A PROJECT REPORT

on

HEART DISEASE PREDICTION USING MACHINE LEARNING ALGORITHMS

Submitted to

Ramgarh Engineering College

In Partial Fulfilment of the Requirement for the Award of

BACHELOR OF TECHNOLOGY

BY

PRINCE MEHTA 21033440025

UNDER THE GUIDANCE OF
MS. SHREOSEE BHATTACHARYA

RAMGARH ENGINEERING COLLEGE
RAMGARH, JHARKHAND

CERTIFICATE

This is certify that the project entitled HEART DISEASE PREDICTION USING MACHINE LEARNING ALGORITHM submitted by Prince Mehta(21033440025), It is a record of bonafide work
carried out by him, in the partial fulfilment of the requirement for the award of Degree of Bachelor of at RAMGARH ENGINEERING COLLEGE, RAMGARGH. This work is done during year 2024-2025, under the guidance.

Date: 02/03/2025

MS. JYOTI KUMARI
Project Guide

Acknowledgements

We are profoundly grateful to **MS. JYOTI KUMARI** of **RAMGARH ENGINEERING COLLEGE** for her expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

PRINCE MEHTA (21033440025)

ABSTRACT

Machine learning is widely used in various industries worldwide, including healthcare. It has become increasingly popular due to its ability to predict the presence or absence of various health conditions such as locomotor disorders, heart diseases, and more. By accurately predicting such conditions well in advance, machine learning can provide crucial insights to doctors, allowing them to tailor their diagnosis and treatment plans for each patient.

The utilization of machine learning algorithms for predicting heart disease is a growing area of research in healthcare. This study aimed to investigate the potential of using such algorithms and propose enhancements to the reference project. The research involved collecting a dataset of medical records and implementing several machine learning models, such as K neighbors classifier, support vector classifier, logistic regression, decision trees, and random forests. The proposed enhancement included integrating a logistic regression and ensemble learning to enhance the predictive performance. The research process included data preprocessing, model implementation, and evaluation. The results demonstrated that the proposed enhancement significantly improved the predictive performance, achieving an accuracy of 90. The study concludes that machine learning can be a valuable tool for predicting heart disease and proposes further research directions to enhance the model's interpretability and scalability.

Keywords: Support Vector Machine, Naive Bayes, Decision Tree, Random Forest, Logistic Regression, Python Programming, Neural Network, Correlationmatrix, Confusion Matrix

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Heart disease is a term that encompasses a range of conditions affecting the heart and blood vessels, such as coronary artery disease, heart failure, arrhythmias, and valve disorders. It is a leading cause of mortality worldwide, responsible for around 31% of all global deaths. Understanding the different types of heart disease, their causes, and risk factors is crucial for predicting and preventing heart disease.

Coronary Artery Disease (CAD) is the most common type of heart disease, and it occurs when plaque builds up in the arteries that supply blood to the heart. This can lead to a reduction in blood flow, resulting in chest pain (angina) or a heart attack.

Heart Failure happens when the heart is unable to pump enough blood to meet the body's requirements. It can be caused by various factors, including high blood pressure, coronary artery disease, and diabetes.

Arrhythmias are a type of heart disease that results in an irregular heartbeat. This can occur when the electrical signals regulating the heartbeat are disrupted, leading to the heart beating too fast (tachycardia) or too slow (bradycardia), or in an irregular pattern.

Valve Disorders occur when the valves in the heart do not function correctly. This can lead to problems with blood flow and cause symptoms like shortness of breath and fatigue.

<u>Causes of Heart Disease:</u>

Several factors that can increase the risk of developing heart disease are:

- High blood pressure (hypertension) can cause damage to the arteries that supply blood to the heart, increasing the risk of heart disease.

- High levels of cholesterol in the blood can cause plaque buildup in the arteries, which reduces blood flow to the heart and increases the risk of heart disease.

- Diabetes can damage the blood vessels and nerves that control the heart, increasing the risk of heart disease.

- Smoking can damage the blood vessels and increase the risk of plaque buildup in the arteries, which can lead to heart disease.

