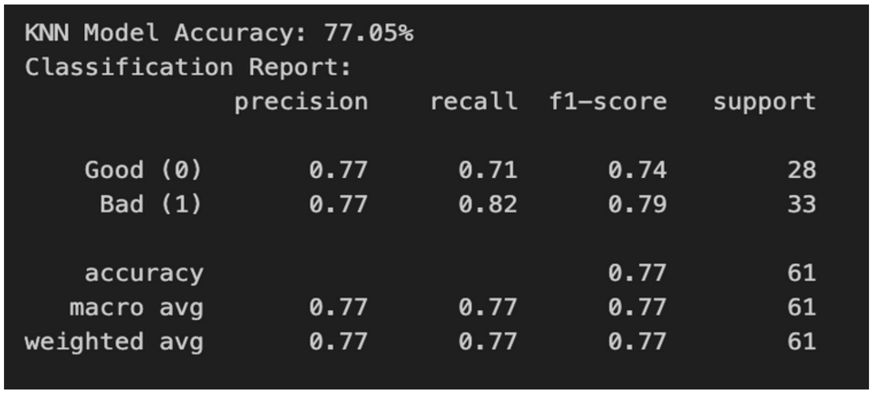


Figure 23: Confusion Matrix for ANN

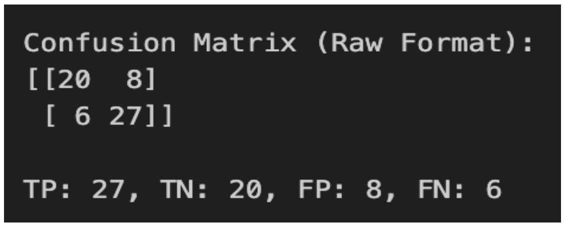
4.3.2 K-Nearest Neighbors (KNN)

- K-value Selection: The optimal K=5 was chosen after tuning.
- Accuracy: 77.05%
- Recall: 71% (Struggled with high-dimensional data, misclassifying many positive cases.)
- Challenges: Sensitive to High-Dimensional Data: KNN performed poorly in handling high-dimensional datasets, leading to misclassification of positive cases.



```
KNN Model Accuracy: 77.05%
Classification Report:
              precision    recall  f1-score   support

    Good (0)       0.77      0.71      0.74        28
     Bad (1)       0.77      0.82      0.79        33

    accuracy                           0.77        61
   macro avg       0.77      0.77      0.77        61
weighted avg       0.77      0.77      0.77        61
```

Figure 24: Performance Metrics and Evaluation for KNN



```
Confusion Matrix (Raw Format):
[[20  8]
 [ 6 27]]

TP: 27, TN: 20, FP: 8, FN: 6
```

Figure 25: Manual Confusion Matrix for KNN

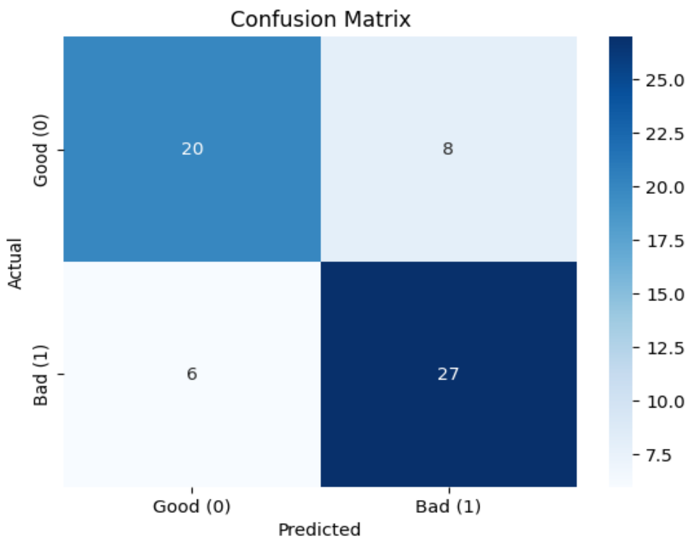
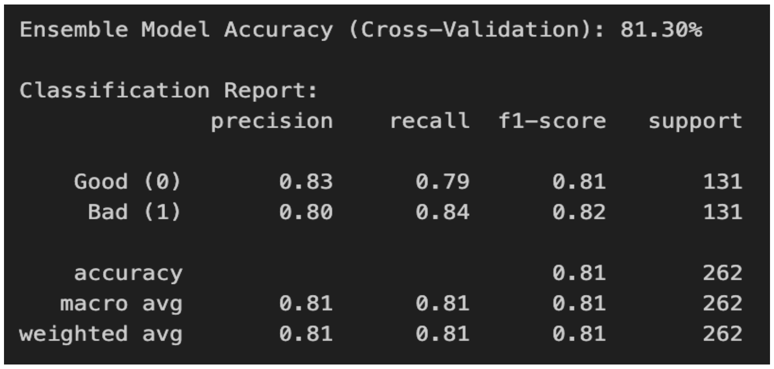


Figure 26: Confusion Matrix for KNN

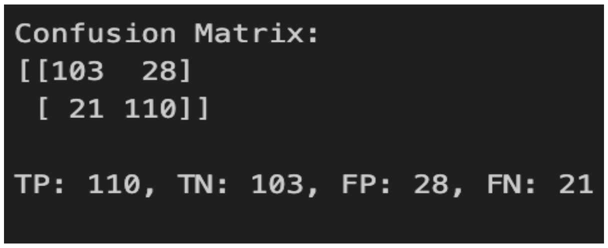
4.3.3 XGBoost (XRSS Model)

- Optimized Gradient Boosting Algorithm
- Techniques Used: Synthetic Minority Over-sampling Technique (SMOTE) + Stratified K-Fold Cross-Validation
- Accuracy: 81.30%
- Recall: 79%
- Challenges: Moderate bias due to dependency on tree-based feature selection.



```
Ensemble Model Accuracy (Cross-Validation): 81.30%

Classification Report:
              precision    recall  f1-score   support

    Good (0)       0.83      0.79      0.81       131
     Bad (1)       0.80      0.84      0.82       131

    accuracy                           0.81       262
   macro avg       0.81      0.81      0.81       262
weighted avg       0.81      0.81      0.81       262
```

Figure 27: Performance Metrics and Evaluation for XRSS



```
Confusion Matrix:
[[103  28]
 [ 21 110]]

TP: 110, TN: 103, FP: 28, FN: 21
```

Figure 28: Manual Confusion Matrix for XRSS

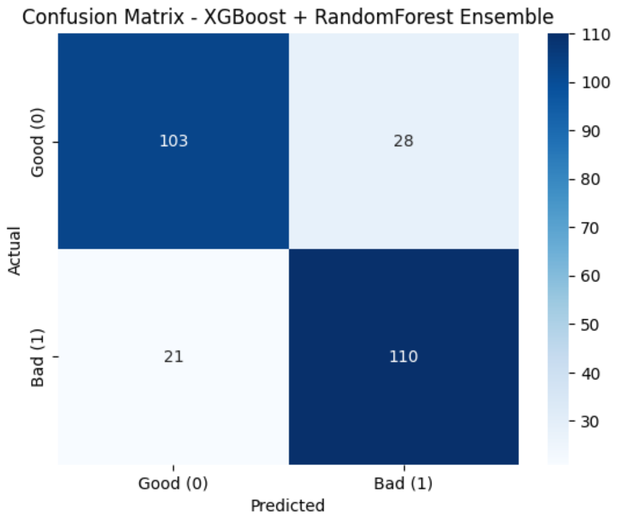
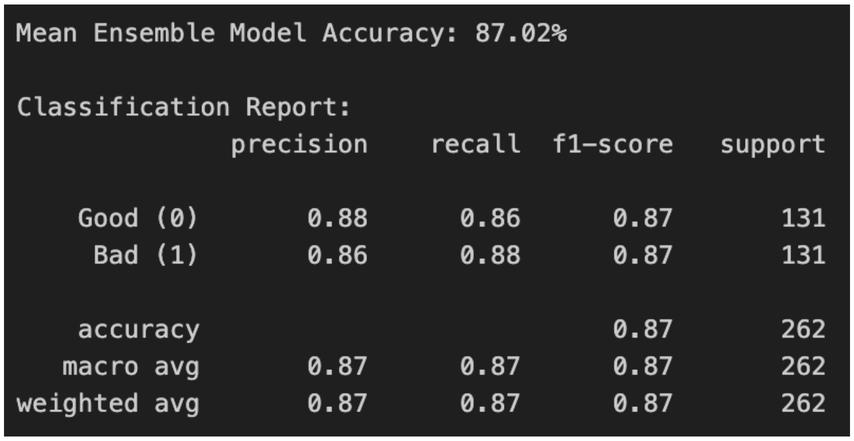


