Mini-Project Report On

CardioCare : ML Powered Heart Disease Risk Prediction and Doctor Recommendation

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

Bachelor of Technology

in

Computer Science & Engineering

By

Fathima Sahliya K S (U2003079)
Iva Sony (U2003098)
Jiya Joy (U2003102)
Khadeeja C R (U2003119)

Under the guidance of
Ms. Seema Safar

RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

This is to certify that the mini-project report entitled ***"CardioCare : ML Powered Heart Disease Risk Prediction and Doctor Recommendation"*** *is a bonafide work done by* ***Ms. Fathima Sahliya KS (U2003079), Ms. Iva Sony (U2003098), Ms. Jiya Joy (U2003102), Ms. Khadeeja C R (U2003119)****, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.	**Mr. Uday Babu P.**	**Ms. Seema Safar**
Head of Department	Mini-Project Coordinator	Mini-Project Guide
Dept. of CSE	Asst. Professor	Asst. Professor
RSET	Dept. of CSE	Dept. of CSE
	RSET	RSET

ACKNOWLEDGEMENTS

We wish to express our sincere gratitude towards **Dr. P. S. Sreejith**, Principal of RSET, and **Dr. Preetha K. G.**, Head of Department of Computer Science and Engineering for providing us with the opportunity to undertake our mini-project, "CardioCare : ML Powered Heart Disease Risk Prediction and Doctor Recommendation".

We are highly indebted to our mini-project coordinators, **Mr. Uday Babu P.**, Assistant Professor, Department of Computer Science and Engineering and **Ms. Tripti C**, Assistant Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our mini-project guide **Ms. Seema Safar**, for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Fathima Sahliya K S

Iva Sony

Jiya Joy

Khadeeja C R

ABSTRACT

Cardiovascular diseases (CVDs) are a major public health concern and the leading cause of death worldwide, with heart diseases being the most prevalent form. Early prediction and finding the right doctor are considered effective measures for controlling CVDs. This project aims to develop a Doctor Recommender Engine to facilitate early detection and provide personalized choices to users seeking medical advice or treatment for cardiovascular issues.The risk prediction of heart disease on patients(men and women) and to recommend a doctor specifically for those patients with the aid of Machine Learning (ML) models is the main purpose of our project.

For risk prediction we did a comparative study of models like Naive Bayes,SVM(Support Vector Machine),Simple ANN(Artificial Neural Network) in terms of accuracy,precision, F1 score.The performance analysis demonstrated that Simple ANN(Artificial Neural Network) is the most appropriate model against Naive Bayes and Support Vector Machine (SVM) with 97.5% accuracy.Doctor recommendation engine is implemented by utilizing KNN algorithm(K Nearest Neighbour) and collaborative filtering technique using SVD(Single Value Decomposition).This system will be used to recommend top doctors based on the current condition and characteristics of patients.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

1.1.1 The Risk of Heart Disease

According to the World Health Organization, every year 12 million deaths occur worldwide due to Heart Disease. The load of cardiovascular disease is rapidly increasing all over the world from the past few years. Many researches have been conducted in attempt to pinpoint the most influential factors of heart disease as well as accurately predict the overall risk.Heart disease encompasses a range of conditions, including coronary artery disease, heart attacks, and heart failure. Lifestyle factors, genetic predispositions, and underlying health conditions contribute to the risk of developing heart disease. Heart Disease is even highlighted as a silent killer which leads to the death of the person without obvious symptoms. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications.However, delays in diagnosis and treatment can lead to serious consequences and complications, highlighting the need for effective prediction models and prompt medical attention. It is very hard or even impossible for humans to derive useful information from these massive amounts of data, that is why machine learning is widely used nowadays to analyze these data and diagnose problems in the healthcare field. A simplified explanation of what the machine learning algorithms would do is, it will learn from previously diagnosed cases of patients. The resulting classifier can have many uses such as helping doctors to diagnose new patients with higher speed and efficiency and training students and non-specialists to diagnose patients [4]

Delaying the diagnosis and treatment of heart disease can have severe implications for individuals and the healthcare system. Prolonged delays can result in the progression

of the disease, leading to irreversible damage to the heart muscle, increased morbidity, and even mortality. Furthermore, delayed treatment can contribute to the exacerbation of symptoms, decreased quality of life, and higher healthcare costs associated with managing advanced stages of heart disease.

1.1.2 Doctor recommendation

Access to appropriate healthcare professionals is crucial for effective diagnosis, treatment, and management of medical conditions. However, many individuals face difficulties in finding doctors who meet their specific needs and preferences. Major difficulties are :

- One of the primary difficulties in finding doctors is the limited availability of comprehensive and reliable information. Patients often struggle to access detailed profiles, including specialties, qualifications, experience, and patient reviews. This lack of information makes it challenging to assess the suitability of doctors and make informed decisions.

- Healthcare systems often rely on general practitioners or primary care physicians for initial consultations. However, patients with complex or specialized medical conditions may require referrals to specialists. Obtaining timely referrals and accessing specialized care can be challenging, leading to delays in diagnosis and treatment.

- Finding a doctor who aligns with a patient's preferences, values and interests is crucial for establishing a strong doctor-patient relationship. However, the existing systems may not prioritize these factors, leading to suboptimal matches and decreased patient satisfaction.

Addressing these challenges through enhanced information availability and personalized doctor-patient matching based on the patients health condition and rating of doctors can contribute to better patient experiences and improved healthcare outcomes.

1.2 Existing System

The traditional approach for risk prediction can be time-consuming, and may lead to delays in receiving appropriate medical treatment..The existing system may offer doctor

recommendations based on general patterns and preferences among users but may not fully account for individual patient needs and preferences.

1.3 Problem Statement

The traditional diagnosis approach entails a patient visiting a doctor, undergoing many medical tests, and then reaching a consensus. This process is very time-consuming. Also doctor recommendations are often based on general knowledge and referrals rather than a personalized approach.

To avoid this we are developing a system that detects the likelihood of heart disease in an individual using simple artificial neural network and recommend doctors based on the preferences and ratings by the patients using K-nearest neighbour and collaborative filtering .

1.4 Objectives

- **Risk prediction of heart disease**- By considering 13 attributes of patients like age,sex,restbp,cholestrol,etc..we predict the risk of heart disease using Simple ANN (Artificial Neural Network)
- **Doctor Recommendation Engine** - Considering predicted risk and some of the attributes of patient we recommend top 5 doctors based on patient ratings and the current condition of patient.

1.5 Scope

The scope of the project includes the development and implementation of a comprehensive cardiovascular disease risk prediction and doctor recommendation system. The project aims to leverage predictive models, such as advanced Artificial Neural Networks (ANNs), to accurately assess a patient's risk of developing heart disease. Additionally, it seeks to incorporate a doctor recommendation engine, utilizing collaborative filtering to suggest top doctors based on patients' specific cardiovascular conditions and individual preferences.

The scope also includes the development of user-friendly interfaces for patients and enabling seamless interaction with the system. The user interface will allow patients to input relevant data, receive personalized risk assessments, and access doctor recommendations.The project's evaluation will involve rigorous performance analysis, comparing the proposed advanced ANN model with traditional models like Naive Bayes and Support Vector Machine (SVM) in terms of accuracy, precision, and F1 score.

The scope extends to conducting real-world tests and validation to ensure the system's accuracy and generalizability in different healthcare settings. The project's success will be measured by the system's ability to provide accurate risk predictions, personalized doctor recommendations, and its impact on improving patient outcomes and reducing the burden of cardiovascular diseases. Throughout the project, a strong emphasis will be placed on adhering to best practices in machine learning, data integration, and user experience design. The proposed system's scalability and adaptability will be considered to facilitate potential future implementations in various healthcare environments, expanding its reach and effectiveness in addressing the global challenge of cardiovascular diseases.

Chapter 2

Literature Review

2.1 Enhanced Heart Disease Prediction Based on Machine Learning and 2 Statistical Optimal Feature Selection Model

This paper [1] proposes a new heart disease classification model based on the support vector machine (SVM) algorithm for improved heart disease detection. To increase prediction accuracy, the 2 statistical optimum feature selection technique was used. The suggested model's performance was then validated by comparing it to traditional models using several performance measures. The proposed model increased accuracy from 85.29% to 89.7%. Additionally, the componential load was reduced by half. In the proposed support vector machine (SVM)-based heart disease prediction system, the most significant stages were data pre-processing, feature selection, and classifying. The feature normalization method was included in the pre-processing block. Training and testing sets were then created from the data. A feature scoring and selection algorithm was employed to ensure that the training subset was free of any biases. The 2 statistical model selected the same feature set for training and testing data. Next, training data with fewer features was fed into the SVM model for training purposes. Finally, using testing data, the trained SVM model was evaluated. The proposed model utilized 14 features from the University of California Irvine (UCI) Heart Disease Repository's Statlog and Cleveland dataset. These features were examined using approaches that successfully predict heart disease. They developed and evaluate the system for heart disease prediction by using Python and the sci-kit learn library.

2.2 Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm

In this paper [2] KNN collaborative filtering algorithm, which is a collaborative filtering algorithm combined with KNN algorithm, use KNN algorithm to select neighbors. The basic steps of the algorithm are user similarity calculation, KNN nearest neighbor selection and predict score calculation.The similarity between users is calculated by evaluating the value of the items evaluated by two users. Each user uses N dimension vector to represent item score, for example, to calculate of similarity of U1 and U3, first find out the set of films that they all scored as m1, M2, M4, m5 and relative scores of these films. The score vector of U1 is 1,3,4,2, and the score vector of U3 is 2,4,1,5. The similarity of U1 and U3 is calculated by the similarity formula.The similarity of u and u' is denoted as (u,u') , the commonly used method of calculating user similarity are Cosine Similarity and Pearson Correlation similarity. In this paper [2] they calculates the similarity between two users by calculating the cosine of the angle between the two vectors.After the calculation of similarity as (u,u') between users, the algorithm selects a number of users the highest similarity as the U's neighbor, denoted as u'. Set a fixed value K for the neighbor selection, select only the most K high similarity as neighbors regardless of the value of the neighbor similarity of users.After determining the user's neighbors, the score can be predicted according to the score of the neighbor to the item.

To sum up, the process of calculating prediction score of user u for i is as follows: Step1. Generate user - item two-dimensional matrix of score as Rmxn, where each score is 2,3. Step2: Use principle of cosine similarity or Pearson correlation similarity to calculate the similarity between each 2 users as (u,u'), and generate the user similarity matrix. Step3: according to the results obtained by Step2, find K number of score which has the maximum weight, the corresponding K users is the neighbors of u. Step4: Use formula 3 to calculate the predictive value of i for target user u. In this way, we can calculate the prediction score of the target users for the non-scored movies, and the N movies with the highest score can be recommended to the user. In this paper, KNN collaborative filtering algorithm based on user is used to implement the recommendation of movie and the collaborative filtering algorithm based on the project is used to implement the recommendation of the associated movie. In addition, it can also recommend the movies to new users according to

user registration information, and it can make new and unpopular movie recommendation according to the film's browsing and score.

R. Perumal et al. [5] developed a heart disease prediction model using the Cleveland dataset of 303 data instances through feature standardization and feature reduction using PCA, where they identified and utilized seven principal components to train the ML classifiers. They concluded that LR and SVM provided almost similar accuracy values (87% and 85%, respectively) compared to that of k-NN with 69%. C. B. C. Latha et al. [5] performed a comparative analysis to improve the predictive accuracy of heart disease risk using ensemble techniques on the Cleveland dataset of 303 observations. They applied the brute force method to obtain all possible attribute set combinations and trained the classifiers. They achieved a maximum increase in the accuracy of a weak classifier of 7.26% based on ensemble algorithm, and produced an accuracy of 85.48% using majority vote with NB, BN, RF, and MLP classifiers using an attribute set of nine attributes. D. Ananey-Obiri et al. [7] developed three classification models, namely, LR, DT, and Gaussian naïve Bayes (GNB), for heart disease prediction based on the Cleveland dataset. Feature reduction was performed using single value decomposition, which reduced the features from 13 to 4. They concluded that both LR and GNB had predictive scores of 82.75% and AUC of 0.87. It was suggested that other models, such as SVM, k-NN, and random forest, be included.

2.3 DeepReco: Deep Learning Based Health Recommender System Using Collaborative Filtering

This paper [3] gives a proposed intelligent HRS using Restricted Boltzmann Machine (RBM)-Convolutional Neural Network (CNN) deep learning method, which provides an insight into how big data analytics can be used for the implementation of an effective health recommender engine, and illustrates an opportunity for the health care industry to transition from a traditional scenario to a more personalized paradigm in a tele-health environment. By considering Root Square Mean Error (RSME) and Mean Absolute Error (MAE) values, the proposed deep learning method (RBM-CNN) presents fewer errors compared to other approaches.

They analyze different privacy-preserving based recommender systems, e.g., the ma-

trix factorization model, SVD, etc. by which the system can generate recommendations without actually seeing the patient ratings.Matrix Factorization These are the most successful latent models which assist in solving high sparsity problems. In its primary form, the characteristics of both items and patients by an array of factors are derived from patient ratings patterns. These techniques have become favorable due to their better scalability and predictive accuracy. It is an influential technique to uncover the hidden structure behind data. It is used for processing large databases and providing scalability solutions.

Comparison

From the proposed method in [1] we implemented the Risk prediction using Machine learning by SVM (Support vector machine) classifier and additionally tried Naive Bayes and analyzed the performance.Moreover we tried a different model from the proposed method in [1] namely simple ANN(artificial neural network). From [2] we used KNN (k nearest neighbours) to find similarity between patients and SVD [3] to predict rating of doctors.

Chapter 3

System Analysis

3.1 Expected System Requirements

The system of user which is a PC is expected to have the following features:

- Operating System Compatibility:The system must be compatible with the operating system
- A minimum Ram size of 4GB is required in the device.

3.2 Feasibility Analysis

3.2.1 Technical Feasibility

The compatibility of the system with different operating systems (e.g., Windows, mac OS,) is generally feasible. Developing applications for multiple operating systems is common practice, and cross-platform frameworks and development tools are available to simplify this process.Requiring a minimum RAM size of 4GB is reasonable, as modern PCs often come with higher RAM capacities. The application can efficiently run on systems with 4GB or more RAM without significant performance issues.

3.2.2 Operational Feasibility

The operations are built in a simple and easy to use for heart disease patients. The system can be seamlessly integrated with existing healthcare infrastructure, and other relevant systems.

3.2.3 Economic Feasibility

The risk prediction of heart disease and doctor recommendation system is economically feasible as the development of the system is zero budget as it was built using free resources.

3.3 Hardware Requirements

The following are the system requirements to develop the System for Risk prediction and doctor recommendation specifically for heart disease patients .

- Processor: Intel Core i5
- RAM: Minimum 8GB

3.4 Software Requirements

The following are the softwares used in the development of the System for Risk prediction and doctor recommendation specifically for heart disease patients .

Operating System: Windows

3.4.1 Python 3.10

Python 3.10 is a major release of the Python programming language, introducing new features, optimizations, and improvements to the language. It was officially released on October 4, 2021. This version builds upon the strengths of its predecessor, Python 3.9, while bringing enhancements that aim to make Python development more efficient and enjoyable.Python 3.10 introduces new syntax warnings to help developers transition to future language changes smoothly. It allows developers to identify and update their code to ensure compatibility with upcoming Python versions.Pattern matching is a powerful new feature that allows developers to match data structures and extract information from them in a concise and readable manner.It simplifies the process of handling complex data structures such as lists, dictionaries, and custom objects, improving code readability and reducing the need for nested conditionals.

3.4.2 Tensorflow

TensorFlow is an open-source machine learning library developed by Google that is widely used for building and training various machine learning and deep learning models. It provides a flexible and efficient ecosystem for developing artificial intelligence applications, especially neural network-based models. TensorFlow is designed to handle both numerical computations and large-scale machine learning tasks, making it a popular choice

for researchers and developers. TensorFlow is an essential tool for implementing machine learning models in the heart disease prediction and doctor recommendation project. It allows developers to build and train predictive models using various algorithms and efficiently deploy them in the web application for real-time prediction and recommendation tasks. With TensorFlow's versatility and scalability, the project can leverage its capabilities to deliver accurate predictions and recommendations to users, enhancing the overall system's performance and user experience.

3.4.3 Scikit

Scikit-learn, commonly referred to as Scikit, is a popular open-source machine learning library for Python. It provides a wide range of tools and algorithms for data mining, data analysis, and machine learning tasks. Scikit-learn is built on top of other Python libraries, including NumPy, SciPy, and matplotlib, and is designed to be easy to use and efficient.Scikit-learn offers a diverse set of machine learning algorithms, including supervised and unsupervised learning algorithms, as well as tools for model selection and evaluation.scikit-learn provides a consistent and intuitive API, making it easy for developers to implement machine learning models and pipelines.

The library offers a wide range of data preprocessing functionalities, such as data scaling, normalization, feature extraction, and handling missing values.Scikit-learn provides tools for model selection, hyperparameter tuning, and evaluation, helping developers choose the best-performing models for their specific tasks.The library includes a comprehensive set of performance metrics for classification, regression, and clustering tasks, allowing users to evaluate the quality of their models.Scikit-learn supports various dimensionality reduction techniques, such as Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE). Scikit-learn is widely used in academia and the industry for various machine learning tasks, including classification, regression, clustering, and dimensionality reduction

Chapter 4

Methodology

4.1 Proposed Method

- Develop a system that predicts the risk of heart disease and recommends doctors.
- We plan to build risk prediction model using simple Artificial Neural Network, and the recommendation engine using k-Nearest Neighbors (KNN) and Singular Value Decomposition (SVD) matrix factorization
- User gives patient attributes(like age, sex, cholesterol, etc) as input and the prediction model predicts his/her risk of heart disease on a scale of 0-4. Then if the user wishes, the recommendation engine recommends doctors, based on the ratings given by similar patients.
- In the GUI user can see his/her last predicted risk of heart disease in percentage.