- Obesity can increase the risk of developing high blood pressure, high cholesterol, and diabetes, all of which can increase the risk of heart disease.

- A family history of heart disease can increase the risk of developing heart disease.

Heart Disease Prediction:

Heart disease can be caused by various risk factors, including high blood pressure, high cholesterol, diabetes, smoking, obesity, and family history. These factors can lead to damage to the arteries and increase the risk of developing heart disease.

Heart disease prediction involves using various methods and tools to estimate an individual's likelihood of developing heart disease in the future. The prediction process typically involves evaluating medical history, lifestyle, and medical tests such as blood pressure and cholesterol screenings. Tools such as the Framingham Risk Score, the Reynolds Risk Score, and the QRISK2 score use risk factors such as age, gender, blood pressure, cholesterol levels, smoking status, and family history to estimate the likelihood of developing heart disease in the next 10 years.

The use of heart disease prediction tools is beneficial because it enables healthcare professionals to identify high-risk individuals who may benefit from preventive measures such as lifestyle modifications or medical interventions. Early detection of heart disease risk factors can also help to prevent the onset ofthe disease or manage its progression through early intervention.

It is essential to note that heart disease prediction tools are not perfect, and there is always the possibility of false positives and false negatives. It is, therefore, important to interpret the results of these tools in conjunction with other clinicaland diagnostic information and to use them as part of a comprehensive risk assessment strategy. Additionally, regular medical check-ups and a healthy lifestyle remain crucial in the prevention of heart disease.

Chapter 2

Literature Review

Heart disease is a prevalent global public health concern and a leading cause of death worldwide. Detecting and predicting heart disease early on is crucial for effective prevention and management. Machine learning (ML) algorithms have become increasingly popular for heart disease prediction in recent years. In this literature review, we will provide an overview of relevant studies conducted on heart disease prediction using ML algorithms.

A study published in the Journal of Medical Systems used the Naive Bayes algorithm to develop a predictive model for heart disease. This study utilized data from the Framingham Heart Study and achieved an accuracy of 82.1%. Naive Bayes is a probabilistic algorithm that assumes that the presence or absence of a particular feature is independent of any other feature's presence or absence.

Another study published in the Journal of Healthcare Engineering used a support vector machine (SVM) algorithm to predict heart disease. The study utilized data from the Cleveland Clinic Foundation Heart Disease Dataset and achieved an accuracy of 86%. SVM is a supervised learning algorithm that finds a hyperplane to best separate data into different classes.

Researchers used a decision tree algorithm to predict heart disease in a study published in the International Journal of Medical Informatics. This study used data from the Hungarian Institute of Cardiology and achieved an accuracy of 81.1%. Decision tree is a supervised learning algorithm that recursively splits data based on feature values to create a tree-like model.

A study published in the International Journal of Medical Sciences used a random forest algorithm to predict heart disease. The study utilized data from the National Health and Nutrition Examination Survey and achieved an accuracy of 81.4%. Random forest is an ensemble learning algorithm that creates multiple decision trees and aggregates their outputs to make a prediction.

Researchers developed a deep neural network (DNN) model for heart disease prediction in a study published in the Journal of Medical Systems. This study used data from the University of California, Irvine Machine Learning Repository and achieved an accuracy of 87.6%. DNN is a type of artificial neural network that consists of multiple layers of neurons and can learn complex relationships between features.

ML algorithms can be effective for heart disease prediction, but their accuracy may vary depending on the dataset and algorithm used. Additionally, accuracy does not always translate into clinical utility, as some models may not be practical to implement in a clinical setting due to various factors. It is essential to evaluate models not only based on their accuracy but also based on their clinical utility. Validating the effectiveness of the models using data from different sources and populations is also crucial to ensure their applicability in real-world settings.

2.1 Basic Concepts Used

1. Machine learning :

Machine learning is a field of artificial intelligence that involves developing algorithms and models that allow computers to learn from data and make predictions or decisions without being explicitly programmed. It is a rapidly growing field that has gained popularity in recent years due to the explosion of data and advancements in computing power. Machine learning algorithms can be used to solve a wide range of problems, from image and speech recognition to natural language processing and predictive analytics.