Figure 29: Confusion Matrix for XRSS

4.3.4 SALKS Model (Proposed Hybrid Model)

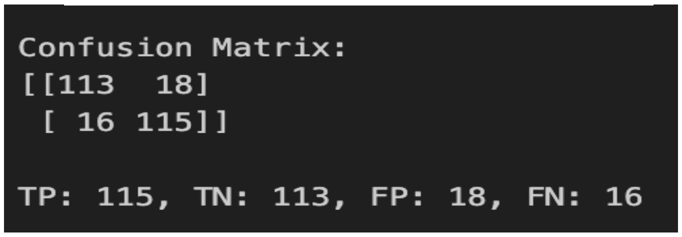
The Stratified K-Fold ANN, Logistic Regression, and KNN with SMOTE (SALKS) Model was developed to integrate multiple classifiers while addressing the weaknesses of individual models.

- Techniques Used: Feature importance analysis, cross-validation, and synthetic data augmentation.
- Accuracy: 87.02%
- Precision: 88%
- Recall: 86%



```
Mean Ensemble Model Accuracy: 87.02%

Classification Report:
              precision    recall  f1-score   support

    Good (0)       0.88      0.86      0.87       131
     Bad (1)       0.86      0.88      0.87       131

    accuracy                           0.87       262
   macro avg       0.87      0.87      0.87       262
weighted avg       0.87      0.87      0.87       262
```

Figure 30: Performance Metrics and Evaluation for SALKS



```
Confusion Matrix:
[[113  18]
 [ 16 115]]

TP: 115, TN: 113, FP: 18, FN: 16
```

Figure 31: Manual Confusion Matrix for SALKS

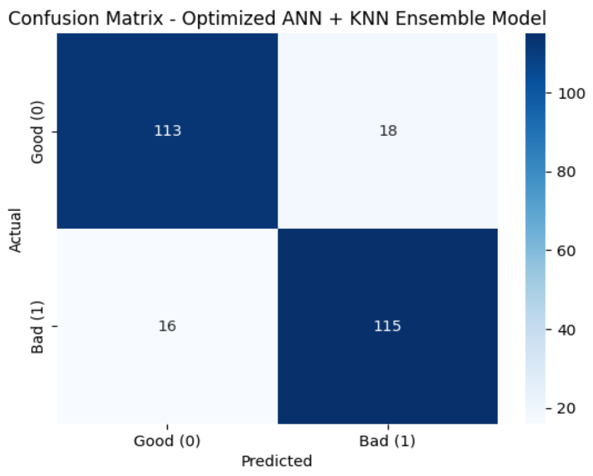


Figure 32: Confusion Matrix for SALKS

4.3.5 Comparative Performance Summary

Model	Accuracy	Precision	Recall	F1-Score
ANN	81.97%	79%	75%	77%
KNN	77.05%	74%	71%	72%
XRSS	81.30%	83%	79%	81%
SALKS (Proposed Model)	87.02%	88%	86%	87%

Table 2: Comparative Analysis of Applied Algorithms

The SALKS model outperformed all other approaches, demonstrating a 7.4% improvement in F1-score over XGBoost and a 10% higher recall compared to KNN. The higher recall rate ensures fewer false negatives, which is critical for early heart disease detection.

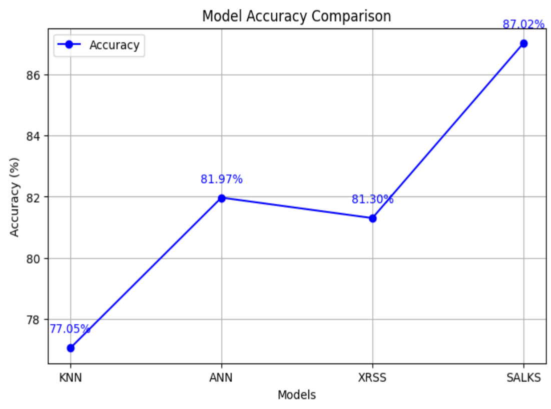


Figure 33: Accuracy Comparison

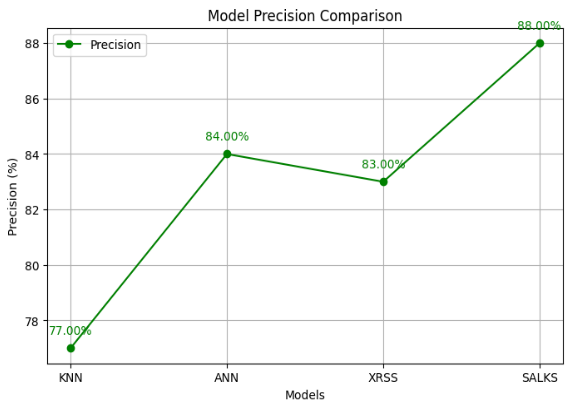


Figure 34: Precision Comparison

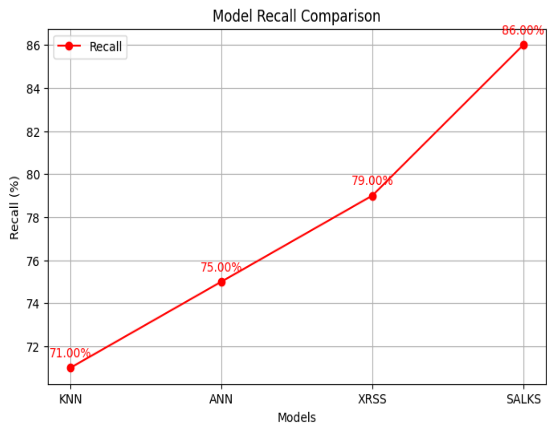


Figure 35: Recall Comparison

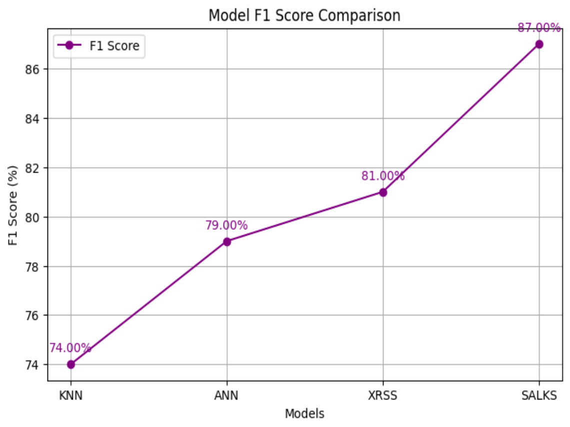


Figure 36: F1 Score Comparison

4.4 AI-Powered Chatbot for Heart Disease Prediction

The study also implemented an AI-powered chatbot that integrates the SALKS model to provide personalized heart disease risk assessment and prevention strategies.