4.1.1 Risk Prediction Model

The risk Prediction Model uses a simple artificial neural network. We tried Naive Bayes and Support Vector Machine(SVM) models first. For Naive Bayes we got an accuracy score of 70.49, F1 score of 0.705, and a precision of 0.709. For SVM we got an accuracy of 82.78, with F1 score of 0.81 and a precision of 0.82. Finally, we tried ANN and got an accuracy of 97.5, an F1 score of 0.98, and a precision of 0.99. The user inputs 13 attributes(13 features that can explain the medical condition of a heart) which include age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting ECG, max heart rate, exercise-induced angina, ST depression, the slope of ST segment, smoke, and Thalassemia type. The code creates a sequential model using Sequential() from Keras. It then adds three layers to the model:

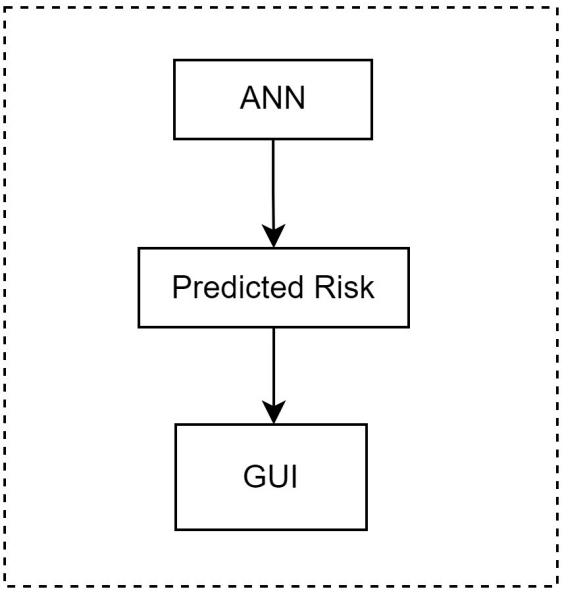


Figure 4.1: Risk Prediction Model

- The first hidden layer (Dense) consists of 13 neurons with a ReLU activation function and an input dimension equal to the number of features in the training data.
- The second hidden layer (Dense) consists of 30 neurons with a ReLU activation function.
- The output layer (Dense) consists of 5 neurons (equal to the number of classes) with a softmax activation function. Softmax is used to convert the raw model outputs into probabilities, allowing for multi-class classification.

The model is compiled using the Adam optimizer, categorical cross-entropy loss (appropriate for multi-class classification), and accuracy as the evaluation metric. The target variables y_train and y_test are one-hot encoded to_categorical from Keras. One-hot encoding converts the class labels into binary vectors, which are suitable for multi-class classification.The model is trained using the fit function with the one-hot encoded target variables. The training data (x_train and y_train_encoded) are used for training, and the validation data (x_test and y_test_encoded) are used for validation during training. The training runs for 175 epochs with a batch size of 8.

Artificial Neural Network(ANN)

Artificial Neural Network (ANN) is a powerful machine learning model for predicting the risk of heart disease. The input data, which includes patient attributes like age, sex, cholesterol levels, etc., is fed into the input layer of the ANN.

The input data is multiplied by the weights and passed through the ReLU activation function in the hidden layer(s) to introduce non-linearity and learn complex patterns.The ReLU function is defined as ReLU(x) = max(0, x), which means if the weighted sum of inputs (z_j) is greater than or equal to zero, the output (a_j) is equal to z_j; otherwise, it is zero.The ReLU activation function is computationally efficient and helps mitigate the vanishing gradient problem, making it suitable for deep networks.

The output of the hidden layer(s) is then multiplied by the weights and passed through the Softmax activation function in the output layer to obtain the predicted risk probabilities for each class (e.g., low risk, medium risk, high risk).The class with the highest probability is considered the predicted risk of heart disease for the patient.It converts the output values into probability-like values, ensuring they sum up to 1, representing the probability distribution over different classes.The Softmax function takes the weighted sum of inputs (z_out) from the output layer and calculates the probability of each class.Mathematically, the Softmax function for a single output neuron is defined as:

Softmax(z_{out}) = $\frac{e^{z_{out}}}{\sum_i e^{z_i}}$ for all output neurons i

Here, exp() is the exponential function, and z_i represents the weighted sum of inputs for the i-th output neuron.The Softmax activation function is particularly useful in multi-class classification tasks, as it allows the ANN to output class probabilities, facilitating confident predictions and decision-making.

The training process involves adjusting the weights and biases of the ANN to minimize prediction errors during the forward pass.The Adam optimizer is used as an optimization algorithm in the training process. It combines the benefits of both the Adaptive Moment Estimation (Adam) and Root Mean Square Propagation (RMSprop) algorithms.Adam optimizes the learning rate for each weight and bias, adapting it dynamically based on the past gradients to achieve faster convergence and better performance.During training, the optimizer computes the gradients of the model's weights and biases using the back-

propagation algorithm and updates them accordingly to minimize the prediction error (loss) on the training data.

Evaluation metrics to assess the model's performance includes,

- Accuracy = $\frac{Numberofcorrectlypredictedinstances}{Totalnumberofinstances}$
- Precision = $\frac{TruePositives}{TruePositives+FalsePositives}$
- Recall = $\frac{TruePositives}{TruePositives+FalseNegatives}$
- F1 Score = $2 \times \left(\frac{Precision \times Recall}{Precision+Recall}\right)$
- RMSE = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{actual,i} - y_{predicted,i})^2}$
- $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_{actual,i} - y_{predicted,i}|$

After obtaining the predicted risk of heart disease for a patient using the trained ANN, the risk prediction output is integrated into the doctor recommendation engine.

4.1.2 Recommendation Engine

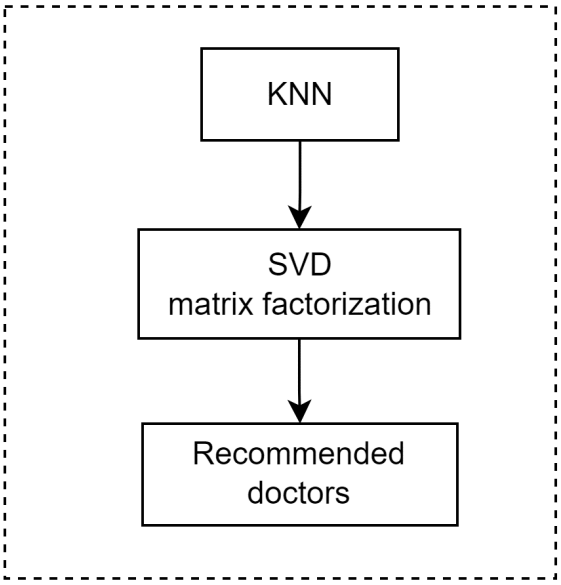


Figure 4.2: Recommendation Engine

Recommendation Engine uses a KNN (k Nearest Neighbour) algorithm to find similar users.Patient attributes like age, sex, resting blood pressure, cholesterol, and the predicted

risk are taken as input for the recommendation engine.The distance between the new user's attributes and those of similar users is calculated, and the IDs of similar users with their respective distances are obtained.SVD algorithm is then imported from the Surprise library to perform collaborative filtering.The Top 5 doctors in the respective similar user is displayed as recommended doctors for the new user.

k Nearest Neighbour

K-Nearest Neighbors (K-NN) is a popular and simple machine learning algorithm used for both classification and regression tasks. It is a type of user-based learning, where the algorithm memorizes the training dataset and uses it to make predictions for new, unseen data points.It is used to identify users with similar characteristics based on their attribute values and predicted risk.The KNN algorithm is applied in the following way to find similar users:

- Data Preparation:
  - The user data is collected and represented as a dataset with each row representing a user and their attributes (e.g., age, sex, restbp, chol, num).
  - The attributes are numerical or can be converted to numerical values for distance calculation.
- Training Phase:
  - In the training phase, the K-NN algorithm stores the user data (excluding any identifying information like user IDs) as the training dataset.
  - The training dataset becomes the reference data for finding similar users.
- Prediction Phase (Finding Similar Users):
  - When a new user (query user) with specific attribute values is given, the K-NN algorithm is applied to find the k-nearest neighbors (similar users) to the query user.
  - The algorithm calculates the distance (e.g., Euclidean distance) between the query user and all other users in the training dataset based on their attribute values.

  - The k-nearest neighbors are the users with the smallest distances to the query user.

- Output:
  - The KNN algorithm returns the k-nearest neighbors, which represent the most similar users to the query user based on their attribute values.

The distance between the new user and the other users are calculated ,the distance taken here is euclidean distance. For 2 points P=(p_1 ,p_2 ,p_3 ... p_n) and Q=(q_1 ,q_2 ,q_3 ... q_n) in an n-dimensional space, the Euclidean distance d is calculated as:

$$d^2 = \sum_{i=1}^{n} (p_i - q_i)^2$$

- where d^2 is the represents the squared Euclidean distance.
- p_i and q_i are the ith coordinates of points P and Q in a n-dimensional space, respectively.
- n is the number of dimensions in the space.

The method(kneighbors) returns two arrays: distances (the distances between the new user and its k-nearest neighbors) and indices (the indices of the k-nearest neighbors in the original DataFrame).The function then plots the attributes of the new user and its k-nearest neighbors using a scatter plot. reads user data from a CSV file, builds a K-NN model using the 'age', 'sex', 'restbp', 'chol', and 'num' attributes, and finds similar users to a new user based on these attributes.The K-NN model is created with k = 5, indicating that it will find 5 nearest neighbors.These 5 users are considered the similar users of the new user.

Singular Value Decomposition(SVD)

The Recommendation Engine implements a collaborative filtering recommendation system using the Singular Value Decomposition (SVD) algorithm. SVD is used to train a collaborative filtering recommendation model on the doctor rating data. SVD is a matrix

factorization method that decomposes a matrix into three separate matrices, which represent the underlying latent features of the data. Given a matrix A, SVD decomposes it into three matrices as follows:

$$A = U * \sum * V^T$$

where:

- A is the original matrix of size m x n.
- U: The left singular vector matrix. It represents the relationships between users and latent factors in the collaborative filtering context. Each row of U represents a user, and each column represents a latent feature.
- $\sum$: The singular value matrix. It is a diagonal matrix containing the singular values in descending order. Singular values represent the importance of each latent feature.
- V^T: The transpose of the right singular vector matrix. It represents the relationships between items (in this case, doctors) and latent factors. Each row of V^T represents an item, and each column represents a latent feature.

SVD can be used to capture latent features of users and items (doctors) from their interactions (ratings). The decomposed matrices can be used to predict unknown ratings for a user-item pair by calculating the dot product of the corresponding rows of U, $\sum$, and V^T. After training the SVD model, it is used to predict the ratings for each user-item pair in the test set. The predicted ratings are then used to find similar users. In this context, similar users refer to users who have rated similar items similarly in the past.The similar users obtained from the KNN algorithm is made to predict a rating for the doctors in the doctor dataset. After obtaining the predicted ratings for the test set, the code proceeds to generate recommendations for the target user (new user) based on the similarity with other users (similar_users).For each doctor (item) in the dataset, the SVD model predicts the rating that the similar user would give to that doctor based on their past preferences and the latent features learned during training.For each similar user, the code predicts ratings for doctors using the trained SVD model.In order to obtain a similarity measure

,the the inverse of the distance is taken.The logic behind this is , closer the similar user, higher is the similarity between them. The predicted ratings are then multiplied by the distance to the corresponding similar user, giving more weight to doctors that were recommended by closer users. The recommendations are sorted based on the predicted ratings and stored in the recommended_doctors list. After obtaining the weighted predictions for all doctors, the code sorts them in descending order based on the weighted predicted ratings. The top N (e.g., top_n = 5) doctors with the highest weighted predicted ratings are selected as the recommendations for the target user.

- The surprise library is used to handle the data and train the SVD model.
- The scikit-learn, library is used for data preprocessing and splitting the data into training and testing sets.

The algorithm aims to find a low-rank approximation of the original rating matrix, capturing the most important latent features while reducing noise and dimensionality. The optimal hyperparameter k for SVD (number of latent features) is found through cross-validation to minimize the Root Mean Squared Error (RMSE) during training. Doctor details from a separate CSV file containing doctor information (e.g., name and phone number) is retrived based on doctor IDs. Overall, by leveraging similar_users found using collaborative filtering and SVD,we generate a personalized recommendation for the target user (new user) based on the historical preferences of similar users and their interactions with doctors.

Chapter 5

System Design

5.1 Architecture Diagram

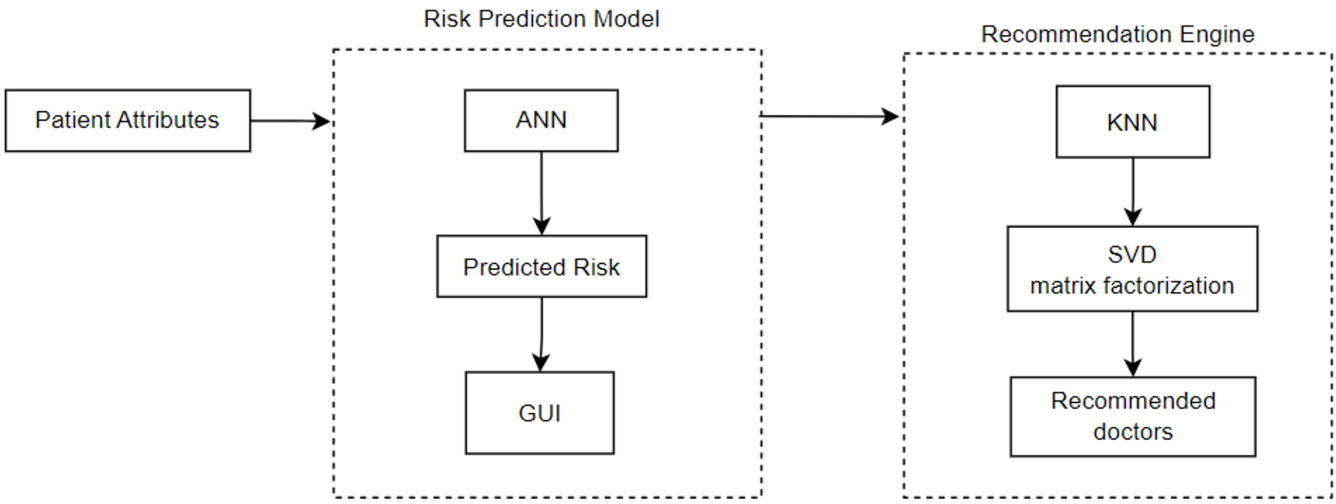


Figure 5.1: Architecture diagram

With KNN and SVD matrix factorization, similar users and ratings that can be provided by them to the doctors are found out. Finally, the doctors are recommended on the basis of predicted ratings and similarity between patients.

5.1.1 Phase 1: Risk Prediction Module

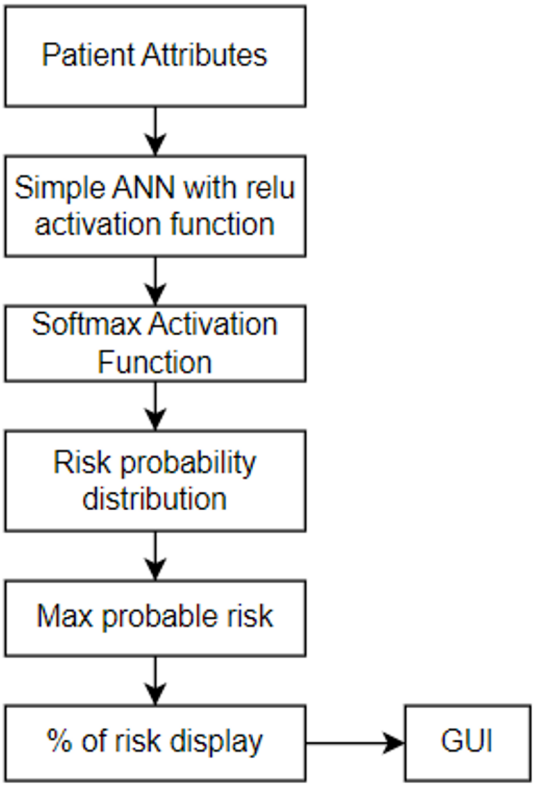


Figure 5.2: Risk Prediction Module

- Input 13 patient attributes from user
- Model used : Simple ANN with 3 layers
- Predict risk

5.1.2 Phase 2: Recommendation Engine

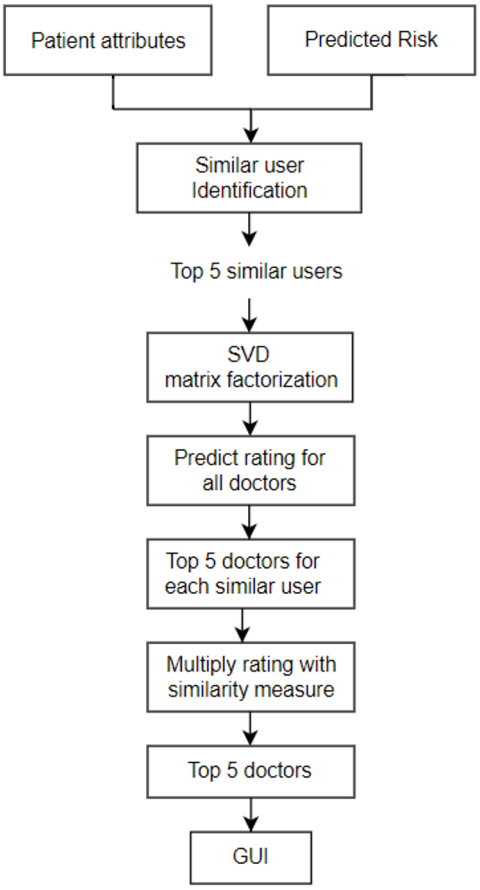


Figure 5.3: Recommendation Engine

- Use few patient attributes + "predicted risk"
- Use KNN to find similar patients
- Find similarity measure between patients
- Considering similarity measure recommend doctors using SVD