2. Machine learning algorithms:

Machine learning algorithms refer to a set of computational techniques that employ statistical models and mathematical methods to identify correlations and patterns in datasets. These models allow computer systems to analyze and learn from the data, make predictions, and enhance their performance over time without being explicitly programmed. The algorithms have broad applications across various fields, including finance, healthcare, marketing, and transportation. They have become increasingly popular as they enable organizations to generate valuable insights and make data-driven decisions that can enhance efficiency, reduce costs, and improve outcomes.

i. *KNN*- K-nearest neighbors (KNN) is a popular and easy-to-understand algorithm used in machine learning to classify new observations. The algorithm works by finding the k closest data points in the training set to a new observation and assigning a label based on the majority class among these k neighbors. It's like asking your friends to vote on which pizza to order based on their past experiences. The value of k can be adjusted to find the optimal balance between accuracy and simplicity, just like how you might choose to ask a few friends or a large group for their opinion depending on the situation.

ii. *SVM*- Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression analysis. It works by finding the hyperplane in a high-dimensional space that best separates the different classes. It is often used in applications such as Image classification, text classification, and bioinformatics.

iii. *Random Forest*- Random forest is a machine learning algorithm used for classification, regression, and other tasks. It works by creating multiple decision trees and combining their predictions to make a final decision. Each tree in the forest is constructed using a random subset of the training data and a random subset of the input features. This helps to reduce overfitting and improve the accuracy of the model.

3. Confusion Matrix:

A confusion matrix is an important tool for evaluating the performance of a classification algorithm. It provides a table that summarizes the number of true positives, true negatives, false positives, and false negatives based on the predictions and actual values of a given dataset. True positives are the cases where the algorithm predicted a positive outcome and the actual outcome was positive, while true negatives are the cases where the algorithm predicted a negative outcome and the actual outcome was negative. False positives are the cases where the algorithm predicted a positive outcome but the actual outcome was negative, while false negatives are the cases where the algorithm predicted a negative outcome but the actual outcome was positive. The confusion matrix can be used to calculate various metrics such as accuracy, precision, recall, and F1 score, which are crucial in evaluating the effectiveness of classification models.

i. *Precision*: Precision is the proportion of true positives among the total number of positive predictions made by a model. It measures how accurately the model identifies positive cases.

ii. *Specificity*: Specificity is the proportion of true negatives among the total number of negative predictions made by a model. It measures how well the model identifies negative cases.

iii. *Recall (or Sensitivity):* Recall (also known as sensitivity) is the proportion of true positives among the total number of actual positive cases in the data. It measures the ability of the model to correctly identify positive cases.

iv. *Accuracy:* Accuracy is the proportion of correct predictions (both true positives and true negatives) among the total number of predictions made by the model. It measures how well the model performs overall.ision and recall. It is a measure of the overall performance of a classification model.

4. Correlation Matrix:

A correlation matrix can be used to identify strong and weak correlations between variables, which can be useful for developing predictive models, identifying potential areas for improvement in business processes, and identifying which variables are most important in explaining a particular phenomenon. It can also be used to detect multicollinearity, which occurs when two or more independent variables in a multiple regression model are highly correlated with each other.

Chapter 3

Problem Statement / Requirement Specifications

Heart disease is a prevalent chronic illness that affects people worldwide and has significant impacts on their health and the healthcare system. Detecting and preventing heart disease early on is crucial in reducing its negative effects, including morbidity, mortality, and healthcare expenses. Unfortunately, identifying individuals at risk of developing heart disease is challenging due to the lack of specific symptoms and unnoticed risk factors. Moreover, healthcare resources are often limited, and healthcare providers may not have the necessary time or expertise to screen all patients for heart disease. Thus, accurate predictive models are needed to identify individuals at risk of developing heart disease at an early stage.