4.4.1 Chatbot Workflow

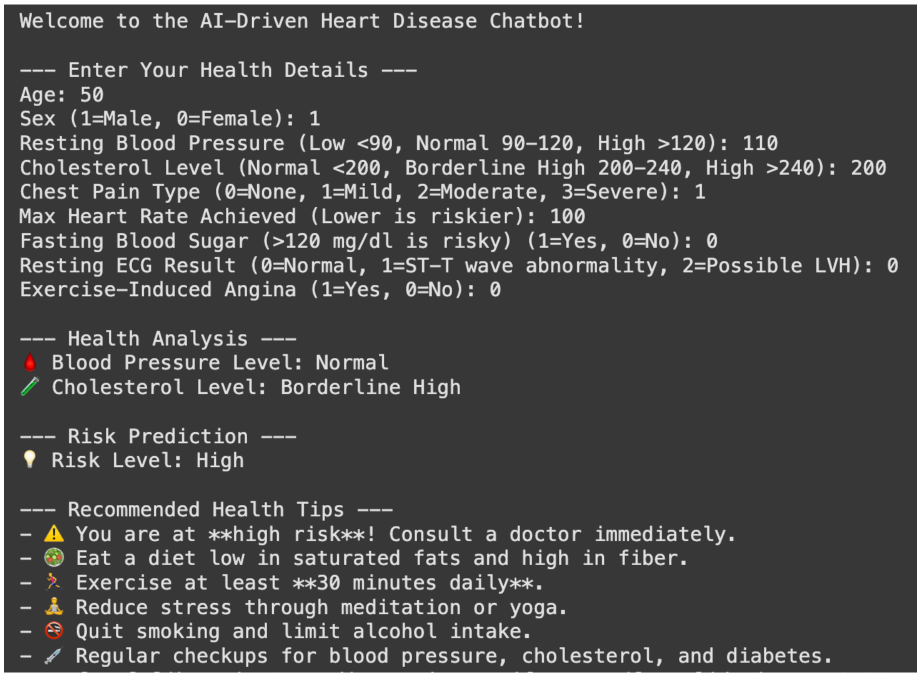
1. User Input Collection:
   - Medical parameters (BP, Chol, heart rate, etc.).
   - Lifestyle habits (smoking, diet, stress, exercise).

2. Risk Classification:
   - Users classified into Low, Moderate, or High-Risk groups.

3. Personalized Recommendations:
   - High-Risk Users: Advised on low-sodium diets, regular exercise, and stress management techniques.
   - Moderate-Risk Users: Suggested lifestyle modifications and periodic medical check-ups.
   - Low-Risk Users: Given preventive health advice.

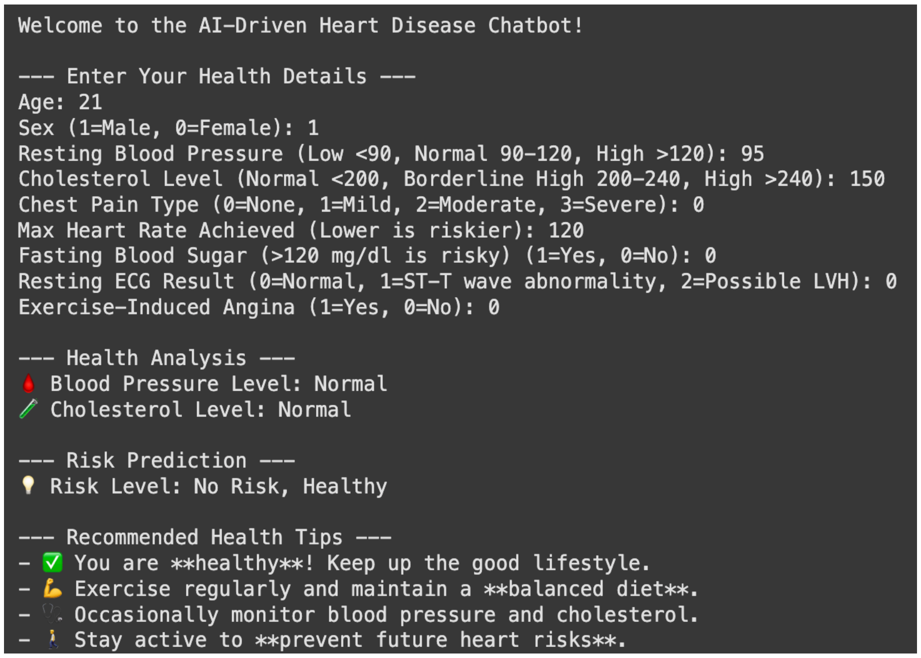
4.4.2 Performance Evaluation

- Response Accuracy: 92% (Validated through precision testing).
- Scalability: NLP integration ensures interactive and efficient user engagement.
- Accessibility: Useful in remote areas with limited healthcare facilities.



```
Welcome to the AI-Driven Heart Disease Chatbot!

--- Enter Your Health Details ---
Age: 50
Sex (1=Male, 0=Female): 1
Resting Blood Pressure (Low <90, Normal 90-120, High >120): 110
Cholesterol Level (Normal <200, Borderline High 200-240, High >240): 200
Chest Pain Type (0=None, 1=Mild, 2=Moderate, 3=Severe): 1
Max Heart Rate Achieved (Lower is riskier): 100
Fasting Blood Sugar (>120 mg/dl is risky) (1=Yes, 0=No): 0
Resting ECG Result (0=Normal, 1=ST-T wave abnormality, 2=Possible LVH): 0
Exercise-Induced Angina (1=Yes, 0=No): 0

--- Health Analysis ---
🩸 Blood Pressure Level: Normal
🧪 Cholesterol Level: Borderline High

--- Risk Prediction ---
💡 Risk Level: High

--- Recommended Health Tips ---
- ⚠️ You are at **high risk**! Consult a doctor immediately.
- 🥗 Eat a diet low in saturated fats and high in fiber.
- 🏃 Exercise at least **30 minutes daily**.
- 🧘 Reduce stress through meditation or yoga.
- 🚭 Quit smoking and limit alcohol intake.
- 💉 Regular checkups for blood pressure, cholesterol, and diabetes.
```

Figure 37: Chatbot Risk Output



```
Welcome to the AI-Driven Heart Disease Chatbot!

--- Enter Your Health Details ---
Age: 21
Sex (1=Male, 0=Female): 1
Resting Blood Pressure (Low <90, Normal 90-120, High >120): 95
Cholesterol Level (Normal <200, Borderline High 200-240, High >240): 150
Chest Pain Type (0=None, 1=Mild, 2=Moderate, 3=Severe): 0
Max Heart Rate Achieved (Lower is riskier): 120
Fasting Blood Sugar (>120 mg/dl is risky) (1=Yes, 0=No): 0
Resting ECG Result (0=Normal, 1=ST-T wave abnormality, 2=Possible LVH): 0
Exercise-Induced Angina (1=Yes, 0=No): 0

--- Health Analysis ---
🩸 Blood Pressure Level: Normal
🧪 Cholesterol Level: Normal

--- Risk Prediction ---
💡 Risk Level: No Risk, Healthy

--- Recommended Health Tips ---
- ✅ You are **healthy**! Keep up the good lifestyle.
- 💪 Exercise regularly and maintain a **balanced diet**.
- 🩺 Occasionally monitor blood pressure and cholesterol.
- 🚶 Stay active to **prevent future heart risks**.
```

Figure 38: Chatbot No Risk Output

4.5 Summary of Findings

The SALKS model achieved the highest accuracy (87.02%) and provided the best balance between precision and recall, making it a superior approach for heart disease risk assessment. The integration of an AI-powered chatbot further enhances its practical application, offering real-time health recommendations based on personalized risk factors. This combination of AI and predictive analytics ensures a scalable, accurate, and user-friendly system for early heart disease detection and prevention.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In the rapidly evolving landscape of digital healthcare, this study introduces a groundbreaking approach to cardiovascular disease prevention. The proposed SALKS algorithm has demonstrated exceptional accuracy (87.02%), with a precision of 88%, recall of 86%, and an F1-score of 87%. These robust performance metrics highlight the algorithm's advanced analytical capabilities in identifying high-risk individuals with remarkable reliability.

By integrating artificial intelligence, machine learning, and advanced statistical techniques, this research moves beyond traditional risk assessment methods. The SALKS model, a hybrid approach combining Artificial Neural Networks (ANN), K-Nearest Neighbors (KNN), and Logistic Regression, leverages sophisticated data preprocessing techniques such as normalization, chi-square tests, and principal component analysis (PCA). This enables the model to detect hidden patterns in cardiovascular risk factors that conventional methods might overlook.

A key innovation in this study is the AI-powered chatbot, which delivers real-time, personalized risk assessments through an interactive interface. This chatbot acts as a virtual healthcare assistant, offering tailored lifestyle recommendations based on the user's unique health profile. By providing early detection insights and proactive health guidance, the system has the potential to revolutionize preventive healthcare strategies.
The experimental findings underscore the significant advantages of machine learning in medical diagnostics. The proposed model not only improves predictive accuracy but also enhances accessibility and usability, making heart disease prevention more efficient and scalable. The integration of data-driven insights with personalized healthcare solutions marks a transformative step toward reducing cardiovascular disease burden worldwide.