5.2 Overall Sequence Diagram

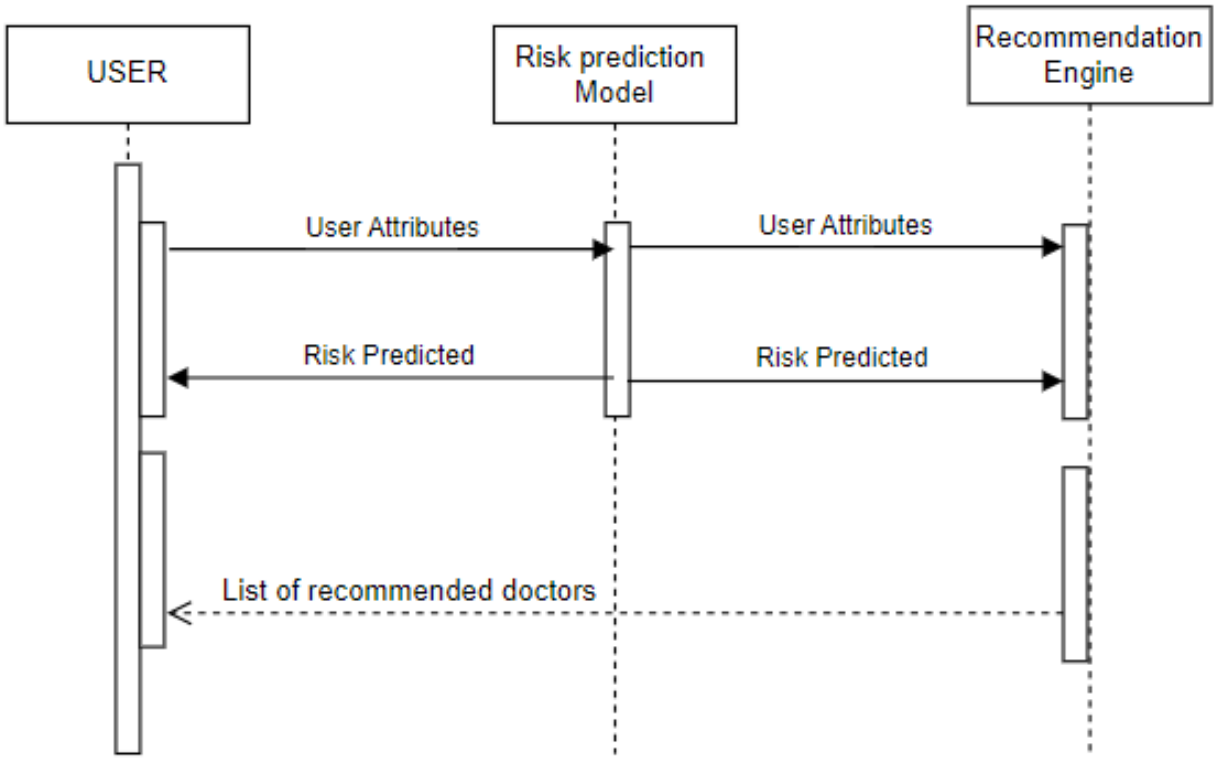


Figure 5.4: Overall Sequence Diagram

5.2.1 Risk Prediction Model

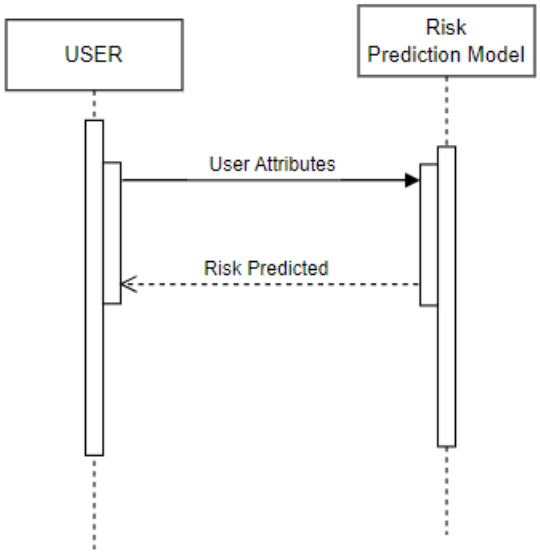


Figure 5.5: Risk Prediction Model

5.2.2 Recommendation Engine

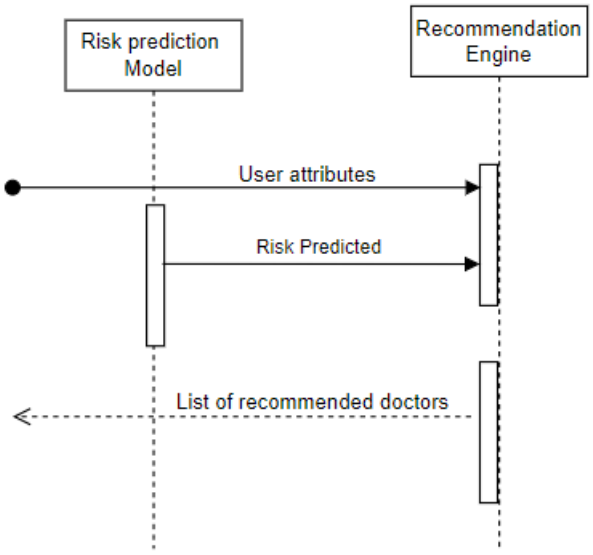


Figure 5.6: Recommendation Engine

Chapter 6

System Implementation

6.1 Risk Prediction Model

6.1.1 Model Architecture

Training a Artificial Neural Network(ANN) from scratch takes a lot of data and also compute power. Instead we use transfer learning, where a model trained on similar data is fine-tuned as per our requirement. We used a simple ANN which was more easier to implement and train. Here Artificial Neural Network (ANN) is implemented using the Keras library, which is a high-level neural networks API written in Python.

Keras provides a user-friendly and modular interface to create and train neural networks.The user enters 13 attributes ,these features explain the medical condition of the heart. They include age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting ECG, max heart rate, exercise-induced angina, ST depression, the slope of ST segment, smoke, and Thalassemia type.These inputs are encoded into numerical values based on predefined mappings.The user inputs and encoded features are combined into a list, and feature scaling is applied to it. A transformed version of the last data point (the user's input) is obtained and reshaped into a 2D array, representing the new user's features. The trained model is then used to predict the heart disease risk for the new user based on their features.

For training the algorithm, we use a Softmax and ReLU activation function. The algorithm has 3 layers ,to introduce non-linearity and learn complicated patterns, the input data is multiplied by the weights and then passed via the ReLU activation function in the hidden layer(s).With a ReLU activation function and an input dimension equal to the number of features in the training data, the first hidden layer (Dense) is made up of 13 neurons.

To acquire the estimated risk probabilities for each class, the output of the hidden layer(s) is then multiplied by the weights and fed through the Softmax activation function in the output layer.30 neurons with a ReLU activation function make up the second hidden layer (Dense), which is hidden. The output layer (Dense) has a softmax activation function and 5 neurons (equivalent to the number of classes). To enable multi-class classification, the raw model outputs are transformed into probabilities using Softmax.

When a user presses the submit button after entering the 13 attributes,the values entered is processed by the algorithm and is displayed as a percentage on the User Interface.The user has 2 options:to sign in as a new user or log in with the correct username and password. If the user chooses to add a new profile, they have to enter the username password(as per the user's choice), phone number, and location, Then click 'Submit' to redirect you to the login page. If the user chooses to log in with the correct details, the system checks the validity of the entered credentials by crosschecking the SQL database. Once access is granted the user has to enter the 13 features mentioned in the user interface.

6.1.2 Dataset

Cleveland dataset consisting of 13 attributes(age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting ECG, max heart rate, exercise-induced angina, ST depression, the slope of ST segment, smoke, and Thalassemia type and target value) and 303 rows, is used for training the prediction model used above. From the Cleveland dataset,20 percent of them were used in the testing, each of them are distinct. The remaining 80 percent was used for training. Data augmentation is performed to increase the amount of training data and improve the model's generalization. It applies adding random noise to the features on a scale factor of (-0.1,0.1) . This creates augmented data points that are small variations of the original data.Feature scaling is applied to normalize the features between 0 and 1. The StandardScaler from scikit-learn is used to scale the features.

6.2 Recommendation Engine

6.2.1 Model Architecture

Recommendation Engine uses KNN algorithm and collaborative filtering with Singular Value Decomposition (SVD).

The first step being Data Preprocessing, The function (knnmodel) reads user data from a CSV file named 'users_final - Sheet1.csv', which contains user features such as age, sex, restbp (resting blood pressure), chol (cholesterol), and num (target variable/risk of heart disease).The user features are then scaled using StandardScaler to normalize them between 0 and 1.The scaled features are stored and used later for k-Nearest Neighbors (k-NN) computation.

Next, a k-NN model is trained using the NearestNeighbors class from sklearn.neighbors. The function find_similar_users is used to find similar users to the new user. It takes the user's features (age, sex, restbp, chol, and num) as input and returns the indices of similar users in the dataset and their distances from the new user.We calculate the distance between the new user and the other users.The top 5 users with the shortest distance is considered as the similar users.The assumption we make being, shorter the distance higher the similarity. The inverse of the distance is taken as a similarity measure.The similar users are selected based on the features' proximity to the new user.

Finally, After finding similar users, the function (knnmodel) proceeds to recommend doctors for the new user. The function loads doctor rating data from a CSV file, which contains doctor ratings given by users. This data is used to build a collaborative filtering model using the SVD (Singular Value Decomposition) algorithm from surprise library.SVD is a popular and effective matrix factorization technique used in collaborative filtering recommendation systems to provide personalized recommendations to users based on their past interactions with items. The code then performs cross-validation to find the optimal random state for train-test split and initializes the SVD model with the optimal random state. The SVD model is trained on the doctor rating data. Once the collaborative filtering model is trained, the function goes through each similar user found in the previous step and predicts their ratings for different doctors. It multiplies the predicted ratings by the distances from the new user, giving more weight to the ratings

of more similar users. The recommendations are sorted based on the predicted ratings, and the top 5 unique doctors.

The user on clicking the 'Submit' button gets a predicted risk in percentage, which is displayed on to the user interface. This is the output obtained from the Risk Prediction Model. This predicted risk along with 4 user attributes are the inputs for the Recommendation Engine.Clicking the 'Next' button the user is redirected to the Recommendation page.

Here on clicking the 'Recommend Doctor' button the list of doctors are printed onto the interface along with their doctor IDs,names and phone numbers.

6.2.2 Dataset

A dataset with 50 users (user id, age, sex, resting blood pressure,cholesterol, and target value)was created as a subset of Cleveland dataset used above to find similar users.The k-neighbors method from scikit-learn's NearestNeighbors is used to find the k nearest neighbors to the new user's features. The indices and distances of similar users are obtained. A dataset with doctor ratings provided by previous users(user id, doctor id, rating) was also created. The dataset is split into training and testing sets using the train_test_split function from the Surprise library. The reader object from the Surprise library is used to define the rating scale (in this case, the range is from 0 to 30). Multiple iterations are performed to find the optimal random state that results in the lowest Root Mean Squared Error (RMSE) on the test set. From a dataset that contains doctor details(doctor id, doctor name,phone number) doctor names and phone numbers based on doctor IDs are retrieved.

Chapter 7

Testing

7.1 Risk Prediction and Doctor Recommendation

Our Risk prediction models such as Naive Bayes,SVM(support vector machine) ,simple ANN was trained on the test set of Cleveland dataset and the performance of all models were analysed . It was tested using 20% of the Cleveland dataset. We got 70.49 % accuracy for Naive Bayes model with F1 score 0.705 and precision 0.709.For SVM model we got an accuracy of 82.78 % with F1 score 0.81 and precision 0.82.Finally ANN model was found out to have accuracy of 97.5 % with F1 score 0.98 and precision 0.99. The figure shown below is a graph plotted for training and validation loss of the ANN model for a given number of epochs.For doctor recommendation engine we tested the RMSE (Root Mean Square Error) and MAE(Mean Absolute Error) of the predicted rating and we got RMSE as 1.26 and MAE as 1.04.

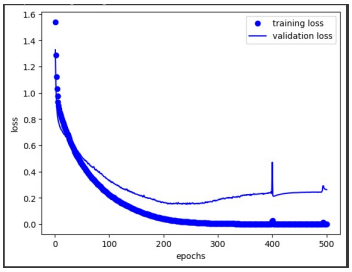


Figure 7.1: Loss over Epochs graph

Chapter 8

Results

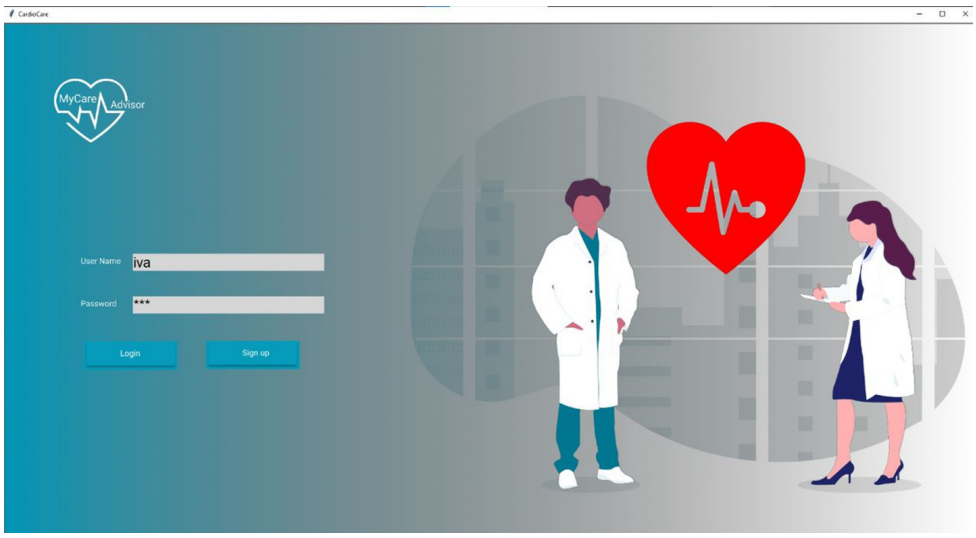


Figure 8.1: Login

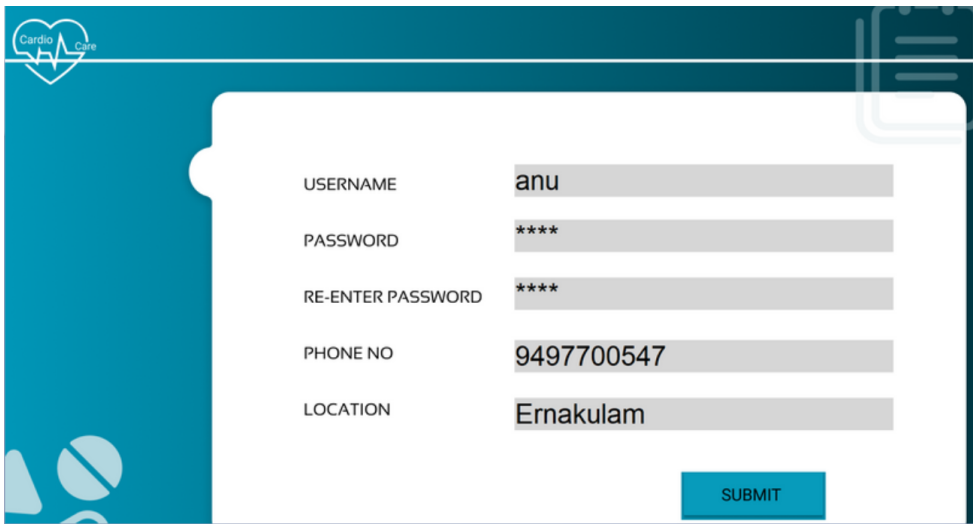


Figure 8.2: New User

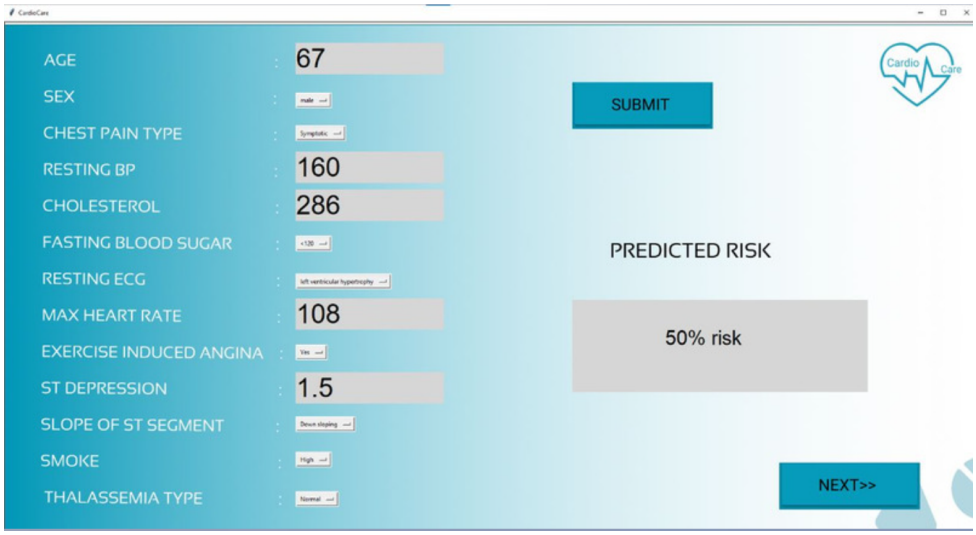


Figure 8.3: Input user data

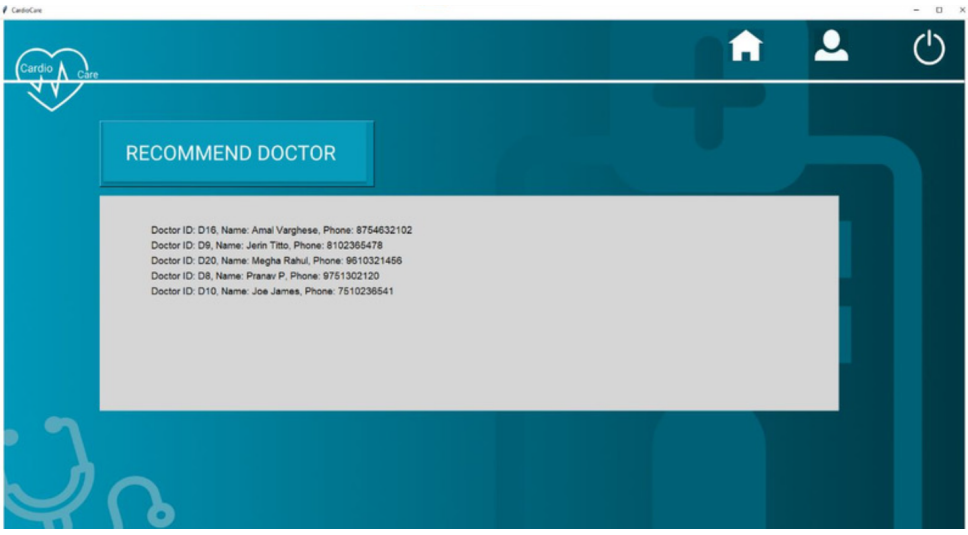


Figure 8.4: Recommend Doctor

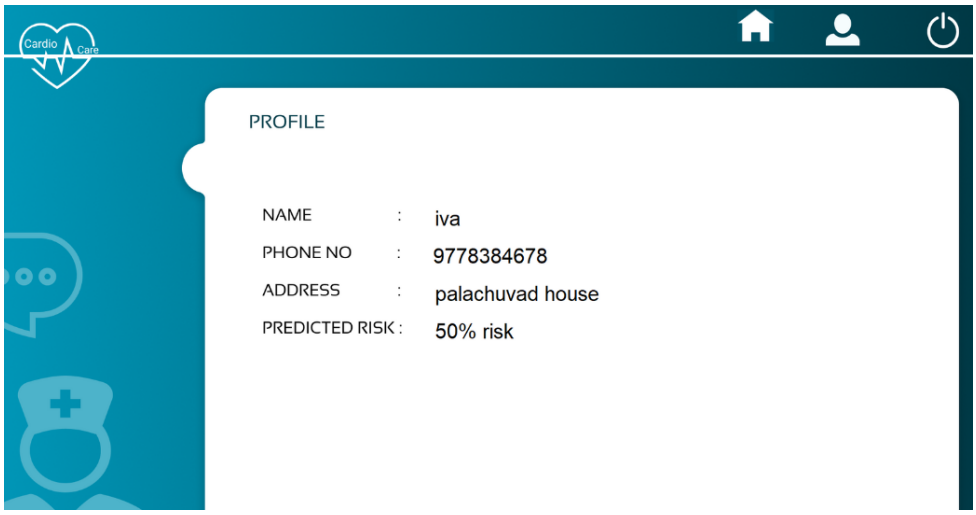


Figure 8.5: User Profile

Chapter 9

Risks and Challenges

1. To obtain an accurate dataset.

2. Incomplete or inconsistent data lead to biased or inaccurate predictions.

3. Availability and accuracy of doctor data.

4. Had to create a dummy dataset for doctor rating and doctor details

Chapter 10

Conclusion

We have developed a system that predicts the risk of heart disease in a person. And, if the user wishes, doctors are also recommended. All of these features have been verified to be working as intended.