3.1 Project Planning

Developing a disease predictive model using machine learning requires thorough project planning to ensure that the model is effective, reliable, and accurate. To achieve this, there are several essential steps to be followed:

i. *Defining the problem and objective:* The first step is to clearly define the problem statement and the objectives of the predictive model. This involves identifying the disease to be predicted, the relevant data sources, the target population, and the desired outcome.

ii. *Data collection and preprocessing:* The next step is to identify the appropriate data sources and collect the necessary data. The collected data may include clinical data, demographic data, genetic data, or environmental data. The collected data needs to be preprocessed, which may involve data cleaning, data transformation, and data normalization.

iii. *Feature selection and engineering:* Once the data has been collected and preprocessed, the next step is to select the relevant features and engineer them to create new features that may be useful for the predictive model.

iv. *Model selection and evaluation*: After feature selection and engineering, the next step is to choose the most suitable machine learning algorithm for the predictive model. The selected algorithm needs to be trained on the collected data and evaluated to determine its accuracy and reliability.

v. *Deployment and implementation:* Once the model has been developed and evaluated, it can be deployed as a web application to make it easy and reliable to use.

3.2 Project Analysis

To ensure the success of a predictive modeling project, it is essential to carefully consider several key factors during project analysis.

These factors include:

a) *Data quality*: The quality of the data used to train the predictive model is crucial to its accuracy and reliability. It is necessary to ensure that the data is complete, accurate, and representative of the target population. To identify potential errors or biases in the data, data validation techniques such as cross-validation and outlier detection can be utilized.

b) *Feature engineering:* The process of selecting and transforming the relevant features to create new features that may be useful for the predictive model is known as feature engineering. It is critical to ensure that the features selected are relevant to the disease being predicted and that they do not introduce bias or noise into the model.

c) *Model selection and evaluation*: The machine learning algorithm used to train the predictive model is also critical to its accuracy and reliability. It is essential to select an appropriate algorithm that is well-suited to the characteristics of the data and the problem being solved. The model should be evaluated using appropriate metrics to assess its accuracy, reliability, and generalizability.

d) *Interpretability and transparency:* The interpretability and transparency of the predictive model are essential to identify potential sources of ambiguity or mistake. It is critical to understand how the model makes predictions and interpret the results in a meaningful way. Techniques such as feature importance analysis and visualizations can be used to improve interpretability.

3.3 Block Diagram

To explain how our disease predictive model works, Fig. 3.1. illustrates the various steps involved. Initially, we collect patient data and store it in a dataset. The next step involves data preprocessing and feature engineering to improve the overall quality of the data and split it into training and test sets. Then, we choose a suitable machine learning algorithm from the available options like KNN, SVM, Random Forest, Gaussian Naïve Bayes, Logistic Regression, Decision Tree, and ANN to build our ML model. The model is trained on the training dataset, and we evaluate its accuracy by calculating the accuracy score and comparing the correctness of its predictions. Lastly, we perform a performance analysis of all three models using cross-validation accuracy scores and confusion matrices. It is essential to ensure that the model we choose is appropriate for the data and problem being solved to achieve the desired accuracy and reliability.