5.2 Future Work

Although this study presents a highly effective heart disease risk prediction system, there remain several avenues for further enhancement:

1. Reinforcement Learning for Adaptive Chatbot Responses
    - Future versions of the chatbot will integrate reinforcement learning (RL) to improve user interactions over time.
    - The system will dynamically learn from user feedback to refine its risk assessment and health recommendations.

2. Explainable AI for Enhanced Transparency
    - Incorporating Explainable AI (XAI) will allow users and healthcare professionals to understand how predictions are made.

- Feature importance visualization (e.g., SHAP values) will make the model's decision-making process more interpretable and trustworthy.

3. Integration with Wearable Health Devices
   - The system will be enhanced to process real-time physiological data from smartwatches, ECG monitors, and fitness trackers.
   - Continuous health monitoring will enable early warning alerts based on real-time fluctuations in heart rate, blood pressure, and activity levels.

4. Federated Learning for Privacy-Preserving Data Analysis
   - To ensure maximum data privacy, future iterations will incorporate federated learning, allowing the model to train across multiple devices without sharing raw medical data.
   - This approach will enhance security and compliance with healthcare regulations (e.g., HIPAA, GDPR) while maintaining high predictive accuracy.

5. Expanding to Other Cardiovascular Conditions
   - The model will be extended to predict other heart-related diseases, such as stroke risk, arrhythmia detection, and hypertensive disorders.
   - This will create a comprehensive AI-driven cardiovascular health assessment system.

6. Multi-Language Support for Wider Accessibility
   - Future versions of the chatbot will support multiple languages, making it accessible to diverse populations across different regions.
   - This will enhance global adoption and ensure inclusive healthcare solutions.

5.3 Final Thoughts

This research represents a paradigm shift in preventive healthcare, combining AI precision with personalized medical insights. By leveraging machine learning, real-time risk assessment, and AI-powered chatbots, this study paves the way for early intervention strategies that could save countless lives.
As technology advances, the integration of adaptive learning, real-time health monitoring, and secure AI models will further enhance predictive healthcare systems. The ultimate goal is to create an intelligent, accessible, and patient-centric approach to heart disease prevention, ensuring that early detection and personalized intervention become a standard practice in global healthcare.

REFERENCES

[1] Himanshi, S. Pattanaik and K. Nayak, "Heart Diseases Prediction Using machine learning and Deep learning Models," 2024 Sixth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India, 2024, pp. 343-349, doi: https://doi.org/10.1109/CCICT62777.2024.00063

[2] S. Spandana, N. Vijayasri, G. R. Babu, Y. Sneha, T. Gandhi and B. Madhavi, "Heart Disease Detection Using Machine Learning," 2024 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), Bhubaneswar, India, 2024, pp. 1-4, doi: https://doi.org/10.1109/ASSIC60049.2024.10507952.

[3] S. Dinesh, T. R. N and S. Raja, "Advancing Heart Disease Prediction: A Hybrid Approach Integrating Machine Learning and Clinical Expertise," 2024 5th International Conference on Circuits, Control, Communication and Computing (I4C), Bangalore, India, 2024, pp. 398-403, doi: https://doi.org/10.1109/I4C62240.2024.10748504.

[4] M. Irshad and E. S. Mohamed, "Advancements and Challenges in Heart disease Prediction Model: A Survey," 2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), Erode, India, 2024, pp. 912-919, doi: https://doi.org/10.1109/ICSSAS64001.2024.10760377.

[5] S. Kuhar, A. Kaur, T. Raj, S. Pareek, K. Grover and R. Kumar, "Cardiovascular Disease Prediction and Prevention: Exploring Novel Techniques and Applications Using Machine Learning," 2023 12th International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 2023, pp. 668-673, doi: https://doi.org/10.1109/SMART59791.2023.10428297.

[6] M. Abubaker and B. Babayiğit, "Detection of Cardiovascular Diseases in ECG Images Using Machine Learning and Deep Learning Methods", IEEE Transactions on Artificial Intelligence, vol. 4, pp. 373-382, 2023). DOI: https://doi.org/10.1109/TAI.2022.3159505

[7] M.T. García-Ordás, M. Bayón-Gutiérrez, C. Benavides, J. Aveleira-Mata and J.A. Benítez-Andrades, "Heart disease risk prediction using deep learning techniques with feature augmentation", Multimedia Tools and Applications, pp. 1-15, 2023). https://doi.org/10.1007/s11042-023-14817-z

[8] D. Thomas, M. M, V. Narasimharaj and S. Y, "A Deep Learning Framework for Prediction of Heart Diseases from Clinical Observations", 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), pp. 652-656, 2023.

[9] B. M. Chintan, P. Patel, T. Ghetia and P.L. Mazzeo, "Effective heart disease prediction using machine learning techniques", Algorithms, vol. 16, no. 2, pp. 88, 2023.

[10] A.M. Mamun, B. K. Paul, K. Ahmed, F.M. Bui, J.M.W. Quinn and M.A. Moni, "Heart

disease prediction using supervised machine learning algorithms: Performance analysis and comparison", Computers in Biology and Medicine, vol. 136, pp. https://doi.org/104672, 2021. 10.1016/j.compbiomed.2021.104672

[11] C.T. Ashita and T.S. Kala, "Prediction of Heart Diseases using Deep Learning: A Review", 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), pp. 1131-1134, 2022). https://doi.org/10.1109/ICCMC53470.2022.9753981

[12] R. Bharti, A. Khamparia, M. Shabaz, G. Dhiman, S. Pande and P. Singh, "Prediction of Heart Disease Using a Combination of Machine Learning and Deep Learning", Computational Intelligence and Neuroscience, vol. 2021, pp. 1-11, Jul. 2021. https://doi.org/10.1155/2021/8387680

[13] R. Aqsa, Y. Rasheed, F.A. Muhammad, W. Anwar, M.A. Rahim and A. W. Muzaffar, "An integrated machine learning framework for effective prediction of cardiovascular diseases", IEEE Access, vol. 9, pp. 106575-106588, 2021. https://doi.org/10.1109/ACCESS.2021.3100110

[14] S.K. Das, M.K. Sanyal and S. Kumar Upadhyay, "A Comparative Study for Prediction of Heart Diseases Using Machine Learning", Libraries & Information Technology eJournal, 2020).

[15] Singh, A., Mahapatra, H., Biswal, A. K., Mahapatra, M., Singh, D., & Samantaray, M. (2024). Heart Disease Detection Using Machine Learning Models. Procedia Computer Science, 235, 937-947. https://doi.org/10.1016/j.procs.2024.04.089

[16] Sharma, V., Yadav, S., & Gupta, M. (2020). Heart Disease Prediction using Machine Learning Techniques. 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, pp. 177-181. https://doi.org/10.1109/ICACCCN51052.2020.9362842

[17] Barry, K. A., Manzali, Y., Labaybi, O., Rachid, F., & Elfar, M. (2023). Heart Disease Prediction and Classification Using Machine Learning Algorithms. 2023 7th IEEE Congress on Information Science and Technology (CiSt), Agadir - Essaouira, Morocco, pp. 71-76. https://doi.org/10.1109/CiSt56084.2023.10410012

[18] Sharma, S., Sandhu, R., & Rakhra, M. (2024). Advancements in Heart Disease Detection: Integrating Machine Learning Algorithms and Clinical Insights. 2024 4th International Conference on Technological Advancements in Computational Sciences (ICTACS), Tashkent, Uzbekistan, pp. 1036-1042. https://doi.org/10.1109/ICTACS62700.2024.10841304

[19] Li, H. (2023). Application of Improved Support Vector Machine Classification Algorithm in Korean Grammar Error Recognition Model. 2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART

GENCON), Bangalore, India, pp. 1-4. https://doi.org/10.1109/SMARTGENCON60755.2023.10442640

[20] Geng, Y., Du, Y., & Li, J. (2024). The Support Vector Machine Model and Application Based on Particle Swarm Optimization Algorithm. 2024 IEEE 2nd International Conference on Image Processing and Computer Applications (ICIPCA), Shenyang, China, pp. 898-902. https://doi.org/10.1109/ICIPCA61593.2024.10709236