We have used ANN model for risk prediction. The Softmax activation function was used in the final layer of the neural network to obtain the predicted probabilities for different classes (risk levels) after training the multi-class classification model. The GUI presents the predicted risk as a percentage.

Doctor recommendation was done using collaborative filtering approach.The KNN algorithm identifies the k most similar users to the new user.SVD model is trained on the doctor rating data to predict the ratings for doctors based on the ratings of similar users. The top recommended doctors to the new user based on the predicted ratings are identified. These recommendations are displayed on the GUI, allowing the user to access the information about the recommended doctors.

Future Scope

- To create a valid dataset for doctors-A valid dataset with rich and diverse information will improve the quality of doctor recommendations, leading to better user satisfaction and engagement with the system.
- To implement this system using new technologies and models- ML-Exploring and implementing state-of-the-art ML models and techniques can enhance the doctor recommendation system's accuracy and efficiency.
- To Implement web application or Android-specific technologies-Expands the project's reach and accessibility which increases the project's usability and user base, making it more accessible to a larger audience.

References

[1] Sarra, R.R.; Dinar, A.M.; Mohammed, M.A.; Abdulkareem, K.H. Enhanced Heart Disease Prediction Based on Machine Learning and 2 Statistical Optimal Feature Selection Model. Designs 2022, 6, 87. https://doi.org/10.3390/designs6050087

[2] Cui, Bei-Bei. "Design and implementation of movie recommendation system based on Knn collaborative filtering algorithm." ITM web of conferences. Vol. 12. EDP Sciences, 2017

[3] Sahoo, A.K.; Pradhan, C.; Barik, R.K.; Dubey, H. DeepReco: Deep Learning Based Health Recommender System Using Collaborative Filtering. Computation 2019, 7, 25. https://doi.org/10.3390/computation7020025

[4] Aljanabi, M., Mahmoud H. Qutqut, and Mohammad Hijjawi. "Machine learning classification techniques for heart disease prediction: a review." International Journal of Engineering Technology 7.4 (2018): 5373-5379.

[5] Perumal, Ramya, and A. C. Kaladevi. "Early prediction of coronary heart disease from cleveland dataset using machine learning techniques." Int. J. Adv. Sci. Technol 29 (2020): 4225-4234.

[6] Latha, C. Beulah Christalin, and S. Carolin Jeeva. "Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques." Informatics in Medicine Unlocked 16 (2019): 100203.

[7] naney-Obiri, Daniel, and Enoch Sarku. "Predicting the presence of heart diseases using comparative data mining and machine learning algorithms." International Journal of Computer Applications 176.11 (2020): 17-21.

Appendix A: Base Paper

designs

MDPI

Article

Enhanced Heart Disease Prediction Based on Machine Learning and χ^2 Statistical Optimal Feature Selection Model

Raniya R. Sarra [1], Ahmed M. Dinar [1,*], Mazin Abed Mohammed [2] and Karrar Hameed Abdulkareem [3]

[1] Computer Engineering Department, University of Technology-Iraq, Baghdad 19006, Iraq
[2] College of Computer Science and Information Technology, University of Anbar, Anbar 31001, Iraq
[3] College of Agriculture, Al-Muthanna University, Samawah 66001, Iraq
* Correspondence: ahmed.m.dinar@uotechnology.edu.iq; Tel.: +964-770-307-2072

Abstract: Automatic heart disease prediction is a major global health concern. Effective cardiac treatment requires an accurate heart disease prognosis. Therefore, this paper proposes a new heart disease classification model based on the support vector machine (SVM) algorithm for improved heart disease detection. To increase prediction accuracy, the χ^2 statistical optimum feature selection technique was used. The suggested model's performance was then validated by comparing it to traditional models using several performance measures. The proposed model increased accuracy from 85.29% to 89.7%. Additionally, the componential load was reduced by half. This result indicates that our system outperformed other state-of-the-art methods in predicting heart disease.

Keywords: heart disease; feature selection; machine learning; prediction; diagnosis

Citation: Sarra, R.R.; Dinar, A.M.; Mohammed, M.A.; Abdulkareem, K.H. Enhanced Heart Disease Prediction Based on Machine Learning and χ^2 Statistical Optimal Feature Selection Model. *Designs* **2022**, *6*, 87. https://doi.org/10.3390/designs6050087

Academic Editors: Elisabetta Sieni and Alexandre Schmid

Received: 20 June 2022
Accepted: 5 September 2022
Published: 29 September 2022

1. Introduction

At present, the load on a person has increased significantly due to increased work. Because of this dire circumstance that cannot be avoided, there is a high probability that the person will suffer from heart disease [1–3]. According to the 2018 World Heart Federation study, heart disease causes millions of deaths annually. A decrease in the amount of blood flowing to the brain, heart, lungs, and other important organs is the cause of heart disorders (HDs). Congestive heart failure is the most common type of cardiovascular disease and the least serious. In human anatomy, blood veins are responsible for transporting blood to the heart. Other factors that contribute to heart disease include defective heart valves, which can result in heart failure. A typical symptom of cardiac disease is muscle soreness in the upper abdominal area, which might be accompanied by anesthesia. Decreasing blood pressure, minimizing cholesterol, and engaging in regular physical activity are all suggested to lower the risk of heart disease. Heart disease is most related to angina pectoris, dilated cardiomyopathy, stroke, and congestive heart failure, among other things. As a result, it is necessary to monitor cardiovascular disease (CVD) biomarkers and consult with healthcare physicians [4–6].

Since ancient times, humans have significantly improved in terms of machines and health care. In modern times, after the entry of machines and artificial intelligent (AI) in medicine and health care, there has been significant developments and improvements in medicine and health care [7–10]. When it comes to heart disease, determining a person's risk of heart failure is a major concern [11,12]. The application of multiple longitudinal study auto-regression analyses results in the construction of the prediction method [13]. Because of changes in technology, healthcare facilities now must store a huge amount of data in their databases, which makes it very hard to figure out what the data means.

The study of how computers acquire knowledge via observation and experience is known as machine learning. Machine-learning (ML) algorithms have the potential to tackle a wide range of problems in the management of specific medical centers and the analysis

of data [14]. Analyzing and interpreting large datasets is made easier using these tools and methods. Heart disease is characterized by several factors, including age, cholesterol, weight, height, sex, blood pressure, resting ECG (electrocardiogram), chest pain, smoking, obesity, and eating habits [15,16]. Another challenge in this field is the large number of features used in heart disease prediction, which makes the task somewhat difficult. Additionally, the number of features makes it difficult to classify in machine learning and, in turn, affects performance and thus reduces the accuracy value of ML systems. Therefore, the following offers a significant contribution to this developed heart disease diagnosis:

1. To develop a new heart disease (HD) classification model based on the ML (SVM) algorithm to improve the detection of heart disease;
2. To implement an optimal feature selection model using the χ^2 statistical method for the extraction of the most informative attributes to improve prediction accuracy;
3. To validate the proposed heart disease diagnosis model's accuracy by comparing it with traditional models through the analysis of several performance metrics.

2. Methodology

The following are the two primary steps that have been carried out to meet the objectives of this work:

2.1. Support Vector Machine (SVM) Model

Figure 1 depicts the model's operation in detail. In the proposed support vector machine (SVM)-based heart disease prediction system, the most significant stages were data pre-processing, feature selection, and classifying. The feature normalization method was included in the pre-processing block. Training and testing sets were then created from the data. A feature scoring and selection algorithm was employed to ensure that the training subset was free of any biases. The χ^2 statistical model selected the same feature set for training and testing data. Next, training data with fewer features was fed into the SVM model for training purposes. Finally, using testing data, the trained SVM model was evaluated. The proposed model utilized 14 features from the University of California Irvine (UCI) Heart Disease Repository's Statlog and Cleveland datasets [17–20]. These features were examined using approaches that successfully predict heart disease. It was possible to develop and evaluate the system for heart disease prediction by using Python and the sci-kit learn library [21].

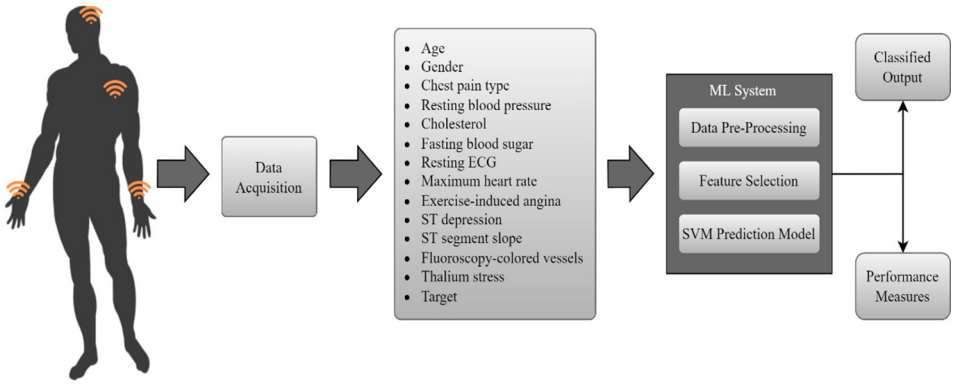


Figure 1. The proposed model's overall working process.

Vladimir N. V. and Alexey Ya were the first to propose the SVM method in their study of statistical learning theory [4]. The support vector machine (SVM) algorithm, a supervised learning ML method, can be used for both classification and regression purposes. Prediction of cardiovascular disease using the SVM method is more accurate and less error prone [22].

The SVM model learns how to separate various categories by a significant distance. It finds the optimum hyperplane for dividing sensitive data with the most significant margin. This margin is displayed as the distance between the hyperplane and the closest data

points on each side of it in Figure 2. The hyperplane computations based on the fact that many points of one group fall on one side of the plane [23]. The SVM model uses a trick kernel to transform data and then find the optimal surface that divides between different classes [21]. This paper used a radial basis function (RBF) kernel, denoted in Equation (1). SVM classification relies heavily on the RBF, commonly known as the Gaussian kernel, to map input data into a feature space. For each feature, the kernel function calculates the inner product of the data points.

$$k(x, y) = e^{-gamma\ \|x-y\|^2} \quad (1)$$

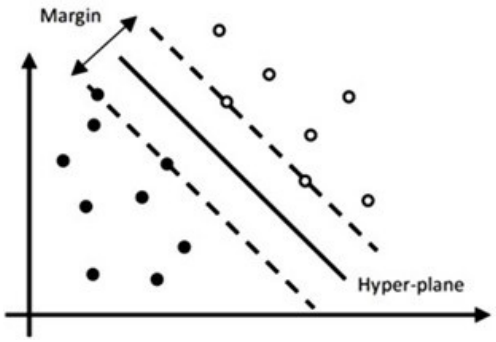


Figure 2. The margin between hyperplane and data.

Here, x and y are the two data points. The core functions determine the kernel parameters. For example, this experiment's RBF kernel had a parameter gamma, which had to be tweaked to find the optimal hyperplane. It specified how much of an impact a single training sample can have. Higher gamma values indicate a close influence [22,23].

Another factor that needed to be tuned was penalty factor C (Cost). The model's accuracy decreased as C decreased, while its generalizability increased. A more significant number for C improved model accuracy but reduced its generalizability. In our SVM (RBF) model, we set gamma equal to 1, divided by the number of features seen during model fitting, and C = 1.0 to maximize efficiency.

2.1.1. Heart Disease Dataset Features

In this study, we utilized two publicly available heart disease datasets, the Cleveland and Statlog (Heart) datasets, which were obtained from the University of California at the Irvine (UCI) machine-learning repository [19,20]. The datasets were chosen because they are the most widely used datasets by various researchers on heart disease prediction to find their model's effectiveness [24].

All 303 cases in the Cleveland dataset have 76 attributes. These include patients' identification numbers, ages, social security numbers, as well as a variety of health data, including details on the location and type of chest pain they were experiencing, measurements of their blood pressure and cholesterol levels, along with their fasting blood sugar and electrocardiogram readings [25]. Although the Cleveland heart disease database has 76 different features, most researchers only used 14 of them in their experiments [24]. With 270 instances, the Statlog (heart) dataset has 14 characteristics.

Table 1 shows a description of the attributes of both datasets, which have the same type and number of features as one another. In the prediction of heart disease, thirteen attributes are used, with the last attribute serving as the output that decides whether a person has heart disease.

Table 1. Attribute descriptions of Cleveland datasets [19].

Name	Type	Description
Age	Numeric	Age in years
Sex	Categorial	0 = Female or 1 = male
Cp	Categorial	Type of Chest pain (1 = typical angina, 2 = atypical angina, 3 = non anginal pain, 4 = asymptomatic)
Trestbps	Numeric	Resting blood pressure (mm hg)
Chol	Numeric	Serum cholesterol (mg/dL)
Fbs	Categorial	Fasting blood sugar > 120 mg/dL (0 = false, 1 = true)
Restecg	Categorial	Resting electrocardiography results (0 = normal, 1 = ST-T wave abnormality, 2 = probable or definite left ventricular hypertrophy)
Thalach	Numeric	Maximum heart rate achieved during thalium stress test
Exang	Categorial	Exercise-induced angina (1 = yes, 0 = no)
Oldpeak	Numeric	St depression induced by exercise relative to rest
Slope	Categorial	Slope of peak exercise ST segment (1 = upsloping, 2 = flat, 3 = downsloping)
Ca	Categorial	Number of significant vessels colored by fluoroscopy
Thal	Categorial	Thalium stress test result (3 = normal, 6 = fixed, 7 = reversible defect)
Num	Categorial	Heart disease status (0 = < 50% diameter narrowing, 1 = > 50% diameter narrowing)

The data distribution is critical when attempting to predict [26]. Figure 3 depicts the expected attribute class distributions for the two used datasets. There are 138 people in the Cleveland dataset who have no cardiac disease, and 165 patients in the dataset who do. A total of 120 people do not have heart disease in the Statlog dataset, whereas a total of 150 people do. Consequently, we have a dataset that is almost evenly distributed in terms of target output, which helps to prevent the overfitting issue.

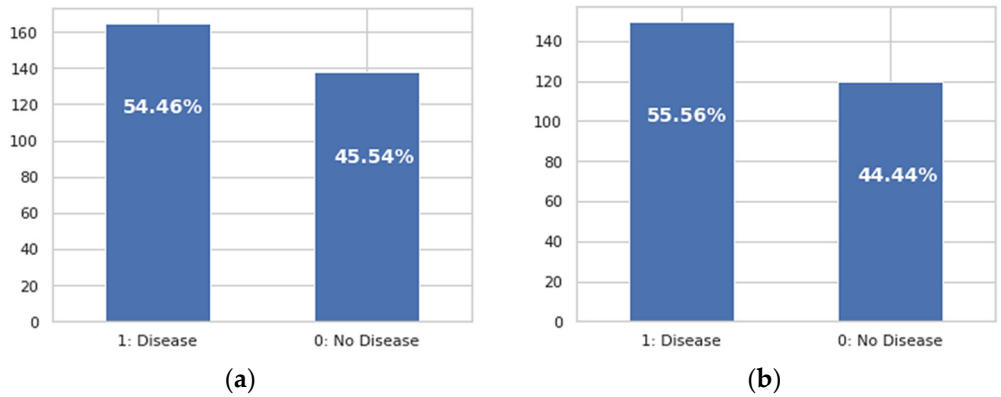
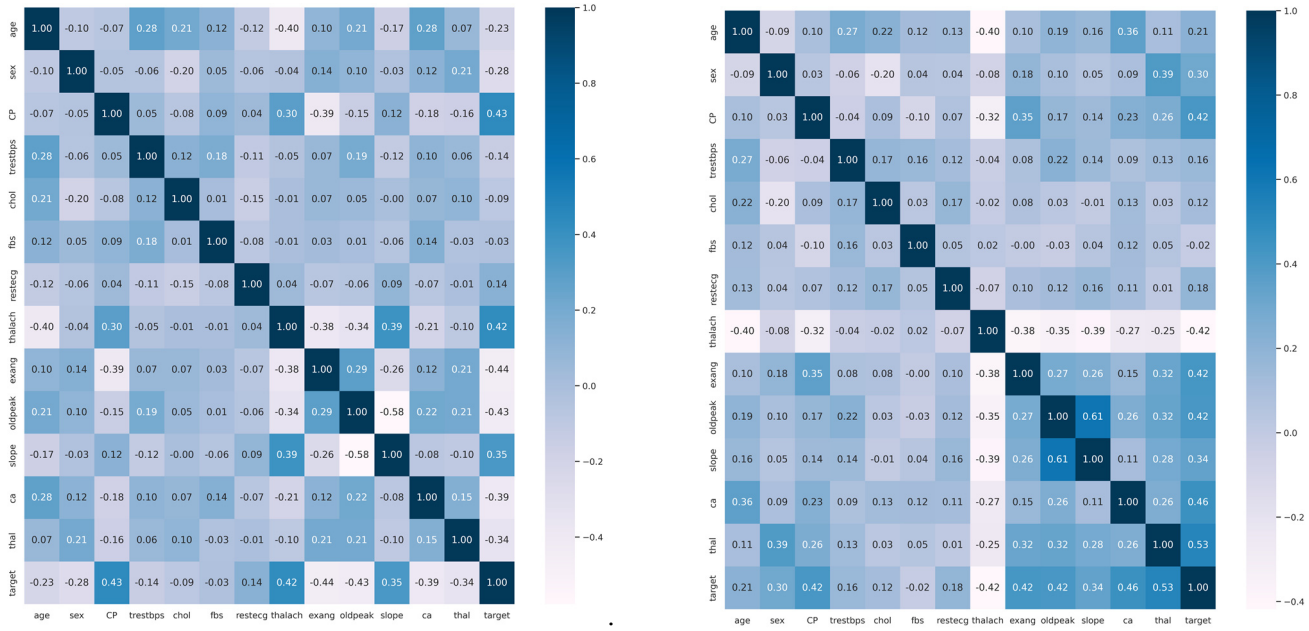


Figure 3. Heart disease classes. (**a**) Cleveland dataset; (**b**) Statlog dataset.