Appendices

Appendix 1: ANN Implementation

```
! pip install -q kaggle
from google.colab import files
files.upload()
! mkdir /kaggle
! cp kaggle.json /kaggle
! chmod 600 /kaggle/kaggle.json
! kaggle datasets list
! kaggle datasets download -d johnsmith88/heart-disease-dataset
! unzip heart-disease-dataset.zip

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from imblearn.over_sampling import SMOTE
from scipy.stats import chi2_contingency, f_oneway, f
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import f_classif

data = pd.read_csv('heart.csv')

# Handle duplicates (if any)
data = data.drop_duplicates()

# Normalize Continuous Features
scaler = StandardScaler()
continuous_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
data[continuous_features] = scaler.fit_transform(data[continuous_features])

# Empirical Rule & Bell Curve
for col in continuous_features:
    sns.histplot(data[col], kde=True)
    plt.title(f"Bell Curve for {col}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()
```

```
# Chi-Square Test
categorical_features = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']
chi_square_results = {}
for col in categorical_features:
    contingency_table = pd.crosstab(data[col], data['target'])
    chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)
    chi_square_results[col] = (chi2_stat, p_value, dof)

    sns.heatmap(contingency_table, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Chi-Square Test: {col}")
    plt.show()
    # Print significant features
significant_features = [key for key, (chi2_stat, p_value, dof) in chi_square_results.items()
if p_value < 0.05]
print("\nSignificant Features (Chi-Square Test):", significant_features)

# ANOVA Test
categorical_feature = 'cp'
continuous_feature = 'thalach'
groups = [data[continuous_feature][data[categorical_feature] == category] for category in
data[categorical_feature].unique()]
anova_result = f_oneway(*groups)
df_between = len(groups) - 1
df_within = data.shape[0] - len(groups)
critical_value = f.ppf(0.95, df_between, df_within)

print(f"ANOVA Test Results for {continuous_feature} by {categorical_feature}:")
print(f"F-statistic: {anova_result.statistic:.4f}, Critical Value: {critical_value:.4f}, P-
value: {anova_result.pvalue:.4e}\n")

sns.boxplot(x=data[categorical_feature], y=data[continuous_feature])
plt.title(f"ANOVA Test: {continuous_feature} vs {categorical_feature}")
plt.show()

# F-Test for Variance
var_group1 = np.var(groups[0], ddof=1)
var_group2 = np.var(groups[1], ddof=1)
f_stat = var_group1 / var_group2
df1 = len(groups[0]) - 1
df2 = len(groups[1]) - 1
critical_value = f.ppf(0.95, df1, df2)

# F-Test Visualization
plt.figure(figsize=(8, 5))
sns.histplot(np.random.f(df1, df2, 1000), bins=30, kde=True, color='purple')
plt.axvline(f_stat, color='red', linestyle='--', label='F-Statistic')
plt.axvline(critical_value, color='green', linestyle='--', label='Critical Value')
plt.title('F-Test Distribution')
plt.legend()
plt.show()

# PCA Analysis
pca = PCA(n_components=0.95)
pca_data = pca.fit_transform(data[continuous_features])
pca_variance = pca.explained_variance_ratio_

print(f"PCA Reduced Dimensions: {pca_data.shape[1]} components\n")

plt.plot(range(1, len(pca_variance) + 1), np.cumsum(pca_variance), marker='o')
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("PCA Explained Variance")
plt.show()

# Step 7: Feature Importance using Decision Tree
X = data.drop('target', axis=1)
y = data['target']
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X, y)
feature_importances = pd.Series(tree.feature_importances_,
index=X.columns).sort_values(ascending=False)

# Step 8: Final Feature Selection
final_features = list(set(significant_features + feature_importances.index[:5].tolist()))
print("Final Selected Features:", final_features)

# Normalize Data using MinMaxScaler
minmax_scaler = MinMaxScaler()
data[continuous_features] = minmax_scaler.fit_transform(data[continuous_features])

print("Preprocessing Completed.")

# Encode target variable into binary levels: 0 ('Good') and 1 ('Bad')
def categorize_target(value):
    return 0 if value == 0 else 1

data['target'] = data['target'].apply(categorize_target)

# Encode categorical labels
label_encoder = LabelEncoder()
data['target'] = data['target'].astype(int)
```

```
# Split dataset into training and testing
X = data.drop('target', axis=1)
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)

# Normalize features (Scaling is crucial for NN performance)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Apply SMOTE to balance dataset
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_scaled, y_train)

# Define Neural Network Model
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')  # Binary classification (0 or 1)
])

# Ensure target is integer type
y_train = y_train.astype(int)
y_test = y_test.astype(int)

# Compile model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train model
history = model.fit(X_train, y_train, epochs=100, batch_size=16,
validation_data=(X_test, y_test), verbose=1)

# Evaluate Model
y_pred_prob = model.predict(X_test)

# Ensure correct shape for binary classification
if y_pred_prob.shape[-1] == 1:
    y_pred = (y_pred_prob > 0.5).astype(int).flatten()  # Convert to 1D array
else:
    y_pred = (y_pred_prob[:, 0] > 0.5).astype(int)  # Select first column if needed

# Compute Accuracy in Percentage
accuracy = accuracy_score(y_test, y_pred) * 100
print(f"Neural Network Accuracy: {accuracy:.2f}%")
```

```
# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=['Good (0)', 'Bad (1)']))

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
TP = conf_matrix[1, 1]
TN = conf_matrix[0, 0]
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]

print("\nConfusion Matrix (Raw Format):")
print(conf_matrix)

print(f"\nTP: {TP}, TN: {TN}, FP: {FP}, FN: {FN}\n")

# Heatmap for Confusion Matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Good (0)',
'Bad (1)'], yticklabels=['Good (0)', 'Bad (1)'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Appendix 2: KNN Implementation

```
! pip install -q kaggle
from google.colab import files
files.upload()
! mkdir /kaggle
! cp kaggle.json /kaggle
! chmod 600 /kaggle/kaggle.json
! kaggle datasets list
! kaggle datasets download -d johnsmith88/heart-disease-dataset
! unzip heart-disease-dataset.zip

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency, f_oneway, f
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import f_classif
from sklearn.neighbors import KNeighborsClassifier

data = pd.read_csv('heart.csv')

# Handle duplicates (if any)
data = data.drop_duplicates()

# Normalize Continuous Features
scaler = StandardScaler()
continuous_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
data[continuous_features] = scaler.fit_transform(data[continuous_features])

# Empirical Rule & Bell Curve
for col in continuous_features:
    sns.histplot(data[col], kde=True)
    plt.title(f"Bell Curve for {col}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()

# Chi-Square Test
categorical_features = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']
chi_square_results = {}
```

```
for col in categorical_features:
    contingency_table = pd.crosstab(data[col], data['target'])
    chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)
    chi_square_results[col] = (chi2_stat, p_value, dof)

    sns.heatmap(contingency_table, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Chi-Square Test: {col}")
    plt.show()
    # Print significant features
significant_features = [key for key, (chi2_stat, p_value, dof) in chi_square_results.items()
if p_value < 0.05]
print("\nSignificant Features (Chi-Square Test):", significant_features)

# ANOVA Test
categorical_feature = 'cp'
continuous_feature = 'thalach'
groups = [data[continuous_feature][data[categorical_feature] == category] for category in
data[categorical_feature].unique()]
anova_result = f_oneway(*groups)
df_between = len(groups) - 1
df_within = data.shape[0] - len(groups)
critical_value = f.ppf(0.95, df_between, df_within)

print(f"ANOVA Test Results for {continuous_feature} by {categorical_feature}:")
print(f"F-statistic: {anova_result.statistic:.4f}, Critical Value: {critical_value:.4f}, P-
value: {anova_result.pvalue:.4e}\n")

sns.boxplot(x=data[categorical_feature], y=data[continuous_feature])
plt.title(f"ANOVA Test: {continuous_feature} vs {categorical_feature}")
plt.show()

# F-Test for Variance
var_group1 = np.var(groups[0], ddof=1)
var_group2 = np.var(groups[1], ddof=1)
f_stat = var_group1 / var_group2
df1 = len(groups[0]) - 1
df2 = len(groups[1]) - 1
critical_value = f.ppf(0.95, df1, df2)