In Figure 4, a heatmap is used to display the correlation analysis of all attributes and the target. For each attribute, the heatmap's color indicates the degree to which that attribute is correlated with all the others and with the output target class. In general, the stronger the correlation, the warmer the color. For the Cleveland dataset, the target attribute was most closely associated with the Cleveland dataset's features of exercise-induced depression, the kind of chest pain, exercise-induced angina, and the maximum heart rate reached. Meanwhile, for the Statlog dataset, the highest correlated features with the target were thalium, no. of major vessels, ST depression, exercise-induced angina, maximum heart rate, and chest pain.



(**a**) (**b**)

Figure 4. Correlation heat map. (**a**) Cleveland dataset; (**b**) Statlog dataset.

2.1.2. Data Pre-Processing

There were no missing values in any of the derived datasets. Additionally, there was a roughly equal proportion of individuals with and without cardiac illness in the target variable, as indicated in Figure 3. This means that no target weighting would be needed in any way. However, a feature scaling approach was applied to ensure a normal distribution of the data. Features were scaled based on the standard scaler approach, which standardizes a feature by removing the mean and then scaling to a unit variance. The term "unit variance" refers to the process of dividing all the values by the standard deviation. As shown in Equation (2), to obtain the new data point (x), the mean (μ) value was subtracted from the old data point in a particular column, and the result was divided by the standard deviation (σ) [22].

$$\text{Standard outcome} = \frac{x - \mu}{\sigma} \quad (2)$$

Two common issues in any prediction system are the underfitting and overfitting of the training data. Underfitting happens when the generated model does not learn enough from the training data, resulting in poor training and testing data performance. Overfitting occurs when a model learns too much from the training data and achieves unsatisfactory results from even minor details [27,28]. Irrelevant features in the training data often result in model overfitting. Even if the SVM model performs well on training data, it may not generalize well. We propose eliminating irrelevant features by using χ^2 statistical model to overcome this issue.

2.2. Enhanced SVM Model with Feature Selection

Having too many features causes overfitting. Thus, it is important to select the right features for both training and testing data to improve model performance [29]. Features that are relevant to the ML model are selected, while those that are irrelevant or noisy are discarded [28]. This paper used the chi-squared (χ^2) statistical model [30,31] to select the essential features before applying the SVM model.

A chi-squared (χ^2) test is a correlation-based feature selection method that determines the correlation between the features and the predicted class. Each non-negative feature (X_i) computes chi-square statistics to determine which features depend on the predicted attribute. The higher the chi-square score, the more dependent the feature is on the

predicted class [24]. First, the commonly used 13 features are ranked according to their χ^2 test score. The χ^2 test rank features for a binary classification problem are as follows: Let us assume there are (t) instances and two classes, positive and negative. To determine the χ^2 test score, we construct Table 2.

Table 2. Calculation table for the χ^2 test score [27].

	Positive Class	**Negative Class**	**Total**
Feature X_i occurs	α	b	$\alpha + b = m$
Feature X_i does not occur	λ	y	$\lambda + y = t - m$
Total	$\alpha + \lambda = p$	$b + y = t - p$	t

Where (m) represents the sum of instances that include the feature (X_i), $(t - m)$ represents the sum of instances that do not include the feature (X_i), (p) represents the sum of positive instances, and $(t - p)$ represents the sum of all instances that are not positive.

The χ^2 test examines the difference between the expected count (E), and the observed count (O). The observed count (O) is the observed data (α, b, λ, and y), and the expected count (E) is calculated from the row total, column total, and overall total. If two features are independent, the observed count and the expected count are close. The α, b, λ, and y represent the observed values, and E_α, E_b, E_λ, and E_y represent the expected values. Then, assuming that the two occurrences are unrelated, the expected value (E_α) is calculated using Equation (3). Similarly, E_b, E_λ, and E_y are calculated. Finally, based on the general χ^2 test form shown in Equation (4), we calculate χ^2 score as shown in Equation (5) [27].

$$E_\alpha = (\alpha + b) \times \frac{(\alpha + b)}{t} \tag{3}$$

$$\chi^2 = \frac{1}{d} \sum_{k=1}^{n} \frac{(O_k - E_k)^2}{E_k} \tag{4}$$

$$\chi^2 = \frac{(\alpha - E_\alpha)^2}{E_\alpha} + \frac{(b - E_b)^2}{E_b} + \frac{(\lambda - E_\lambda)^2}{E_\lambda} + \frac{(y - E_y)^2}{E_y} \tag{5}$$

After ranking the features using Equation (5), we looked for the best subset of features (n) with the highest χ^2 score. In the beginning, we used a subset of (n = 1), i.e., the feature with the highest χ^2 score. This subset was then applied to the SVM model, and the performance results were recorded as we experimented with various hyperparameters. We selected a subset of the two most highly scored attributes (n = 2) as a second approach. Then, this selection was applied to the SVM model, and the results were saved. We iterated this process until we obtained the optimum subset of ranked features (n = 6) that gave the best performance.

The proposed feature selection algorithm, based on the χ^2 statistical method, recognized six notable features that can be selected for model training. As shown in Figure 5, regarding the Cleveland dataset, the algorithm selected the following features: thalach, oldpeak, ca, cp, exang, and chol. While in Figure 6, for the Statlog dataset, the algorithm selected the following features: maximum heart rate, number of major vessels, thallium stress result, exercise-induced ST depression, cholesterol, and exercise-induced angina.

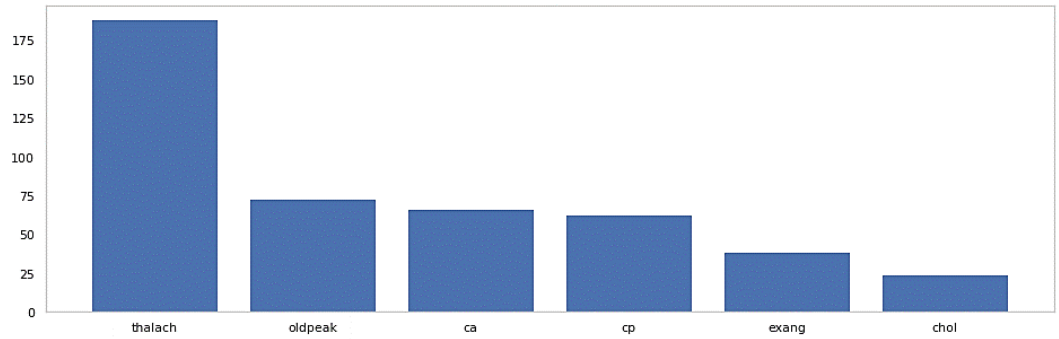


Figure 5. Selected features by highest χ^2 score (Cleveland dataset).

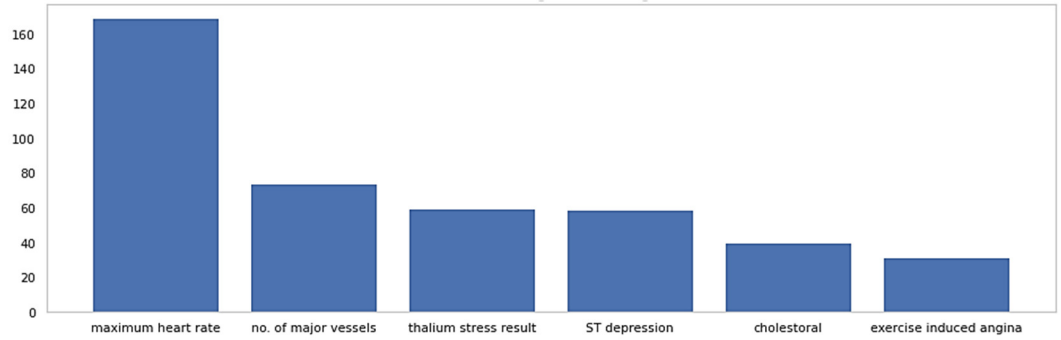


Figure 6. Selected features by highest χ^2 score (Statlog dataset).

3. Results and Discussion

In this paper, the SVM model was enhanced with the χ^2 statistical feature selection method. The feature selection method was used to select the six most important features for the prediction of heart disease. The χ^2-based SVM heart disease prediction model was developed and evaluated using Python and sci-kit learn library [21]. The two collected datasets (i.e., the Cleveland and Statlog (Heart) datasets) were partitioned into train and test sets. Training data was used to train the model, whereas testing data was used to evaluate the performance of the model [32]. To train and evaluate our proposed model, both datasets were split into a train and test set using a 75:25 split ratio. The following four primary parameters were assessed: true negative (T_N), which means that the algorithm prediction output for persons with no heart disease is correct; true positive (T_P), which means that the algorithm prediction output for heart disease patients is correct; false positive (F_P), which means that the patients who have no heart disease are incorrectly classified as having heart disease; and false negative (F_N), which refers to patients who are actually suffering from a cardiac disease but are incorrectly categorized as healthy [24]. The proposed χ^2-based SVM model was evaluated based on the following metrics:

- **Accuracy (Acc):** defined as the proportion of total positive instances of the model to the total number of instances, as shown in Equation (6).

$$\text{Accuracy} = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \tag{6}$$

- **Specificity (Spe):** the percentage of true negatives out of all healthy individuals, calculated by Equation (7). It was used to determine the degree of the attribute to appropriately classify the individuals without diseases.

$$\text{Specificity} = \frac{T_N}{T_N + F_P} \tag{7}$$

- **Sensitivity (Sen):** used to determine the degree of the attribute in order to appropriately classify the individuals who have diseases, as illustrated in Equation (8);

$$\text{Sensitivity} = \frac{T_P}{T_P + F_N} \tag{8}$$

- **F1-score:** defined as the harmonic mean of the specificity and sensitivity. It can be computed as shown in Equation (9);

$$\text{F1 Score} = \frac{2T_P}{2T_P + F_P + F_N} \tag{9}$$

The chi-squared-SVM algorithm was applied to the two collected datasets to see the difference when applied to different datasets. Two approaches were experimented with. In the first approach, the dataset with the total 14 features was normalized, then directly used for prediction using the SVM classifier. In the second approach, the χ^2-based feature selection method was applied to the normalized dataset to choose the six features that are the most significant for heart disease detection before applying the SVM classifier.

As can be shown in Table 3, by using the first approach (feeding the total 14 features of the collected datasets), the SVM classifier achieved the following results for the Cleveland dataset: accuracy of 84.21%; sensitivity of 67.45%; specificity of 84.13%; and an F1-score of 84.16%. Meanwhile, for the Statlog dataset, the following results were achieved: diagnostic accuracy of 85.29%; sensitivity of 68.29%; specificity of 85.36%; and an F1-score of 85.29%. The results of the SVM model for the Cleveland dataset are as follows, as shown in Table 4: diagnostic accuracy of 89.47%; sensitivity of 89.40%; specificity of 89.40%; and an F1-score of 89.40%. These results were achieved by applying the second approach, which used the χ^2 feature selection method.

Table 3. Performance of proposed model without feature selection.

Dataset	Total Records	Total Features	Acc (%)	Spe (%)	Sen (%)	F1 (%)
Cleveland	303	14	84.21	84.13	67.45	84.16
Statlog	270	14	85.29	85.36	68.29	85.29

Table 4. Performance of proposed model with feature selection.

Dataset	Total Records	Total Features	Acc (%)	Spe (%)	Sen (%)	F1 (%)
Cleveland	303	6	89.47	89.40	89.40	89.40
Statlog	270	6	89.70	89.70	89.74	89.70

The experimental results of applying these two approaches are shown in Figure 7. From Figure 7, conclusion can be drawn that the χ^2 feature selection method played a critical role in enhancing the accuracy of the SVM model, while also improving the results of sensitivity and specificity, which shows the model's ability to correctly identify people with and without the heart disease. The proposed χ^2-based SVM model improves classification accuracy by 6.25% for the Cleveland dataset and 5.17% for the Statlog dataset, which is important for providing a correct diagnosis and decreasing the rate of false predictions.

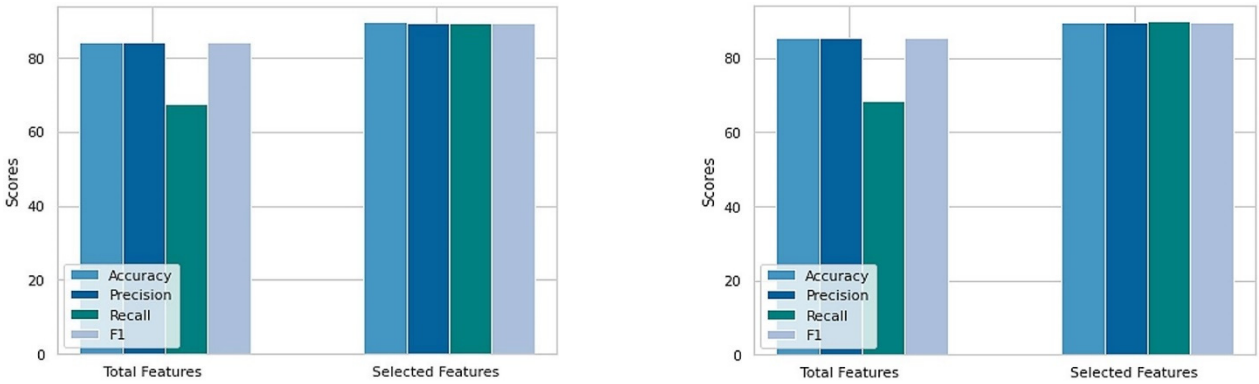


Figure 7. Performance of proposed algorithm. (**a**) Cleveland dataset; (**b**) Statlog dataset.

Furthermore, ROC (receiver operating characteristic) curve and AUC (area under the curve) charts were utilized to assess the performance of the proposed χ^2-based SVM diagnostic model and its ability to identify heart disease occurrence. The ROC and AUC chart is a 2D graph, between the sensitivity and specificity, which evaluates the validity of a diagnostic model. The true positive rate (*Y* axis) and the true negative rate (*X* axis) are plotted in the ROC chart. It indicates that the optimal ROC curve is in the plot's upper left corner. An ROC chart with a bigger AUC is better, which indicates that a diagnostic model can correctly identify people with heart issues [27].

ROC curves are given in Figure 8a before and after the 14 features of the Cleveland dataset were reduced to 6. The AUC of the SVM model was 0.90 after lowering the features by the χ^2 method, but it was 0.91 when using the full set of features. Figure 8b depicts the same thing, but with the Statlog dataset. Before using the χ^2 feature selection method, the SVM model's AUC was 0.94; after using it, the AUC dropped to 0.91. This suggests that the influence of χ^2 feature selection approach was minimal in terms of AUC.

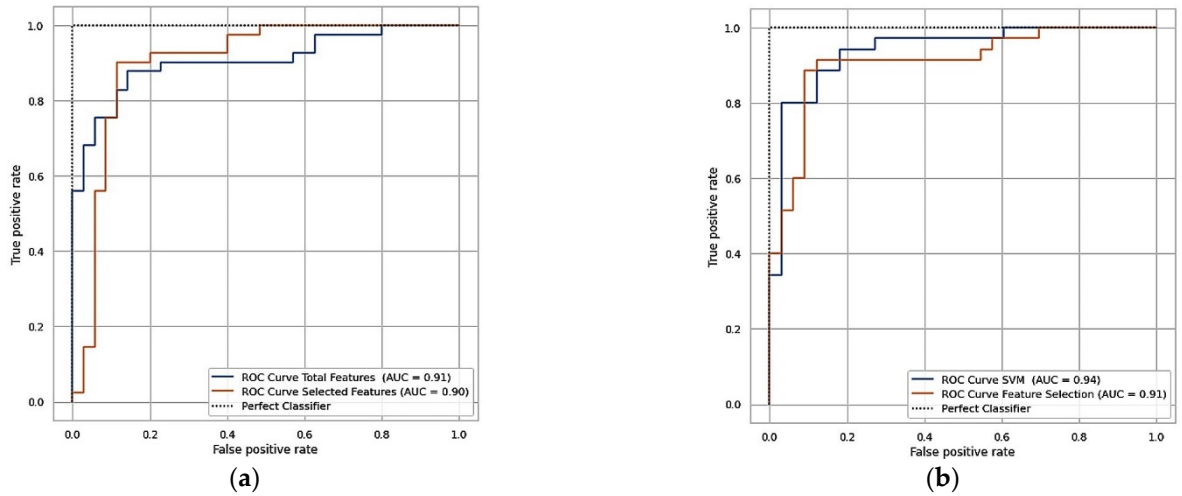
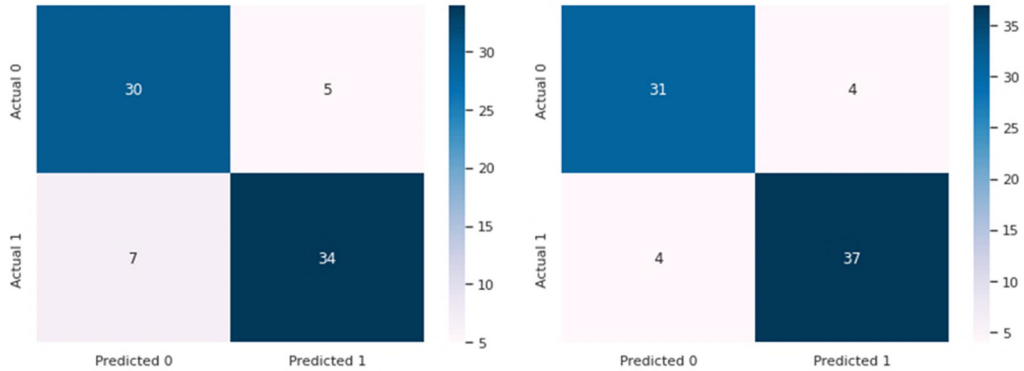


Figure 8. AUC and ROC analysis. (**a**) Cleveland dataset; (**b**) Statlog dataset.

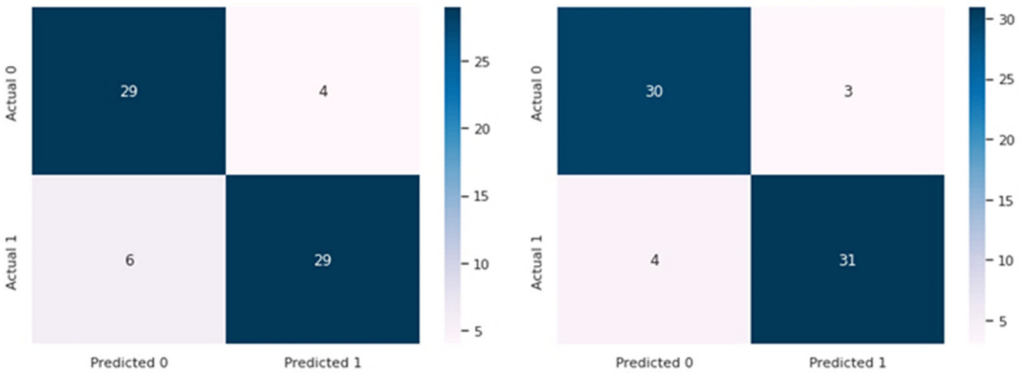
To evaluate the proposed χ^2-based SVM model in detecting heart disease, another metric that goes by the term "confusion matrix" was utilized. In a confusion matrix, the values of the true positive (T_P) and false negative (F_N) parameters are laid out in a format like that of a table. The confusion matrix summarizes the number of correct and incorrect predictions. Figure 9b illustrates the confusion matrix that was produced as a result of using the proposed χ^2-based SVM model on the Cleveland dataset. It shows that the proposed model can correctly detect 37 (predicted 1 and actual 1) heart diseased persons and identify 31 healthy subjects out of 35 (predicted 0 and actual 0). In Figure 10b, the

resulting confusion matrix of the Statlog dataset is presented. This demonstrates that the methodology that was proposed can accurately identify 31 heart disease patients and identify 30 out of 33 healthy subjects. Both Figures 9 and 10 show that the number of incorrect predictions (actual 1 but predicted 0, and actual 0 but predicted 1) made by the SVM model before and after applying the χ^2 feature selection method was reduced from 12 to 8 for the Cleveland dataset, and from 10 to 7 for the Statlog dataset.



(**a**) (**b**)

Figure 9. Confusion matrix of Cleveland dataset. (**a**) Total features; (**b**) Selected features.



(**a**) (**b**)

Figure 10. Confusion matrix of Statlog dataset. (**a**) Total features; (**b**) Selected features.

Furthermore, concerning the accuracy and number of selected features, our proposed model was compared to several existing state-of-the-art techniques, as shown in Table 5. The performance of the suggested chi-squared-SVM methodology was shown to perform better than other methods, with an accuracy of 89.47%.

Table 5. Proposed model comparative analysis with current state-of-the-art methods.

Study	Feature Selection Method	Dataset	Classifier	Total Features	Acc (%)	Selected Features	Acc (%)
[33]	Lasso [1]	Cleveland	Ensemble	13	-	8	75.1
[34]	Information Gain	Cleveland	Ensemble	27	72.2	16	83.5
[26]	Lasso	Cleveland	SVM	13	84.09	-	84.26
[35]	Randomly Generated Feature Set	Cleveland	Ensemble	13	-	9	85.48
[24]	Ruzzo-Tompa	Cleveland	ANN [2]	13	-	7	86.20
[36]	Lasso	Cleveland	SVM (RBF)	13	86	6	88
[37]	PSO-SVM [3]	Cleveland	SVM	13	79.35	6	88.22
[28]	PS-GWO	Statlog and Cleveland	DBN [4]	13	-	-	88.8
[38]	PCA [5]	Cleveland	DL [6]	13	-	-	89
Proposed	Chi-squared		SVM	13	85.29	6	89.47

[1] Lasso: least absolute shrinkage and selection operator; [2] ANN: artificial neural network; [3] PSO: swarm optimizations; [4] DBN: deep belief network; [5] PCA: principal component analysis; [6] DL: deep learning.

4. Conclusions

In this work, an enhanced model was implemented to increase the heart disease diagnosis and prediction accuracy, as well as to reduce computational load. The ML (SVM) algorithm was used as a classification model for enhanced heart disease diagnoses. This model was performed on two famous heart disease datasets. The results showed increasing accuracy from 84.21% to 89.47 and from 85.29% to 89.7% in the Cleveland and Statlog datasets, respectively. Furthermore, the features used in the system were decreased from 14 to 6 features, which means that the computational load was reduced from 100% to approximately 42%. We anticipate that this work will contribute to the future development and implementation of heart disease prediction and diagnosis systems.

Author Contributions: Conceptualization, R.R.S. and A.M.D.; methodology, R.R.S.; writing—original draft preparation, R.R.S.; writing—review and editing, A.M.D., K.H.A. and M.A.M.; supervision, A.M.D. and M.A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The machine-learning repository at UCI provides access to the datasets that were used in this study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Das, S.; Sharma, R.; Gourisaria, M.; Rautaray, S.; Pandey, M. Heart disease detection using core machine learning and deep learning techniques: A comparative study. *Int. J. Emerg. Technol.* **2020**, *11*, 531–538.
2. Hasan, T.T.; Jasim, M.H.; Hashim, I.A. FPGA Design and Hardware Implementation of Heart Disease Diagnosis System Based on NVG-RAM Classifier. In Proceedings of the 2018 Third Scientific Conference of Electrical Engineering (SCEE), Baghdad, Iraq, 19–20 December 2018; pp. 33–38. [CrossRef]
3. Rahman, A.U.; Saeed, M.; Mohammed, M.A.; Jaber, M.M.; Garcia-Zapirain, B. A novel fuzzy parameterized fuzzy hypersoft set and riesz summability approach based decision support system for diagnosis of heart diseases. *Diagnostics* **2022**, *12*, 1546. [CrossRef] [PubMed]
4. Javid, I.; Khalaf, A.; Ghazali, R. Enhanced accuracy of heart disease prediction using machine learning and recurrent neural networks ensemble majority voting method. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 540–551. [CrossRef]
5. Muhsen, D.K.; Khairi, T.W.A.; Alhamza, N.I.A. Machine Learning System Using Modified Random Forest Algorithm. In *Intelligent Systems and Networks, Singapore*; Tran, D.-T., Jeon, G., Nguyen, T.D.L., Lu, J., Xuan, T.-D., Eds.; Springer: Singapore, 2021; pp. 508–515.
6. Wah, T.Y.; Mohammed, M.A.; Iqbal, U.; Kadry, S.; Majumdar, A.; Thinnukool, O. Novel DERMA fusion technique for ECG heartbeat classification. *Life* **2022**, *12*, 842.
7. Mohammed, M.A.; Abdulkareem, K.H.; Al-Waisy, A.S.; Mostafa, S.A.; Al-Fahdawi, S.; Dinar, A.M.; Alhakami, W.; Abdullah, B.A.Z.; Al-Mhiqani, M.N.; Alhakami, H.; et al. Benchmarking methodology for selection of optimal COVID-19 diagnostic model based on entropy and TOPSIS methods. *IEEE Access* **2020**, *8*, 99115–99131. [CrossRef]
8. Dinar, A.M.; Zain, A.M.; Salehuddin, F. Utilizing of CMOS ISFET sensors in DNA applications detection: A systematic review. *J. Adv. Res. Dyn. Control Syst.* **2018**, *10*, 569–583.
9. Soni, M.; Gomathi, S.; Kumar, P.; Churi, P.P.; Mohammed, M.A.; Salman, A.O. Hybridizing Convolutional Neural Network for Classification of Lung Diseases. *Int. J. Swarm Intell. Res. (IJSIR)* **2022**, *13*, 1–15. [CrossRef]
10. Nasser, A.R.; Hasan, A.M.; Humaidi, A.J.; Alkhayyat, A.; Alzubaidi, L.; Fadhel, M.A.; Santamaría, J.; Duan, Y. IoT and Cloud Computing in Health-Care: A New Wearable Device and Cloud-Based Deep Learning Algorithm for Monitoring of Diabetes. *Electronics* **2021**, *10*, 2719. Available online: https://www.mdpi.com/2079-9292/10/21/2719 (accessed on 1 May 2022).
11. Diwakar, M.; Tripathi, A.; Joshi, K.; Memoria, M.; Singh, P. Latest trends on heart disease prediction using machine learning and image fusion. *Mater. Today Proc.* **2021**, *37*, 3213–3218. [CrossRef]
12. Rahman, A.U.; Saeed, M.; Mohammed, M.A.; Krishnamoorthy, S.; Kadry, S.; Eid, F. An Integrated Algorithmic MADM Approach for Heart Diseases' Diagnosis Based on Neutrosophic Hypersoft Set with Possibility Degree-Based Setting. *Life* **2022**, *12*, 729. [CrossRef] [PubMed]
13. Hu, G.; Root, M.M. Building prediction models for coronary heart disease by synthesizing multiple longitudinal research findings. *Eur. J. Prev. Cardiol.* **2005**, *12*, 459–464. [CrossRef] [PubMed]
14. Deo, R.C. Machine learning in medicine. *Circulation* **2015**, *132*, 1920–1930. [CrossRef] [PubMed]

15. Mythili, T.; Mukherji, D.; Padalia, N.; Naidu, A. A heart disease prediction model using SVM-decision trees-logistic regression (SDL). *Int. J. Comput. Appl.* **2013**, *68*, 0975–8887.
16. Elhoseny, M.; Mohammed, M.A.; Mostafa, S.A.; Abdulkareem, K.H.; Maashi, M.S.; Garcia-Zapirain, B.; Mutlag, A.A.; Maashi, M.S. A new multi-agent feature wrapper machine learning approach for heart disease diagnosis. *Comput. Mater. Contin* **2021**, *67*, 51–71. [CrossRef]
17. Detrano, R.; Janosi, A.; Steinbrunn, W.; Pfisterer, M.; Schmid, J.J.; Sandhu, S.; Guppy, K.H.; Lee, S.; Froelicher, V. International application of a new probability algorithm for the diagnosis of coronary artery disease. *Am. J. Cardiol.* **1989**, *64*, 304–310. [CrossRef]
18. Gennari, J.H.; Langley, P.; Fisher, D. Models of incremental concept formation. *Artif. Intell.* **1989**, *40*, 11–61. [CrossRef]
19. Janosi, A.; Steinbrunn, W.; Pfisterer, M.; Detrano, R. UCI Machine Learning Repository: Heart Disease Dataset [Online]. Available online: https://archive-beta.ics.uci.edu/ml/datasets/heart+disease (accessed on 1 March 2022).
20. Machine Learning Repository: Statlog (Heart) [Online]. Available online: http://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29 (accessed on 1 March 2022).
21. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
22. Sajja, T.K.; Kalluri, H.K. A Deep Learning Method for Prediction of Cardiovascular Disease Using Convolutional Neural Network. *Rev. D'intelligence Artif.* **2020**, *34*, 601–606. [CrossRef]
23. Guo, C.; Zhang, J.; Liu, Y.; Xie, Y.; Han, Z.; Yu, J. Recursion enhanced random forest with an improved linear model (RERF-ILM) for heart disease detection on the internet of medical things platform. *IEEE Access* **2020**, *8*, 59247–59256. [CrossRef]
24. Ali, S.A.; Raza, B.; Malik, A.K.; Shahid, A.R.; Faheem, M.; Alquhayz, H.; Kumar, Y.J. An optimally configured and improved deep belief network (OCI-DBN) approach for heart disease prediction based on Ruzzo–Tompa and stacked genetic algorithm. *IEEE Access* **2020**, *8*, 65947–65958. [CrossRef]
25. Vijayashree, J.; Parveen Sultana, H. Heart disease classification using hybridized Ruzzo-Tompa memetic based deep trained Neocognitron neural network. *Health Technol.* **2020**, *10*, 207–216. [CrossRef]
26. Bharti, R.; Khamparia, A.; Shabaz, M.; Dhiman, G.; Pande, S.; Singh, P. Prediction of Heart Disease Using a Combination of Machine Learning and Deep Learning. *Comput. Intell. Neurosci.* **2021**, *2021*, 8387680. [CrossRef] [PubMed]
27. Ali, L.; Rahman, A.; Khan, A.; Zhou, M.; Javeed, A.; Khan, J.A. An Automated Diagnostic System for Heart Disease Prediction Based on $\chi 2$ Statistical Model and Optimally Configured Deep Neural Network. *IEEE Access* **2019**, *7*, 34938–34945. [CrossRef]
28. Aliyar Vellameeran, F.; Brindha, T. A new variant of deep belief network assisted with optimal feature selection for heart disease diagnosis using IoT wearable medical devices. *Comput. Methods Biomech. Biomed. Eng.* **2021**, *25*, 387–411. [CrossRef]
29. Yue, W.; Wang, Z.; Chen, H.; Payne, A.; Liu, X. Machine Learning with Applications in Breast Cancer Diagnosis and Prognosis. *Designs* **2018**, *2*, 13. [CrossRef]
30. Ali, L.; Zhu, C.; Zhou, M.; Liu, Y. Early diagnosis of Parkinson's disease from multiple voice recordings by simultaneous sample and feature selection. *Expert Syst. Appl.* **2019**, *137*, 22–28. [CrossRef]
31. Liu, H.; Setiono, R. Chi2: Feature selection and discretization of numeric attributes. In Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, 5–8 November 1995; pp. 388–391.
32. Maldonado, S.; Pérez, J.; Weber, R.; Labbé, M. Feature selection for support vector machines via mixed integer linear programming. *Inf. Sci.* **2014**, *279*, 163–175. [CrossRef]
33. Shorewala, V. Early detection of coronary heart disease using ensemble techniques. *Inform. Med. Unlocked* **2021**, *26*, 100655. [CrossRef]
34. Ali, F.; El-Sappagh, S.; Islam, S.R.; Kwak, D.; Ali, A.; Imran, M.; Kwak, K.S. A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion. *Inf. Fusion* **2020**, *63*, 208–222. [CrossRef]
35. Latha, C.B.C.; Jeeva, S.C. Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques. *Inform. Med. Unlocked* **2019**, *16*, 100203. [CrossRef]
36. Haq, A.U.; Li, J.P.; Memon, M.H.; Nazir, S.; Sun, R. A Hybrid Intelligent System Framework for the Prediction of Heart Disease Using Machine Learning Algorithms. *Mob. Inf. Syst.* **2018**, *2018*, 3860146. [CrossRef]
37. Vijayashree, J.; Sultana, H.P. A Machine Learning Framework for Feature Selection in Heart Disease Classification Using Improved Particle Swarm Optimization with Support Vector Machine Classifier. *Program. Comput. Softw.* **2019**, *44*, 388–397. [CrossRef]
38. Tuli, S.; Basumatary, N.; Gill, S.S.; Kahani, M.; Arya, R.C.; Wander, G.S.; Buyya, R. HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments. *Future Gener. Comput. Syst.* **2020**, *104*, 187–200. [CrossRef]

Appendix B: Sample Code

```
from tkinter import *
import mysql.connector
import tkinter.messagebox
#import mysql.connector as mcon
import tkinter.messagebox
import pandas as pd
import csv
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import svm
from sklearn.metrics import confusion_matrix, f1_score, roc_curve, auc
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.utils import shuffle
from sklearn.metrics import precision_score
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from surprise import accuracy
from surprise.model_selection import train_test_split

#from sklearn.decomposition import PCA
root = Tk()
root.geometry("1920x1080")
root.title("CardioCare")

mypass = "1234"
mydatabase="db"
con = mysql.connector.connect (host="localhost",user="root",password=mypass,database=mydatabase)
global repassword
print(con)
if(con):
    print("success")
else:
    print("noo")

cur = con.cursor()
```

LOGIN PAGE

```
def login():
    global
loginpage,login_user,login_password,login_login_b,login_newuser_b,login_f,newuser_f,profile_f,cardio_f
    newuser_f.destroy()
    cardio_f.destroy()
    recdoc_f.destroy()
    profpage_f.destroy()


    loginpage = PhotoImage(file = 'Login1.png')
    login_login_b = PhotoImage(file = 'login_b.png')
    login_newuser_b= PhotoImage(file = 'sign_b.png')

    login_f=Frame(root)
    login_f.pack()
    login_c=Canvas(login_f,height=1080,width=1920,bg='#FFFFFF')
    login_c.pack(fill = "both",expand=True)

    login_c.create_image( 0, 0,image =loginpage,anchor=NW)

  #user textbox
    login_user = Entry(login_c,width=25,font=('Poppins', 20, 'normal'), bg='#D6D6D6', bd=0)
    login_user.place(x=255, y=455)
 # user passwrd
    login_password = Entry(login_c,show='*',width=25, font=('Poppins', 20, 'normal'), bg='#D6D6D6', bd=0)
    login_password.place(x=255, y=539)
#login button

login_b=Button(login_c,image=login_login_b,borderwidth=0,bg='#079BBB',command=login_sql,cursor='h
and1')
    login_b.place(x=158, y=628)
#new user button

newuser_b=Button(login_c,image=login_newuser_b,borderwidth=0,bg='#079BBB',command=newuser,cu
rsor='hand2')
    newuser_b.place(x=399, y=627)
```

NEW USER

```
def newuser():
    global
newuser_bg,newuser_username,newuser_pw,newuser_rpw,newuser_city,newuser_ph,newuser_button
    global newuser_b,cardio_f,newuser_f,newuser_addr

    login_f.destroy()
    cardio_f.destroy()
```

```
    recdoc_f.destroy()
    profpage_f.destroy()


    newuser_f=Frame(root)
    newuser_f.pack()
    newuser_bg = PhotoImage(file = 'newuser1.png')
    newuser_button = PhotoImage(file = 'newuser_sub.png')
    #home_sub = PhotoImage(file = 'home_b.png')
    newuser_c=Canvas(newuser_f,height=1080,width=1920,bg='#FFFFFF')
    newuser_c.create_image(0,0,image=newuser_bg,anchor=NW)
    newuser_c.pack(fill = "both",expand=True)
    newuser_username = Entry(newuser_c,width=25, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
    newuser_username.place(x=995, y=310)
    newuser_pw = Entry(newuser_c,width=25,show='*', font=('Poppins', 40, 'normal'), bg='#D6D6D6',
bd=0)
    newuser_pw.place(x=995, y=416)
    newuser_rpw = Entry(newuser_c,width=25,show='*', font=('Poppins', 40, 'normal'), bg='#D6D6D6',
bd=0)
    newuser_rpw.place(x=995, y=527)

    newuser_ph = Entry(newuser_c,width=25, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
    newuser_ph.place(x=995, y=647)
    newuser_addr = Entry(newuser_c,width=25, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
    newuser_addr.place(x=995, y=758)




newuser_b=Button(newuser_c,image=newuser_button,borderwidth=0,bg='#018CAB',command=newuser
_sql,cursor='hand1')
    newuser_b.place(x=1315, y=900)
def login_sql():
    global user,pass_get,query,ls,x

    userTable = "user"
    user =login_user.get()
    pass_get = login_password.get()

    query =  "select * from "+userTable+" where uname = '"+user+"' and password = '"+pass_get+"' "
    cur.execute(query)

    ls= cur.fetchall()

    if(len(ls)!=0):
        name=ls[0][0]
        password=ls[0][1]
        print(name,password)
    #if(pass_get==password and user==name):
```

```
        print('login successful')
        cardio()
    else:

         tkinter.messagebox.showinfo("ERROR",  "Incorrect credentials")

def newuser_sql():
    username=newuser_username.get()
    password=newuser_pw.get()
    repassword=newuser_rpw.get()
    address=newuser_addr.get()
    phone=newuser_ph.get()
    if(password!='' and username!=''):
        if password==repassword:
            insertCars = "insert into user values ('"+username+"','"+password+"','"+phone+"','"+address+"')"
            cur.execute(insertCars)
            con.commit()
            login()
        else:
            tkinter.messagebox.showinfo("ERROR",  "Not the same password!!")
            print("errorr")
    else:
            tkinter.messagebox.showinfo("ERROR",  "PLEASE ENTER A VALID USERNAME AND
PASSWORD")
    s="select * from user"
    q=cur.execute(s)
    print(q)
    print(cur)
    print(cur.fetchall())
```