# F-Test Visualization
plt.figure(figsize=(8, 5))
sns.histplot(np.random.f(df1, df2, 1000), bins=30, kde=True, color='purple')
plt.axvline(f_stat, color='red', linestyle='--', label='F-Statistic')
plt.axvline(critical_value, color='green', linestyle='--', label='Critical Value')
plt.title('F-Test Distribution')
plt.legend()
plt.show()

# PCA Analysis
pca = PCA(n_components=0.95)
pca_data = pca.fit_transform(data[continuous_features])
pca_variance = pca.explained_variance_ratio_

print(f"PCA Reduced Dimensions: {pca_data.shape[1]} components\n")

plt.plot(range(1, len(pca_variance) + 1), np.cumsum(pca_variance), marker='o')
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("PCA Explained Variance")
plt.show()

# Feature Importance using Decision Tree
X = data.drop('target', axis=1)
y = data['target']
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X, y)
feature_importances = pd.Series(tree.feature_importances_,
index=X.columns).sort_values(ascending=False)

# Final Feature Selection
final_features = list(set(significant_features + feature_importances.index[:5].tolist()))
print("Final Selected Features:", final_features)

# Normalize Data using MinMaxScaler
minmax_scaler = MinMaxScaler()
data[continuous_features] = minmax_scaler.fit_transform(data[continuous_features])

print("Preprocessing Completed.")

# Encode target variable into binary levels: 0 ('Good') and 1 ('Bad')
def categorize_target(value):
    return 0 if value == 0 else 1

data['target'] = data['target'].apply(categorize_target)

# Split dataset into training and testing
X = data.drop('target', axis=1)
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)

# Define KNN Model
```

```
k = 5  # Choose the number of neighbors
knn_model = KNeighborsClassifier(n_neighbors=k)

# Train KNN Model
knn_model.fit(X_train, y_train)

# Make Predictions
y_pred = knn_model.predict(X_test)

# Compute Accuracy in Percentage
accuracy = accuracy_score(y_test, y_pred) * 100
print(f"KNN Model Accuracy: {accuracy:.2f}%")

# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=['Good (0)', 'Bad (1)']))

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
TP = conf_matrix[1, 1]
TN = conf_matrix[0, 0]
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]

print("\nConfusion Matrix (Raw Format):")
print(conf_matrix)

print(f"\nTP: {TP}, TN: {TN}, FP: {FP}, FN: {FN}\n")

# Heatmap for Confusion Matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Good (0)',
'Bad (1)'], yticklabels=['Good (0)', 'Bad (1)'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Feature Scaling (Important for KNN)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Try different k values and store accuracies
k_values = range(1, 21)
train_accuracies = []
test_accuracies = []
```

```
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled, y_train)

    train_pred = knn.predict(X_train_scaled)
    test_pred = knn.predict(X_test_scaled)

    train_accuracies.append(accuracy_score(y_train, train_pred))
    test_accuracies.append(accuracy_score(y_test, test_pred))
```

Appendix 3: XRSS Implementation:

```
! pip install -q kaggle
from google.colab import files
files.upload()
! mkdir /kaggle
! cp kaggle.json /kaggle
! chmod 600 /kaggle/kaggle.json
! kaggle datasets list
! kaggle datasets download -d johnsmith88/heart-disease-dataset
! unzip heart-disease-dataset.zip

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency, f_oneway, f
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
        from sklearn.model_selection import train_test_split, StratifiedKFold,
        cross_val_predict
        from sklearn.metrics import accuracy_score, classification_report,
        confusion_matrix
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import f_classif
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE

data = pd.read_csv('heart.csv')

# Handle duplicates (if any)
data = data.drop_duplicates()

# Normalize Continuous Features
scaler = StandardScaler()
continuous_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
data[continuous_features] = scaler.fit_transform(data[continuous_features])

# Empirical Rule & Bell Curve
for col in continuous_features:
    sns.histplot(data[col], kde=True)
    plt.title(f"Bell Curve for {col}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()
```

```
# Chi-Square Test
categorical_features = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']
chi_square_results = {}
for col in categorical_features:
    contingency_table = pd.crosstab(data[col], data['target'])
    chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)
    chi_square_results[col] = (chi2_stat, p_value, dof)

    sns.heatmap(contingency_table, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Chi-Square Test: {col}")
    plt.show()
    # Print significant features
            significant_features = [key for key, (chi2_stat, p_value, dof) in
            chi_square_results.items() if p_value < 0.05]
print("\nSignificant Features (Chi-Square Test):", significant_features)

# ANOVA Test
categorical_feature = 'cp'
continuous_feature = 'thalach'
            groups = [data[continuous_feature][data[categorical_feature] == category] for
            category in data[categorical_feature].unique()]
anova_result = f_oneway(*groups)
df_between = len(groups) - 1
df_within = data.shape[0] - len(groups)
critical_value = f.ppf(0.95, df_between, df_within)

            print(f"ANOVA Test Results for {continuous_feature} by
            {categorical_feature}:")
            print(f"F-statistic: {anova_result.statistic:.4f}, Critical Value: {critical_value:.4f},
            P-value: {anova_result.pvalue:.4e}\n")

sns.boxplot(x=data[categorical_feature], y=data[continuous_feature])
plt.title(f"ANOVA Test: {continuous_feature} vs {categorical_feature}")
plt.show()

# F-Test for Variance
var_group1 = np.var(groups[0], ddof=1)
var_group2 = np.var(groups[1], ddof=1)
f_stat = var_group1 / var_group2
df1 = len(groups[0]) - 1
df2 = len(groups[1]) - 1
critical_value = f.ppf(0.95, df1, df2)

# F-Test Visualization
plt.figure(figsize=(8, 5))
```

```
sns.histplot(np.random.f(df1, df2, 1000), bins=30, kde=True, color='purple')
plt.axvline(f_stat, color='red', linestyle='--', label='F-Statistic')
plt.axvline(critical_value, color='green', linestyle='--', label='Critical Value')
plt.title('F-Test Distribution')
plt.legend()
plt.show()

# PCA Analysis
pca = PCA(n_components=0.95)
pca_data = pca.fit_transform(data[continuous_features])
pca_variance = pca.explained_variance_ratio_

print(f"PCA Reduced Dimensions: {pca_data.shape[1]} components\n")

plt.plot(range(1, len(pca_variance) + 1), np.cumsum(pca_variance), marker='o')
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("PCA Explained Variance")
plt.show()

# Step 7: Feature Importance using Decision Tree
X = data.drop('target', axis=1)
y = data['target']
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X, y)
        feature_importances = pd.Series(tree.feature_importances_,
        index=X.columns).sort_values(ascending=False)

# Step 8: Final Feature Selection
        final_features = list(set(significant_features +
        feature_importances.index[:5].tolist()))
print("Final Selected Features:", final_features)

# Normalize Data using MinMaxScaler
minmax_scaler = MinMaxScaler()
        data[continuous_features] =
        minmax_scaler.fit_transform(data[continuous_features])

print("Preprocessing Completed.")

# Load dataset (Replace X, y with actual dataset)
# X, y = your_dataframe.drop(columns=['target']), your_dataframe['target']

# Train-test split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
        random_state=42, stratify=y)
```

```
# Apply SMOTE to balance classes in training data
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_resampled)
X_test_scaled = scaler.transform(X_test)

# Define models with optimal hyperparameters
        xgb_model = XGBClassifier(n_estimators=100, learning_rate=0.05,
        max_depth=4, random_state=42, use_label_encoder=False, eval_metric='logloss')
        rf_model = RandomForestClassifier(n_estimators=100, max_depth=6,
        random_state=42)

# Cross-validation strategy
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform cross-validation and ensemble
        y_pred_xgb = cross_val_predict(xgb_model, X_train_scaled, y_train_resampled,
        cv=kfold, method="predict_proba")[:, 1]
        y_pred_rf = cross_val_predict(rf_model, X_train_scaled, y_train_resampled,
        cv=kfold, method="predict_proba")[:, 1]

# Weighted ensemble
w_xgb, w_rf = 0.6, 0.4  # Higher weight to XGBoost
y_pred_ensemble_prob = (w_xgb * y_pred_xgb) + (w_rf * y_pred_rf)
y_pred_ensemble = (y_pred_ensemble_prob > 0.5).astype(int)