HOME PAGE

```
def cardio():
    global
cardio_bg,profpage_f,cardio_age,cardio_n,cardio_sex,cardio_cp,cardio_rbp,cardio_chol,cardio_fbs
    global cardio_recg,cardio_thl,cardio_exang,cardio_olp,cardio_slope,cardio_ca,cardio_thal,cardio_doc
    global
cardio_sub,recdoc_f,cardio_f,cardio_next,cardio_c,profile_f,op_sex,cardio_sex,op_cp,op_exang,op_fbs,o
p_recg,op_thal,op_slope,op_ca
    profpage_f.destroy()
    login_f.destroy()
    newuser_f.destroy()
    profpage_f.destroy()
    recdoc_f.destroy()

    print('hello')
    cardio_f=Frame(root)
    cardio_f.pack()
```

```
cardio_bg = PhotoImage(file = 'cardio_bg.png')
cardio_sub = PhotoImage(file = 'newuser_sub.png')
cardio_n= PhotoImage(file = 'next_sub.png')

cardio_c=Canvas(cardio_f,height=1080,width=1920,bg='#FFFFFF')
cardio_c.create_image(0,0,image=cardio_bg,anchor=NW)

lsex=['male','female']
lcp=['Atypical Angina','Typical Angina','Asymptotic','Symptotic']
lfbs=['<120','>120']
lrecg=['Normal','ST-T wave abnormalities','left ventricular hypertrophy']
lexang=['No','Yes']
lslope=['Unsloping','Flat','Down sloping']
lthal=['Normal','Fixed','Reversible effect']
lca=['Very low','Low','High','Very high']
OPTIONS = lsex
OPTIONS_CP=lcp
OPTIONS_FBS=lfbs
OPTIONS_RECG=lrecg
OPTIONS_EXANG=lexang
OPTIONS_SLOPE=lslope
OPTIONS_THAL=lthal
OPTIONS_CA=lca
op_ca = StringVar()
op_ca.set("Select option")
op_thal = StringVar()
op_thal.set("Select option")
op_recg = StringVar()
op_recg.set("Select option")
op_exang = StringVar()
op_exang.set("Select option")
op_slope = StringVar()
op_slope.set("Select option")
op_fbs = StringVar()
op_fbs.set("Select option")
op_cp = StringVar()
op_cp.set("Select option")
op_sex = StringVar()
op_sex.set("Select option")

def clear_placeholder(event):
    current_text = cardio_rbp.get("1.0", "end-1c")
    if current_text == range_rbp:
        cardio_rbp.delete("1.0", "end")
def clear_placeholder_chol(event):
    current_text = cardio_chol.get("1.0", "end-1c")
    if current_text == range_chol:
        cardio_chol.delete("1.0", "end")
```

```
def clear_placeholder_thl(event):
    current_text = cardio_thl.get("1.0", "end-1c")
    if current_text == range_thl:
        cardio_thl.delete("1.0", "end")

def clear_placeholder_olp(event):
    current_text = cardio_olp.get("1.0", "end-1c")
    if current_text == range_olp:
        cardio_olp.delete("1.0", "end")


range_rbp = "94 - 200"
range_chol = "126 - 564"
range_thl = "71 - 202"
range_olp = "0 - 6.2"

cardio_c.pack(fill = "both",expand=True)
cardio_age = Text(cardio_c,width=10,height=1, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
cardio_age.place(x=575, y=42)
cardio_sex = OptionMenu(cardio_c,op_sex,*OPTIONS)
cardio_sex.place(x=575, y=140)
cardio_cp = OptionMenu(cardio_c,op_cp,*OPTIONS_CP)
cardio_cp.place(x=575, y=205)
cardio_rbp = Text(cardio_c,width=10,height=1, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
cardio_rbp.insert('end', range_rbp)
cardio_rbp.place(x=575, y=257)
cardio_rbp.bind("<FocusIn>", clear_placeholder)
cardio_chol = Text(cardio_c,width=10,height=1, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
cardio_chol.insert('end', range_chol)
cardio_chol.bind("<FocusIn>", clear_placeholder_chol)
cardio_chol.place(x=575, y=332)
cardio_fbs = OptionMenu(cardio_c,op_fbs,*OPTIONS_FBS)
cardio_fbs.place(x=575, y=423)
cardio_recg = OptionMenu(cardio_c,op_recg,*OPTIONS_RECG)
cardio_recg.place(x=575, y=498)
cardio_thl = Text(cardio_c,width=10,height=1, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
cardio_thl.insert('end', range_thl)
cardio_thl.place(x=575,y=547)
cardio_thl.bind("<FocusIn>", clear_placeholder_thl)
cardio_exang = OptionMenu(cardio_c,op_exang,*OPTIONS_EXANG)
cardio_exang.place(x=575, y=638)
cardio_olp = Text(cardio_c,width=10,height=1, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
cardio_olp.insert('end', range_olp)
cardio_olp.place(x=575, y=693)
cardio_olp.bind("<FocusIn>", clear_placeholder_olp)
cardio_slope = OptionMenu(cardio_c,op_slope,*OPTIONS_SLOPE)
cardio_slope.place(x=575, y=780)
cardio_ca = OptionMenu(cardio_c,op_ca,*OPTIONS_CA)
cardio_ca.place(x=575, y=850)
```

```
    cardio_thal = OptionMenu(cardio_c,op_thal,*OPTIONS_THAL)
    cardio_thal.place(x=575, y=928)


    cardio_doc = Text(cardio_c,width=20,height=3, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
    cardio_doc.place(x=1120, y=550)

cardio_b=Button(cardio_c,image=cardio_sub,command=cardio_check,borderwidth=0,bg='#018CAB',curs
or='hand1')
    cardio_b.place(x=1120, y=120)


cardio_next=Button(cardio_c,image=cardio_n,borderwidth=0,bg='#018CAB',command=recdoc,cursor='ha
nd2')
    cardio_next.place(x=1528, y=872)


def cardio_check():
    global cardio_rbp_value,cardio_chol_value
    print("cardio check works")
    cardio_rbp_value = cardio_rbp.get("1.0", "end-1c")
    cardio_chol_value = cardio_chol.get("1.0", "end-1c")


    if(cardio_age.get("1.0", "end-1c")==" or op_sex.get()=="or cardio_rbp_value.strip()==" or
cardio_rbp_value=="94 - 200" or cardio_chol_value.strip()==" or cardio_chol_value=="126 - 564"):
        tkinter.messagebox.showinfo("ERROR",  "*AGE,SEX,RESTING BP,CHOLESTEROL are required
fields!!*")
        print("msg box donee")
    else:
        prediction()
```

RISK PREDICTION - ANN

```
def prediction():
    global
recdoc_bg,recdoc_loc,recdoc_doc,recdoc_sub,recdoc_b,recdoc_f,op_sex,op_cp,op_exang,op_fbs,op_re
cg,op_thal,op_slope,op_ca
    global risk_prediction
    df = pd.read_csv('heartdisease.csv')
    #lt = list(map(float, input("Enter 13 values: ").split()))
    '''max_age = float(df.iloc[1:,0].max())
    print(max_age)
    a = float(df.iloc[1:,1].max())
    print(a)
    b= float(df.iloc[1:,2].max())
    print(b)
```

```
    c = float(df.iloc[1:,3].max())
    d = float(df.iloc[1:,4].max())
    e= float(df.iloc[1:,5].max())
    f = float(df.iloc[1:,6].max())
    g = float(df.iloc[1:,7].max())
    h = float(df.iloc[1:,8].max())
    i = float(df.iloc[1:,9].max())
    j = float(df.iloc[1:,10].max())
    k =float(df.iloc[1:,11].max())
    l = float(df.iloc[1:,12].max())

    print("hi",c)
    print(d)
    print(e,f,g,h,i,j,k,l)

    age1=lt[0]
    a1=lt[1]
    b1=lt[2]
    c1=lt[3]
    d1=lt[4]
    e1=lt[5]
    f1=lt[6]
    g1=lt[7]
    h1=lt[8]
    i1=lt[9]
    j1=lt[10]
    k1=lt[11]
    l1=lt[12]'''
    df = pd.read_csv('heartdisease.csv', header = None)

    training_x=df.iloc[1:df.shape[0],0:13]
    #print(training_set)

    training_y=df.iloc[1:df.shape[0],13:14]
    #print(testing_set)


    # converting dataframe into arrays
    x=np.array(training_x)
    y=np.array(training_y)

    #  load dataset into X and y (features and labels)
# X should contain the numerical features, and y should contain the class labels

    # Apply feature scaling to normalize the features between 0 and 1
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(x)
    #print(X_scaled)
    # Create a copy of the scaled features and labels
```

```
X_augmented = X_scaled.copy()
y_augmented = y.copy()
print(X_scaled[5])
# Define the range for random scaling
scale_range = (0.8, 1.2)

# Apply random scaling augmentation
for i in range(len(x)):
    scale_factor = np.random.uniform(*scale_range)
    X_augmented[i] = X_scaled[i] * scale_factor

# Concatenate the augmented data with the original data
X_combined = np.concatenate((X_scaled, X_augmented), axis=0)
y_combined = np.concatenate((y, y_augmented), axis=0)




a=cardio_age.get("1.0", "end-1c")
age1=float(a)

if(op_sex.get()=='male'):
    cardio_sex='1'
else:
    cardio_sex='0'



if(op_cp.get()=='Atypical Angina'):
    cardio_cp='1'
elif(op_cp.get()=='Typical Angina'):
    cardio_cp='2'
elif(op_cp.get()=='Asymptotic'):
    cardio_cp='3'
elif(op_cp.get()=='Symptotic'):
    cardio_cp='4'
else:
    cardio_cp='2'

#crbp=cardio_rbp.get("1.0", "end-1c")
#cchol=cardio_chol.get("1.0", "end-1c")

if(op_fbs.get()=='<120'):
    cardio_fbs='0'
elif(op_fbs.get()=='>120'):
    cardio_fbs='1'
else:
    cardio_fbs='0'
```

```
if(op_recg.get()=='Normal'):
    cardio_recg='0'
elif(op_recg.get()=='ST-T wave abnormalities'):
    cardio_recg='1'
elif(op_recg.get()=='left ventricular hypertrophy'):
    cardio_recg='2'
else:
    cardio_recg='0'

#cthal=cardio_thl.get("1.0", "end-1c")

if(op_exang.get()=='No'):
    cardio_exang='0'
elif (op_exang.get()=='Yes'):
    cardio_exang='1'
else:
    cardio_exang='0'
colp=cardio_olp.get("1.0", "end-1c")
if(op_slope.get()=='Unsloping'):
    cardio_slope='0'
elif(op_slope.get()=='Flat'):
    cardio_slope='1'
elif(op_slope.get()=='Down sloping'):
    cardio_slope='2'
else:
    cardio_slope='1'

if(op_thal.get()=='Normal'):
    cardio_thal='3'
elif(op_thal.get()=='Fixed'):
    cardio_thal='6'
elif(op_thal.get()=='Reversible effect'):
    cardio_thal='7'
else:
    cardio_thal='7'

#smoke
if(op_ca.get()=='Very low'):
    cardio_ca='0'
elif(op_ca.get()=='Low'):
    cardio_ca='1'
elif(op_ca.get()=='High'):
    cardio_ca='2'
elif(op_ca.get()=='Very high'):
    cardio_ca='3'
else:
    cardio_ca='0'
```

```
a1=float(cardio_sex)
b1=float(cardio_cp)
e1=float(cardio_fbs)
f1=float(cardio_recg)
h1=float(cardio_exang)
j1=float(cardio_slope)
l1=float(cardio_thal)
k1=float(cardio_ca)

c1 = float(cardio_rbp.get("1.0", "end-1c"))
d1 = float(cardio_chol.get("1.0", "end-1c"))
crbp=cardio_rbp.get("1.0", "end-1c")
cchol=cardio_chol.get("1.0", "end-1c")
cthal=cardio_thl.get("1.0","end-1c")
colp=cardio_olp.get("1.0","end-1c")
cardio_thl_value = cardio_thal.get("1.0", "end-1c")

if cardio_thl_value.strip() != '':

    cardio_thl_value = cardio_thl_value.replace('71 - 202', '').strip()

    numeric_part = ''.join(filter(str.isdigit, cardio_thl_value))
    if numeric_part != '':
        g1 = float(numeric_part)
    else:

        age_converted = float(cardio_age.get("1.0", "end-1c"))
        g1 = 220 - age_converted
else:

    age_converted = float(cardio_age.get("1.0", "end-1c"))
    g1 = 220 - age_converted


cardio_olp_value = cardio_olp.get("1.0", "end-1c")
if cardio_olp_value.strip() != '':

    cardio_olp_value = cardio_olp_value.replace('0 - 6.2', '').strip()

    parts = cardio_olp_value.split()
    if len(parts) > 0:

        numeric_part = ''.join(filter(lambda x: x.isdigit() or x == '.', parts[0]))
        if numeric_part != '':
            i1 = float(numeric_part)
        else:

            i1 = 0.2
    else:
```

```
        i1 = 0.2
else:

    i1 = 0.2



global topass
topass=[]


import pickle
model_loaded=pickle.load(open('model_pred','rb'))
sc = StandardScaler()
combined_list = [age1,a1, b1, c1, d1, e1, f1, g1, h1, i1, j1, k1, l1]
topass=[age1,a1,c1,d1]
print(combined_list)
x=list(x)
lt_transformed=x.append(combined_list)
lt_transformed=np.array(x)
s = sc.fit_transform(lt_transformed)
lt_transformed = lt_transformed.astype('float64')
lt_array = np.array(s[-1]).reshape(1, 13)
m =model_loaded.predict(lt_array)
m = [np.argmax(i) for i in m]
print("Predicted heart disease:", m[0])
p=''
if m[0]==0:
    print("10% risk ")
    p='10% risk'
elif m[0]==1:
    print("30% risk")
    p='30% risk'
elif m[0]==2:
    print("50% risk")
    p='50% risk'
elif m[0]==3:
    print("70% risk")
    p='70% risk'
elif m[0]==4:
    print("90% risk")
    p='90% risk'
print(p)

result =Label(cardio_c,text=p, font=("Arial", 30),bg='#D6D6D6')
result.place(x=1300, y=600)
topass.append(float(m[0]))
print(topass)
```

```
    query =  "select * from user_details where uname = '"+user+"'"
    cur.execute(query)
    ls= cur.fetchall()
    if(len(ls)!=0):
        query2 = "UPDATE user_details SET prediction='"+str(m[0])+"' WHERE uname='"+str(user)+"'"
        cur.execute(query2)
        con.commit()
    else:
    #print(type(user))
        query1 = "INSERT INTO user_details VALUES ('" + str(user) + "','" + str(a) + "','" + str(cardio_sex) +
"','" + str(cardio_cp) + "','" + str(crbp) + "','" + str(cchol) + "','" + str(cardio_fbs) + "','" + str(cardio_recg) +
"','" + str(cthal) + "','" + str(cardio_exang) + "','" + str(colp) + "','" + str(cardio_slope) + "','" + str(cardio_ca) +
"','" + str(cardio_thal) + "','" + str(m[0]) + "')"
        cur.execute(query1)
        con.commit()
```

FINDING SIMILAR USERS

```
def find_similar_users(lt_array):
        df_u = pd.read_csv('users_final - Sheet1.csv')
        #print(df_u)
        global similar_users
        #new_user_features = np.array([new_user_age, new_user_sex, new_user_restbp, new_user_chol,
new_user_num]).reshape(1, -1)
        new_user_features = np.array(lt_array).reshape(1, -1)
        distances, indices = knn.kneighbors(new_user_features)
        similar_users = df_u.iloc[indices[0]]['user_id'].values
        distance = distances[0]
        print("distance:", distance)
        print(similar_users)

        similar_user_indices = indices[0]
        similar_user_distances = distances[0]
        similar_user_features = X_scaled[similar_user_indices]

        new_user_feature_df = pd.DataFrame(new_user_features, columns=['age', 'sex', 'restbp', 'chol',
'num'])
        # Combine the features of similar users and the new user into one DataFrame for plotting
        combined_df = pd.concat([new_user_feature_df] * (len(similar_user_indices) + 1))
        combined_df['User'] = ['New User'] + ['Similar User ' + str(i+1) for i in
range(len(similar_user_indices))]
        plt.figure(figsize=(12, 6))

        for i, feature in enumerate(new_user_feature_df.columns):
            plt.subplot(2, 3, i + 1)
            plt.scatter(new_user_feature_df[feature], np.ones(len(new_user_feature_df)) * 0, c='b', marker='o',
label='New User')
```

```
            plt.scatter(similar_user_features[:, i], np.ones(len(similar_user_features)) * 1, c='r', marker='x',
label='Similar Users')
            plt.yticks([0, 1], ['New User', 'Similar Users'])
            plt.xlabel(feature)
            plt.legend()

        plt.tight_layout()
        return similar_users,distance




        ""df_u = pd.read_csv('users_final - Sheet1.csv')
        print(df_u)
        global similar_users
        new_user_features = np.array([new_user_age, new_user_sex, new_user_restbp, new_user_chol,
new_user_num]).reshape(1, -1)
        distances, indices = knn.kneighbors(new_user_features)

        similar_users = df_u.iloc[indices[0]]['user_id'].values
        distance = distances[0]
        #print("distance:", distance)
        return similar_users,distance

        # Example usage
          # Predicted risk of heart disease for the new user
        new_user_age = 54 # Age of the new user
        new_user_sex=1
        new_user_restbp = 130  # Value of additional attribute 1 for the new user
        new_user_chol = 195 # Value of additional attribute 2 for the new user
        new_user_num = 0'"
```

KNN

```
def knnmodel():
        global X_scaled
        df_u = pd.read_csv('users_final - Sheet1.csv')
        #print(df_u)
        training_x=df_u.iloc[1:df_u.shape[0],1:]
        print(training_x)
        x=np.array(training_x)
        scaler = StandardScaler()
        X_scaled = scaler.fit_transform(x)
        print(X_scaled)
        from sklearn.neighbors import NearestNeighbors
        global X
        global distance
```

```
    # Preprocess the dataset
    # Assuming your dataset is in a pandas DataFrame called 'df' with columns 'userid', 'predicted_risk',
'age', 'attribute1', 'attribute2', ...
    #X = df_u[[ 'age', 'sex', 'restbp','chol','num']].values
    k = 5
    global knn
    #global X_scaled
    # Create the k-NN model and fit the data
    knn = NearestNeighbors(n_neighbors=k)
    knn.fit(X_scaled)
    distance=[]
    ""new_user_age=65
    new_user_sex=1
    new_user_restbp=155
    new_user_chol=280
    new_user_num=3'"
    new_user_age=topass[0]
    new_user_sex=topass[1]
    new_user_restbp=topass[2]
    new_user_chol=topass[3]
    new_user_num=topass[4]
    sc = StandardScaler()
    combined_list = [new_user_age,new_user_sex,new_user_restbp, new_user_chol,new_user_num]
    print(combined_list)
    x=list(x)
    lt_transformed=x.append(combined_list)
    lt_transformed=np.array(x)
    s = sc.fit_transform(lt_transformed)
    lt_transformed = lt_transformed.astype('float64')
    lt_array = np.array(s[-1]).reshape(1, 5)
    print(lt_array)

    similar_users,distance = find_similar_users(lt_array)
    print("Similar users:", similar_users)
    print("distance:", distance)
```

SVD

```
    from surprise import Dataset, Reader
    from surprise import SVD  # Example collaborative filtering model



    dt = pd.read_csv('doctor_rating - Sheet1.csv')
    print(dt)
```

```
reader = Reader(rating_scale=(0, 30))  # Define the rating scale
data = Dataset.load_from_df(dt[['user_id', 'doctor_id', 'rating']], reader)
val=0;
ac=999;

for i in range(500):
  trainset,testset = train_test_split(data, test_size=0.2,random_state=i)
  model = SVD()
  model.fit(trainset)
  predictions = model.test(testset)
  rmse = accuracy.rmse(predictions,verbose=False)
  if(rmse<ac):
    ac=rmse;
    val=i;

print(val)
trainset, testset = train_test_split(data, test_size=0.2,random_state=val)
model = SVD()
model.fit(trainset)
predictions = model.test(testset)
#print("\n",predictions)
rmse = accuracy.rmse(predictions,verbose=False)
mae=accuracy.mae(predictions,verbose=False)
print("RMSE for SVD:" ,rmse)
print("MAE for SVD:" ,mae)



# Step 1: Prepare the Dataset
# Assuming your dataset is in a format like CSV with columns: userid, doctorid, rating, risk
""dt = pd.read_csv('doctor_rating - Sheet1.csv')
print(dt.columns)
reader = Reader(rating_scale=(0, 30))  # Define the rating scale
data = Dataset.load_from_df(dt[['user_id', 'doctor_id', 'rating']], reader)

# Step 2: Data Preprocessing (if required)

# Step 3: Train a Collaborative Filtering Model
model = SVD()  # Choose the collaborative filtering model
trainset = data.build_full_trainset()  # Build the training set
model.fit(trainset)  # Train the model

# Step 4: Determine Similar Users
# Assuming you have the risk of heart disease for the new user
new_user_risk = 3  # Example risk value for the new user
# Find similar users based on their historical ratings and predicted risks
similar_users = dataset[dataset['risk'] == new_user_risk]['userid']"'

# Step 5: Generate Recommendations
```

```
top_n = 5  # Number of doctors to recommend
print("hello",similar_users)
recommended_doctors = []
i=0;
reversed_distance = distance[::-1]
distance=reversed_distance.copy()
print("new distance:", distance)

for user in similar_users:
    predictions = []
    print(distance[i])
    print(i)
    for doctorid in dt['doctor_id'].unique():
        predicted_rating = model.predict(user, doctorid).est
        predicted_rating=predicted_rating*distance[i]

        predictions.append((doctorid, predicted_rating))

    predictions.sort(key=lambda x: x[1], reverse=True)
    #print(predictions) # Sort predictions by rating
    i=i+1
    #top_doctors = [doctorid for doctorid, _ in predictions[:top_n]]
    top_doctors = [(doctor_id, prediction) for doctor_id, prediction in predictions[:top_n]]
    #print(top_doctors)
    recommended_doctors.extend(top_doctors)
    recommended_doctors.sort(key=lambda x: x[1], reverse=True)

# Read the doctor details CSV file
doctor_details = pd.read_csv('doctor_details - Sheet1.csv')

# Function to get doctor name and phone number based on doctor ID
def get_doctor_info(doctor_id):
    print("im in")
    doctor_info = doctor_details[doctor_details['doctor_id'] == doctor_id]
    if not doctor_info.empty:
        return doctor_info.iloc[0]['doctor_name'], doctor_info.iloc[0]['doctor_no']
    else:
        return None, None

# Assuming you have the recommended_doctors list already defined elsewhere in your code

unique_doctor_ids = set()
top_5_doctors = []

# Iterate over the recommended doctors list
for doctor_id, _ in recommended_doctors:
    if doctor_id not in unique_doctor_ids:
        # Get doctor name and phone number
        doctor_name, doctor_no = get_doctor_info(doctor_id)
```

```
            if doctor_name and doctor_no:
                top_5_doctors.append((doctor_id, doctor_name, doctor_no))
                unique_doctor_ids.add(doctor_id)

            # Break the loop after 5 unique doctors are found
            if len(top_5_doctors) == 5:
                break

    # Function to create GUI and display recommended doctors' details
    def display_doctors_details():

        print("im in")
        # Print the recommended doctors
        print("Top 5 Recommended Doctors:")
        for i, (doctor_id, doctor_name, doctor_no) in enumerate(top_5_doctors):
            doc_label = Label(recdoc_c, text=f"Doctor ID: {doctor_id}, Name: {doctor_name}, Phone:
{doctor_no}", font=("Arial", 14), bg='#D6D6D6')
            doc_label.place(x=300, y=400 + i * 30)


    # Call the function to create and display the GUI
    display_doctors_details()

    '''unique_doctor_ids = set()
    counter = 0
    #y=500
    id=[]
    for doctor_id, _ in recommended_doctors:
        if doctor_id not in unique_doctor_ids:
            #doc.place(x=1120,y=y+200)
            id.append(doctor_id)
            print(doctor_id)
            unique_doctor_ids.add(doctor_id)
            counter += 1
            if counter == 5:
                break
    # Print the recommended doctors
    print(recommended_doctors[0:5])
    doc=Label(recdoc_c,text=id[0], font=("Arial", 30),bg='#D6D6D6')
    doc.place(x=900,y=400)
    doc=Label(recdoc_c,text=id[1], font=("Arial", 30),bg='#D6D6D6')
    doc.place(x=1000,y=400)
    doc=Label(recdoc_c,text=id[2], font=("Arial", 30),bg='#D6D6D6')
    doc.place(x=1100,y=400)
    doc=Label(recdoc_c,text=id[3], font=("Arial", 30),bg='#D6D6D6')
    doc.place(x=900,y=550)
    doc=Label(recdoc_c,text=id[4], font=("Arial", 30),bg='#D6D6D6')
    doc.place(x=1000,y=550)'''
def recdoc():
```

```
    global
recdoc_bg,recdoc_doc,recdoc_sub,recdoc_b,recdoc_f,recdoc_c,rec_logout,rec_home,pro_b,recdoc_logo
ut,recdoc_pro,recdoc_home,profpage_f

    cardio_f.destroy()
    login_f.destroy()
    newuser_f.destroy()
    profpage_f.destroy()

    recdoc_f=Frame(root)
    recdoc_f.pack()

    recdoc_bg = PhotoImage(file = 'rec_bg.png')
    recdoc_sub = PhotoImage(file = 'rec_b.png')
    recdoc_logout = PhotoImage(file = 'log_b.png')
    recdoc_home= PhotoImage(file = 'home_b.png')
    recdoc_pro =PhotoImage(file = 'prof_icon.png')

    recdoc_c=Canvas(recdoc_f,height=1080,width=1920,bg='#FFFFFF')
    recdoc_c.create_image(0,0,image=recdoc_bg,anchor=NW)
    recdoc_c.pack(fill = "both",expand=True)

    #recdoc_loc = Text(recdoc_c,width=10,height=1, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
    #recdoc_loc.place(x=169, y=458)

    recdoc_doc = Text(recdoc_c,width=50,height=7, font=('Poppins', 40, 'normal'), bg='#D6D6D6', bd=0)
    recdoc_doc.place(x=202, y=347)

recdoc_b=Button(recdoc_c,image=recdoc_sub,command=knnmodel,borderwidth=2,bg='#018CAB',cursor
='hand1')
    recdoc_b.place(x=202, y=200)


rec_home=Button(recdoc_c,image=recdoc_home,borderwidth=0,bg='#015467',command=cardio,cursor='
hand2')
    rec_home.place(x=1436,y=20)


pro_b=Button(recdoc_c,image=recdoc_pro,borderwidth=0,bg='#013C49',command=profile,cursor='hand2
')
    pro_b.place(x=1606, y=20)


rec_logout=Button(recdoc_c,image=recdoc_logout,borderwidth=0,bg='#013C49',command=login,cursor='
hand2')
    rec_logout.place(x=1800, y=20)
```

PROFILE PAGE

```
def profile():
    global profpage,prof_logout,prof_home,logout_b,profpage_c,rec_home,profpage_f

    cardio_f.destroy()
    login_f.destroy()
    newuser_f.destroy()
    recdoc_f.destroy()

    profpage_f=Frame(root)
    profpage_f.pack()

    profpage = PhotoImage(file = 'profile_bg.png')
    prof_logout = PhotoImage(file = 'log_b.png')
    prof_home= PhotoImage(file = 'home_b.png')
    cardio_n= PhotoImage(file = 'next_sub.png')


    profpage_c=Canvas(profpage_f,height=1080,width=1920,bg='#FFFFFF')
    profpage_c.create_image( 0, 0,image =profpage,anchor=NW)
    profpage_c.pack(fill = "both",expand=True)
    userTable = "user"
    query =  "select * from "+userTable+" where uname = '"+user+"' and password = '"+pass_get+"' "
    cur.execute(query)
    ls= cur.fetchall()
    if(len(ls)!=0):
        name=ls[0][0]
        password=ls[0][1]
        phone=ls[0][2]
        address=ls[0][3]
    user_detailsTable="user_details"
    query1=  "select * from "+user_detailsTable+" where uname = '"+user+"'"
    cur.execute(query1)
    ls1=cur.fetchall()
    if(len(ls1)!=0):
        risk_prediction=ls1[0][14]
    if risk_prediction=='0':
        risk_prediction='10% risk'
    elif risk_prediction=='1':
        risk_prediction='30% risk'
    elif risk_prediction=='2':
        risk_prediction='50% risk'
    elif risk_prediction=='3':
        risk_prediction='70% risk'
    elif risk_prediction=='4':
        risk_prediction='90% risk'
    prof_name =Label(profpage_c,text=name, font=("Arial", 30),bg='#FFFFFF')
    prof_name.place(x=853, y=396)
```

```
    prof_phone=Label(profpage_c,text=phone, font=("Arial", 30),bg='#FFFFFF')
    prof_phone.place(x=849, y=468)
    prof_address=Label(profpage_c,text=address, font=("Arial", 30),bg='#FFFFFF')
    prof_address.place(x=853, y=540)
    prof_risk=Label(profpage_c,text=risk_prediction, font=("Arial", 30),bg='#FFFFFF')
    prof_risk.place(x=853, y=612)


rec_home=Button(profpage_c,image=prof_home,borderwidth=0,bg='#015467',command=cardio,cursor='
hand2')
    rec_home.place(x=1436,y=20)


#home_b=Button(profpage_c,image=prof_home,borderwidth=0,bg='#015467',command=cardio,cursor='h
and2')
    #home_b.place(x=1436,y=25)


logout_b=Button(profpage_c,image=prof_logout,borderwidth=0,bg='#013C49',command=login,cursor='ha
nd2')
    logout_b.place(x=1800, y=25)


loginpage = PhotoImage(file = "login1.png")
login_f=Frame(root)
login_f.pack()

login_f = Frame(root)
homepage_f=Frame(root)
homepage_f.pack()

newuser_f=Frame(root)
newuser_f.pack()

cardio_f=Frame(root)
cardio_f.pack

recdoc_f =Frame(root)
recdoc_f.pack
profpage_f =Frame(root)
profpage_f.pack()

login()

root.mainloop()
```

Appendix C: CO-PO And CO-PSO Mapping

COURSE OUTCOMES:

After completion of the course the student will be able to

SL. NO	DESCRIPTION	Blooms' Taxonomy Level
CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply)	Level 3: Apply

CO-PO AND CO-PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO3
CO1	3	3	3	3		2	2	3	2	2	2	3	2	2	2
CO2	3	3	3	3	3	2		3	2	3	2	3	2	2	2
CO3	3	3	3	3	3	2	2	3	2	2	2	3			2
CO4	2	3	2	2	2			3	3	3	2	3	2	2	2
CO5	3	3	3	2	2	2	2	3	2		2	3	2	2	2

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

MAPPING	LOW/ MEDIUM/ HIGH	JUSTIFICATION
100003/CS622T.1-PO1	**HIGH**	Identify technically and economically feasible problems by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.1-PO2	**HIGH**	Identify technically and economically feasible problems by analysing complex engineering problems reaching substantiated conclusions using first principles of mathematics.
100003/CS622T.1-PO3	**HIGH**	Design solutions for complex engineering problems by identifying technically and economically feasible problems.
100003/CS622T.1-PO4	**HIGH**	Identify technically and economically feasible problems by analysis and interpretation of data.
100003/CS622T.1-PO6	**MEDIUM**	Responsibilities relevant to the professional engineering practice by identifying the problem.
100003/CS622T.1-PO7	**MEDIUM**	Identify technically and economically feasible problems by understanding the impact of the professional engineering solutions.
100003/CS622T.1-PO8	**HIGH**	Apply ethical principles and commit to professional ethics to identify technically and economically feasible problems.
100003/CS622T.1-PO9	**MEDIUM**	Identify technically and economically feasible problems by working as a team.
100003/CS622T.1-PO10	**MEDIUM**	Communicate effectively with the engineering community by identifying technically and economically feasible problems.
100003/CS622T.1-P011	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles by selecting the technically and economically feasible problems.
100003/CS622T.1-PO12	**HIGH**	Identify technically and economically feasible problems for long term learning.
100003/CS622T.1-PSO1	**MEDIUM**	Ability to identify, analyze and design solutions to identify technically and economically feasible problems.
100003/CS622T.1-PSO2	**MEDIUM**	By designing algorithms and applying standard practices in software project development and Identifying technically and economically feasible problems.
100003/CS622T.1-PSO3	**MEDIUM**	Fundamentals of computer science in competitive research can be applied to Identify technically and economically feasible problems.
100003/CS622T.2-PO1	**HIGH**	Identify and survey the relevant by applying the knowledge of mathematics, science, engineering fundamentals.

100003/CS622T.2-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems get familiarized with software development processes.
100003/CS622T.2-PO3	**HIGH**	Design solutions for complex engineering problems and design based on the relevant literature.
100003/CS622T.2-PO4	**HIGH**	Use research-based knowledge including design of experiments based on relevant literature.
100003/CS622T.2-PO5	**HIGH**	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes by using modern tools.
100003/CS622T.2-PO6	**MEDIUM**	Create, select, and apply appropriate techniques, resources, by identifying and surveying the relevant literature.
100003/CS622T.2-PO8	**HIGH**	Apply ethical principles and commit to professional ethics based on the relevant literature.
100003/CS622T.2-PO9	**MEDIUM**	Identify and survey the relevant literature as a team.
100003/CS622T.2-PO10	**HIGH**	Identify and survey the relevant literature for a good communication to the engineering fraternity.
100003/CS622T.2-PO11	**MEDIUM**	Identify and survey the relevant literature to demonstrate knowledge and understanding of engineering and management principles.
100003/CS622T.2-PO12	**HIGH**	Identify and survey the relevant literature for independent and lifelong learning.
100003/CS622T.2-PSO1	**MEDIUM**	Design solutions for complex engineering problems by Identifying and survey the relevant literature.
100003/CS622T.2-PSO2	**MEDIUM**	Identify and survey the relevant literature for acquiring programming efficiency by designing algorithms and applying standard practices.
100003/CS622T.2-PSO3	**MEDIUM**	Identify and survey the relevant literature to apply the fundamentals of computer science in competitive research.
100003/CS622T.3-PO1	**HIGH**	Perform requirement analysis, identify design methodologies by using modern tools & advanced programming techniques and by applying the knowledge of mathematics, science, engineering fundamentals.
100003/CS622T.3-PO2	**HIGH**	Identify, formulate, review research literature for requirement analysis, identify design methodologies and develop adaptable & reusable solutions.

100003/CS622T.3-PO3	**HIGH**	Design solutions for complex engineering problems and perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO4	**HIGH**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.3-PO5	**HIGH**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
100003/CS622T.3-PO6	**MEDIUM**	Perform requirement analysis, identify design methodologies and assess societal, health, safety, legal, and cultural issues.
100003/CS622T.3-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts and Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PO8	**HIGH**	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions by applying ethical principles and commit to professional ethics.
100003/CS622T.3-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.3-PO10	**MEDIUM**	Communicate effectively with the engineering community and with society at large to perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering requirement analysis by identifying design methodologies.
100003/CS622T.3-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and prior to that perform requirement analysis, identify design methodologies.
100003/CS622T.4-PO1	**MEDIUM**	Prepare technical report and deliver presentation by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.4-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by preparing technical report and deliver presentation.

100003/CS622T.4-PO3	**MEDIUM**	Prepare Design solutions for complex engineering problems and create technical report and deliver presentation.
100003/CS622T.4-PO4	**MEDIUM**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions and prepare technical report and deliver presentation.
100003/CS622T.4-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and Prepare technical report and deliver presentation.
100003/CS622T.4-PO8	**HIGH**	Prepare technical report and deliver presentation by applying ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
100003/CS622T.4-PO9	**HIGH**	Prepare technical report and deliver presentation effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.4-PO10	**HIGH**	Communicate effectively with the engineering community and with society at large by prepare technical report and deliver presentation.
100003/CS622T.4-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work by prepare technical report and deliver presentation.
100003/CS622T.4-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by prepare technical report and deliver presentation.
100003/CS622T.4-PSO1	**MEDIUM**	Prepare a technical report and deliver presentation to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas.
100003/CS622T.4-PSO2	**MEDIUM**	To acquire programming efficiency by designing algorithms and applying standard practices in software project development and to prepare technical report and deliver presentation.
100003/CS622T.4-PSO3	**MEDIUM**	To apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs by preparing technical report and deliver presentation.
100003/CS622T.5-PO1	**HIGH**	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.5-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by applying engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PO3	**HIGH**	Apply engineering and management principles to achieve the goal of the project and to design solutions for complex engineering problems and design system components or processes that meet the specified needs.
100003/CS622T.5-PO4	**MEDIUM**	Apply engineering and management principles to achieve the goal of the project and use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.5-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO6	**MEDIUM**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities by applying engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO8	**HIGH**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice and to use the engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO1	**MEDIUM**	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. Apply engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PSO2	**MEDIUM**	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur and apply engineering and management principles to achieve the goal of the project.