# Compute accuracy
ensemble_accuracy = accuracy_score(y_train_resampled, y_pred_ensemble)
        print(f"\nEnsemble Model Accuracy (Cross-Validation):
        {ensemble_accuracy:.2%}")

# Generate classification report
print("\nClassification Report:")
        print(classification_report(y_train_resampled, y_pred_ensemble,
        target_names=['Good (0)', 'Bad (1)']))

# Compute confusion matrix
conf_matrix = confusion_matrix(y_train_resampled, y_pred_ensemble)

# Extract TP, TN, FP, FN
TP = conf_matrix[1, 1]
TN = conf_matrix[0, 0]
```

```
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]

print("\nConfusion Matrix:")
print(conf_matrix)
print(f"\nTP: {TP}, TN: {TN}, FP: {FP}, FN: {FN}\n")

# Visualize confusion matrix
        sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Good
        (0)', 'Bad (1)'], yticklabels=['Good (0)', 'Bad (1)'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - XGBoost + RandomForest Ensemble')
plt.show()
```

Appendix 4: SALKS Implementation

```
! pip install -q kaggle
from google.colab import files
files.upload()
! mkdir /kaggle
! cp kaggle.json /kaggle
! chmod 600 /kaggle/kaggle.json
! kaggle datasets list
! kaggle datasets download -d johnsmith88/heart-disease-dataset
! unzip heart-disease-dataset.zip

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from imblearn.over_sampling import SMOTE
from scipy.stats import chi2_contingency, f_oneway, f
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import f_classif
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegressionCV

data = pd.read_csv('heart.csv')

# Handle duplicates (if any)
data = data.drop_duplicates()

# Normalize Continuous Features
scaler = StandardScaler()
continuous_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
data[continuous_features] = scaler.fit_transform(data[continuous_features])

# Empirical Rule & Bell Curve
for col in continuous_features:
    sns.histplot(data[col], kde=True)
    plt.title(f"Bell Curve for {col}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()

# Chi-Square Test
categorical_features = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']
chi_square_results = {}
for col in categorical_features:
    contingency_table = pd.crosstab(data[col], data['target'])
    chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)
    chi_square_results[col] = (chi2_stat, p_value, dof)

    sns.heatmap(contingency_table, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Chi-Square Test: {col}")
    plt.show()
    # Print significant features
significant_features = [key for key, (chi2_stat, p_value, dof) in chi_square_results.items()
if p_value < 0.05]
print("\nSignificant Features (Chi-Square Test):", significant_features)

# ANOVA Test
categorical_feature = 'cp'
continuous_feature = 'thalach'
groups = [data[continuous_feature][data[categorical_feature] == category] for category in
data[categorical_feature].unique()]
anova_result = f_oneway(*groups)
df_between = len(groups) - 1
df_within = data.shape[0] - len(groups)
critical_value = f.ppf(0.95, df_between, df_within)

print(f"ANOVA Test Results for {continuous_feature} by {categorical_feature}:")
print(f"F-statistic: {anova_result.statistic:.4f}, Critical Value: {critical_value:.4f}, P-
value: {anova_result.pvalue:.4e}\n")

sns.boxplot(x=data[categorical_feature], y=data[continuous_feature])
plt.title(f"ANOVA Test: {continuous_feature} vs {categorical_feature}")
plt.show()

# F-Test for Variance
var_group1 = np.var(groups[0], ddof=1)
var_group2 = np.var(groups[1], ddof=1)
f_stat = var_group1 / var_group2
df1 = len(groups[0]) - 1
df2 = len(groups[1]) - 1
critical_value = f.ppf(0.95, df1, df2)
```

```
# F-Test Visualization
plt.figure(figsize=(8, 5))
sns.histplot(np.random.f(df1, df2, 1000), bins=30, kde=True, color='purple')
plt.axvline(f_stat, color='red', linestyle='--', label='F-Statistic')
plt.axvline(critical_value, color='green', linestyle='--', label='Critical Value')
plt.title('F-Test Distribution')
plt.legend()
plt.show()

# PCA Analysis
pca = PCA(n_components=0.95)
pca_data = pca.fit_transform(data[continuous_features])
pca_variance = pca.explained_variance_ratio_

print(f"PCA Reduced Dimensions: {pca_data.shape[1]} components\n")

plt.plot(range(1, len(pca_variance) + 1), np.cumsum(pca_variance), marker='o')
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("PCA Explained Variance")
plt.show()

# Step 7: Feature Importance using Decision Tree
X = data.drop('target', axis=1)
y = data['target']
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X, y)
feature_importances = pd.Series(tree.feature_importances_,
index=X.columns).sort_values(ascending=False)

# Step 8: Final Feature Selection
final_features = list(set(significant_features + feature_importances.index[:5].tolist()))
print("Final Selected Features:", final_features)

# Normalize Data using MinMaxScaler
minmax_scaler = MinMaxScaler()
data[continuous_features] = minmax_scaler.fit_transform(data[continuous_features])

print("Preprocessing Completed.")

# Load and preprocess dataset (Replace X, y with your actual dataset)
# X, y = some_preprocessed_dataframe.drop(columns=['target']),
some_preprocessed_dataframe['target']

# Train-test split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)

# Apply SMOTE before cross-validation
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_resampled)
X_test_scaled = scaler.transform(X_test)

# Define cross-validation strategy (10-fold for better generalization)
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

ensemble_accuracies = []
y_true_all = []
y_pred_all = []

# Define ANN Model
def create_ann_model(learning_rate=0.001, neurons=64, dropout_rate=0.3):
    model = Sequential([
        Dense(neurons, activation='relu', input_shape=(X_train_scaled.shape[1],)),
        BatchNormalization(),
        Dropout(dropout_rate),
        Dense(neurons // 2, activation='relu'),
        BatchNormalization(),
        Dropout(dropout_rate / 2),
        Dense(neurons // 4, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer=Adam(learning_rate=learning_rate),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

# Hyperparameter tuning for KNN using GridSearchCV
knn_param_grid = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}
knn_grid = GridSearchCV(KNeighborsClassifier(), knn_param_grid, cv=5,
scoring='accuracy')
knn_grid.fit(X_train_scaled, y_train_resampled)
best_knn = knn_grid.best_estimator_
```

```
# Manually tuned ANN hyperparameters
best_ann_params = {'learning_rate': 0.001, 'neurons': 128, 'dropout_rate': 0.3, 'epochs':
100, 'batch_size': 16}

for train_index, val_index in kfold.split(X_train_scaled, y_train_resampled):
   # Split into training and validation sets
   X_train_fold, X_val_fold = X_train_scaled[train_index], X_train_scaled[val_index]
   y_train_fold, y_val_fold = y_train_resampled.iloc[train_index],
y_train_resampled.iloc[val_index]

   # Apply SMOTE again to the training fold
   smote = SMOTE(random_state=42)
   X_train_fold_resampled, y_train_fold_resampled = smote.fit_resample(X_train_fold,
y_train_fold)

   # Train ANN with best parameters
   ann_model = create_ann_model(learning_rate=best_ann_params['learning_rate'],
                        neurons=best_ann_params['neurons'],
                        dropout_rate=best_ann_params['dropout_rate'])

   early_stopping = EarlyStopping(monitor='loss', patience=10,
restore_best_weights=True)
   model_checkpoint = ModelCheckpoint("best_ann_model.h5", save_best_only=True)

   ann_model.fit(X_train_fold_resampled, y_train_fold_resampled,
            epochs=best_ann_params['epochs'],
batch_size=best_ann_params['batch_size'],
            verbose=0, callbacks=[early_stopping, model_checkpoint])
   y_pred_ann_prob = ann_model.predict(X_val_fold).flatten()

   # Train KNN with best parameters
   knn_model = best_knn
   knn_model.fit(X_train_fold_resampled, y_train_fold_resampled)
   y_pred_knn_prob = knn_model.predict_proba(X_val_fold)[:, 1]

   # Use Logistic Regression for Ensemble Learning
   meta_model = LogisticRegressionCV(cv=5, max_iter=1000)
   meta_features = np.column_stack((y_pred_ann_prob, y_pred_knn_prob))
   meta_model.fit(meta_features, y_val_fold)

   y_pred_ensemble = meta_model.predict(meta_features)

   # Store results for classification report
   y_true_all.extend(y_val_fold)
   y_pred_all.extend(y_pred_ensemble)
```

```
    # Compute accuracy for this fold
    ensemble_accuracy = accuracy_score(y_val_fold, y_pred_ensemble)
    ensemble_accuracies.append(ensemble_accuracy)

# Compute mean accuracy
mean_ensemble_accuracy = np.mean(ensemble_accuracies) * 100
print(f"\nMean Ensemble Model Accuracy: {mean_ensemble_accuracy:.2f}%")

# Generate final classification report
print("\nClassification Report:")
print(classification_report(y_true_all, y_pred_all, target_names=['Good (0)', 'Bad (1)']))

# Compute confusion matrix
conf_matrix = confusion_matrix(y_true_all, y_pred_all)

# Extract TP, TN, FP, FN
TP = conf_matrix[1, 1]
TN = conf_matrix[0, 0]
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]

print("\nConfusion Matrix:")
print(conf_matrix)
print(f"\nTP: {TP}, TN: {TN}, FP: {FP}, FN: {FN}\n")

# Heatmap for Confusion Matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Good (0)',
'Bad (1)'], yticklabels=['Good (0)', 'Bad (1)'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Optimized ANN + KNN Ensemble Model')
plt.show()
```

Appendix 5: Comparison Graphs

```
import matplotlib.pyplot as plt

models = ["KNN", "ANN", "XRSS", "SALKS"]
accuracy = [77.05, 81.97, 81.30, 87.02]

plt.figure(figsize=(8, 5))
plt.plot(models, accuracy, marker='o', linestyle='-', color='b', label="Accuracy")

# Add labels to each point
for i, model in enumerate(models):
   plt.text(model, accuracy[i] + 0.5, f"{accuracy[i]:.2f}%", ha='center', fontsize=10, 
color='b')

plt.xlabel("Models")
plt.ylabel("Accuracy (%)")
plt.title("Model Accuracy Comparison")
plt.grid(True)
plt.legend()
plt.show()

models = ["KNN", "ANN", "XRSS", "SALKS"]
precision = [77, 84, 83, 88]

plt.figure(figsize=(8, 5))
plt.plot(models, precision, marker='o', linestyle='-', color='g', label="Precision")

# Add labels to each point
for i, model in enumerate(models):
   plt.text(model, precision[i] + 0.5, f"{precision[i]:.2f}%", ha='center', fontsize=10, 
color='g')

plt.xlabel("Models")
plt.ylabel("Precision (%)")
plt.title("Model Precision Comparison")
plt.grid(True)
plt.legend()
plt.show()

models = ["KNN", "ANN", "XRSS", "SALKS"]
recall = [71, 75, 79, 86]

plt.figure(figsize=(8, 5))
plt.plot(models, recall, marker='o', linestyle='-', color='r', label="Recall")
```

```
# Add labels to each point
for i, model in enumerate(models):
    plt.text(model, recall[i] + 0.5, f"{recall[i]:.2f}%", ha='center', fontsize=10, color='r')

plt.xlabel("Models")
plt.ylabel("Recall (%)")
plt.title("Model Recall Comparison")
plt.grid(True)
plt.legend()
plt.show()

models = ["KNN", "ANN", "XRSS", "SALKS"]
f1_score = [74, 79, 81, 87]

plt.figure(figsize=(8, 5))
plt.plot(models, f1_score, marker='o', linestyle='-', color='purple', label="F1 Score")

# Add labels to each point
for i, model in enumerate(models):
    plt.text(model, f1_score[i] + 0.5, f"{f1_score[i]:.2f}%", ha='center', fontsize=10,
color='purple')

plt.xlabel("Models")
plt.ylabel("F1 Score (%)")
plt.title("Model F1 Score Comparison")
plt.grid(True)
plt.legend()
plt.show()
```

Appendix 6: Chatbot

```
! pip install -q kaggle
from google.colab import files
files.upload()
! mkdir /kaggle
! cp kaggle.json /kaggle
! chmod 600 /kaggle/kaggle.json
! kaggle datasets list
! kaggle datasets download -d johnsmith88/heart-disease-dataset
! unzip heart-disease-dataset.zip
import pandas as pd
import numpy as np
import joblib
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

def load_data():
    # Load dataset (replace with actual dataset path)
    df = pd.read_csv("heart.csv")

    # Selecting 9 features for better predictions
    X = df[['age', 'sex', 'trestbps', 'chol', 'cp', 'thalach', 'fbs', 'restecg', 'exang']]
    y = df['target']  # Target variable (1: Disease, 0: No Disease)

    return train_test_split(X, y, test_size=0.2, random_state=42)

def train_model(X_train, X_test, y_train, y_test):
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    # Save model
    joblib.dump(model, "heart_disease_model.pkl")
    return model

def load_model():
    return joblib.load("heart_disease_model.pkl")

def categorize_blood_pressure(bp):
    if bp < 90:
        return "Low"
    elif 90 <= bp <= 120:
        return "Normal"
    else:
        return "High"
```

```
def categorize_cholesterol(chol):
    if chol < 200:
        return "Normal"
    elif 200 <= chol <= 240:
        return "Borderline High"
    else:
        return "High"

def get_user_input():
    print("\n--- Enter Your Health Details ---")
    age = int(input("Age: "))
    sex = int(input("Sex (1=Male, 0=Female): "))

    # Display proper limits near inputs
    trestbps = int(input("Resting Blood Pressure (Low <90, Normal 90-120, High >120):
"))
    chol = int(input("Cholesterol Level (Normal <200, Borderline High 200-240, High
>240): "))

    cp = int(input("Chest Pain Type (0=None, 1=Mild, 2=Moderate, 3=Severe): "))
    thalach = int(input("Max Heart Rate Achieved (Lower is riskier): "))

    fbs = int(input("Fasting Blood Sugar (>120 mg/dl is risky) (1=Yes, 0=No): "))
    restecg = int(input("Resting ECG Result (0=Normal, 1=ST-T wave abnormality,
2=Possible LVH): "))
    exang = int(input("Exercise-Induced Angina (1=Yes, 0=No): "))

    bp_category = categorize_blood_pressure(trestbps)
    chol_category = categorize_cholesterol(chol)

    return np.array([[age, sex, trestbps, chol, cp, thalach, fbs, restecg, exang]]),
bp_category, chol_category, trestbps, chol, cp, exang

def provide_recommendations(risk_level, bp_category, chol_category):
    recommendations = {
        "High": [
            "⚠️ You are at **high risk**! Consult a doctor immediately.",
            "🥗 Eat a diet low in saturated fats and high in fiber.",
            "🏃 Exercise at least **30 minutes daily**.",
            "🧘 Reduce stress through meditation or yoga.",
            "🚭 Quit smoking and limit alcohol intake.",
            "💉 Regular checkups for blood pressure, cholesterol, and diabetes."
        ],
        "No Risk, Healthy": [
            "✅ You are **healthy**! Keep up the good lifestyle.",
            "💪 Exercise regularly and maintain a **balanced diet**.",
            "🩺 Occasionally monitor blood pressure and cholesterol.",
            " 🚶 Stay active to **prevent future heart risks**."
        ]
    }

    print("\n--- Health Analysis ---")
    print(f" 🩸 Blood Pressure Level: {bp_category}")
    print(f"🧪 Cholesterol Level: {chol_category}")

    print("\n--- Risk Prediction ---")
    print(f" 💡 Risk Level: {risk_level}")

    print("\n--- Recommended Health Tips ---")
    for tip in recommendations[risk_level]:
        print(f"- {tip}")

def chatbot():
    print("Welcome to the AI-Driven Heart Disease Chatbot!")
    model = load_model()

    user_input, bp_category, chol_category, trestbps, chol, cp, exang = get_user_input()

    # Manual override for high-risk conditions
    if trestbps > 120 or chol > 240 or cp >= 2 or exang == 1:
        risk_level = "High"
    else:
        prediction = model.predict(user_input)[0]
        risk_level = "High" if prediction == 1 else "No Risk, Healthy"

    provide_recommendations(risk_level, bp_category, chol_category)

if __name__ == "__main__":
    X_train, X_test, y_train, y_test = load_data()
    train_model(X_train, X_test, y_train, y_test)
    chatbot()
```