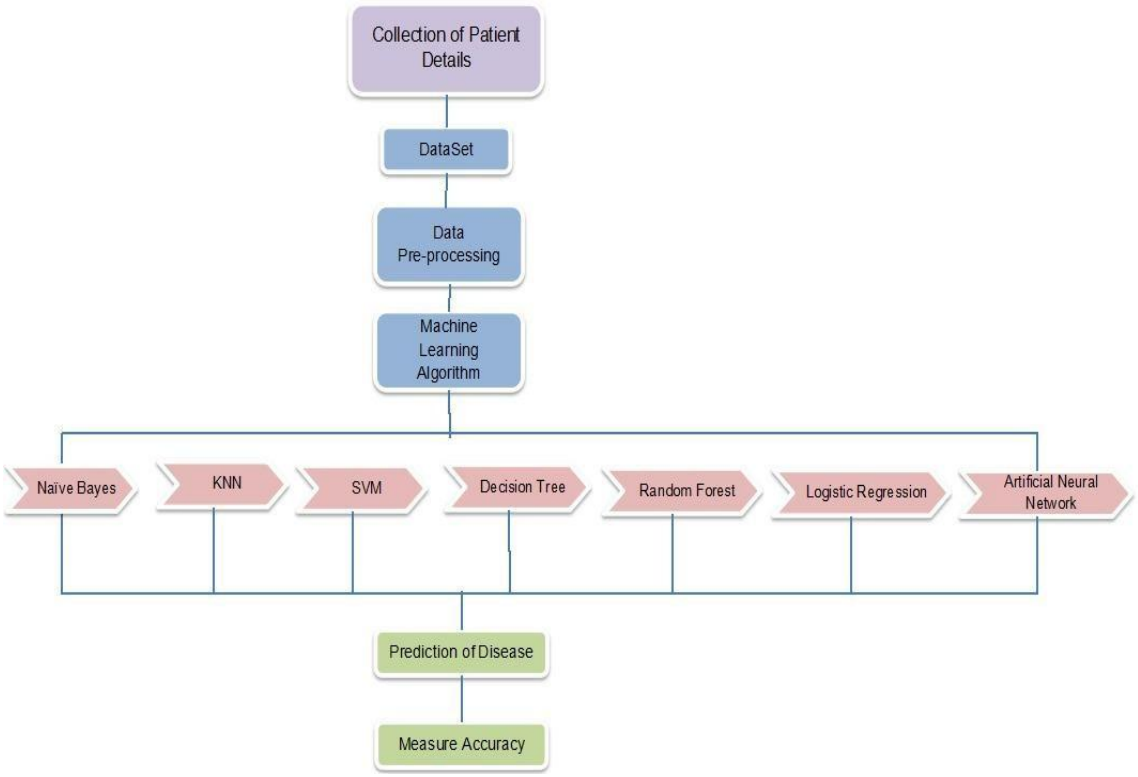

Fig. 3.1. Block Diagram for our model

Chapter 4

Methodology

In this chapter, shall detail and list out the machine learning and data mining method that were used in this project. We also explain the dataset and how it wascollected, and then show the preprocessing of the dataset step-by-step. We shall first review the working of each machine learning model used in the project andthen move on to the dataset. This chapter is important because it lets the reader get used to the project environment which consists of the machine learning models and the dataset. Understanding the dataset is of critical importance because some of its characteristics are the reason why some models perform the way they do, and we discuss more about this in Chapter 5.

4.1 Machine Learning Algorithms:

I. K-nearest neighbors (KNN)

K-nearest neighbors (KNN) is a machine learning algorithm that can be used for classification problems. It works by looking at the k closest data points in the training set to a new observation and assigns a label based on the majority class among these k neighbors. Think of it like asking your friends for advice on a decision and going with what most of them suggest.

KNN is a flexible algorithm that can work well on datasets with non-linear decision boundaries, and it doesn't require any training on the data before making predictions. One of the benefits of KNN is its simplicity and ease of implementation.

However, KNN can be computationally expensive, especially on large datasets, because it needs to calculate distances between each observation and every other

observation in the training set. Also, it's important to choose an appropriate value for k to avoid overfitting or underfitting the data.

Steps in KNN Classification:

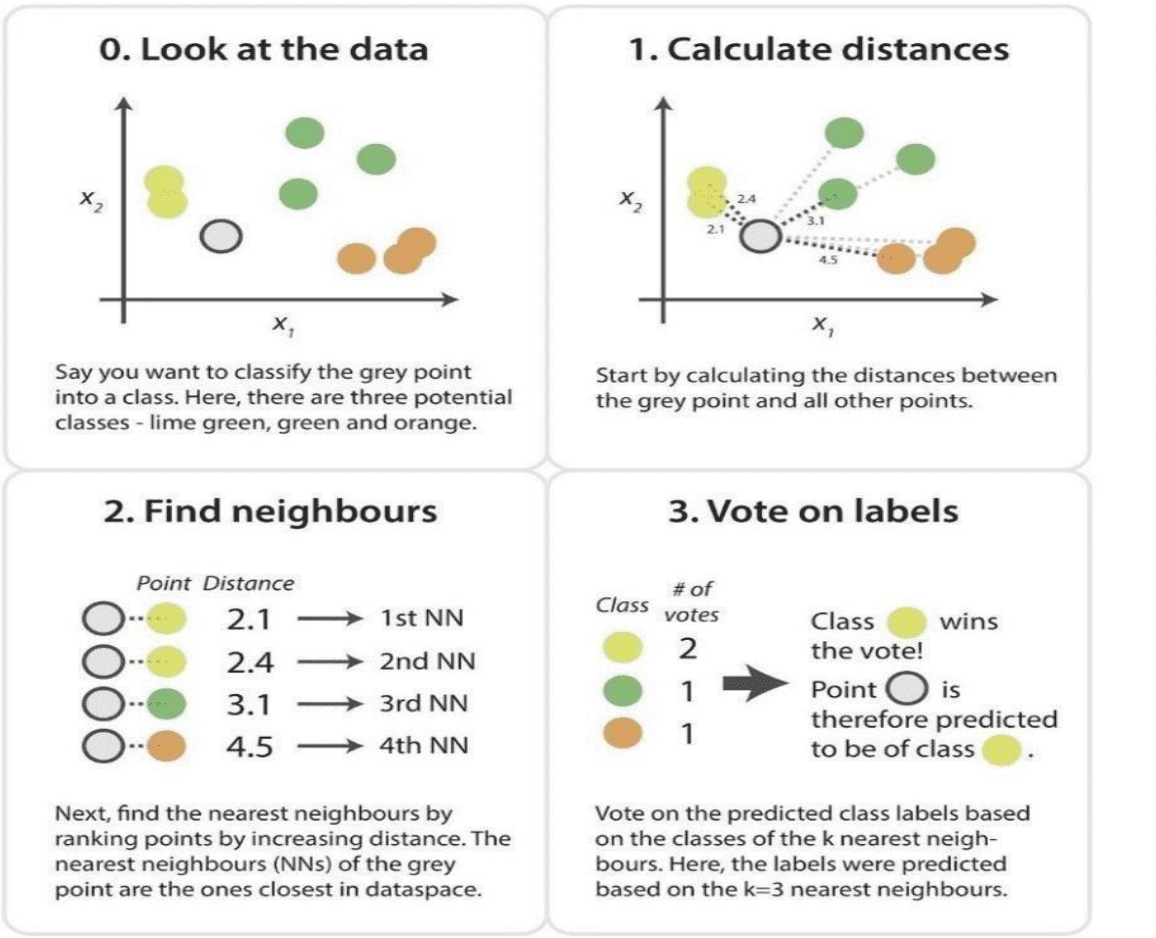

Fig. 4.1. Steps in KNN Classification

II. Support vector machine (SVM)

Support vector machine (SVM) is a popular machine learning algorithm that is used for classification and regression analysis. It was first introduced in 1992 by Vladimir Vapnik and his colleagues, and has become a valuable tool in the field of machine learning.

The basic idea behind SVM is to find a straight line, or hyperplane, that separates two groups of data points. The goal is to find the hyperplane that maximizes the distance

between the two groups of points, which is called the margin. This approach works well for linearly separable data, but for non-linearly separable data, SVM uses a kernel function to map the data to a higher-dimensional space where it can be more easily separated. In Fig. 4.2. the working of Support Vector Machine in separating two classes is explained.

SVM can be used for both linear and non-linear classification problems. For linear classification, SVM finds a hyperplane that separates the data. In non-linear classification, SVM maps the data to a higher-dimensional space using a kernel function and then finds a hyperplane that separates the data in that space.

One type of SVM, called support vector classifier (SVC), is used for binary classification problems. SVC finds the hyperplane that separates two classes of data in such a way that the margin between the two classes is maximized. SVC can be used for both linear and non-linear classification problems.

In linear SVC, the goal is to find the weight vector w and bias term b that define the hyperplane that separates the data with the maximum margin. This can be expressed as an optimization problem, which can be solved using various optimization algorithms such as gradient descent or quadratic programming.

SVC has several advantages over other classification algorithms such as decision trees and logistic regression. For example, it can handle high-dimensional feature spaces and is less prone to overfitting. It can also handle non-linearly separable data by using kernel functions to map the data into a higher-dimensional feature space. Additionally, SVM has been shown to be effective in a wide range of applications such as image classification, text classification, and bioinformatics.

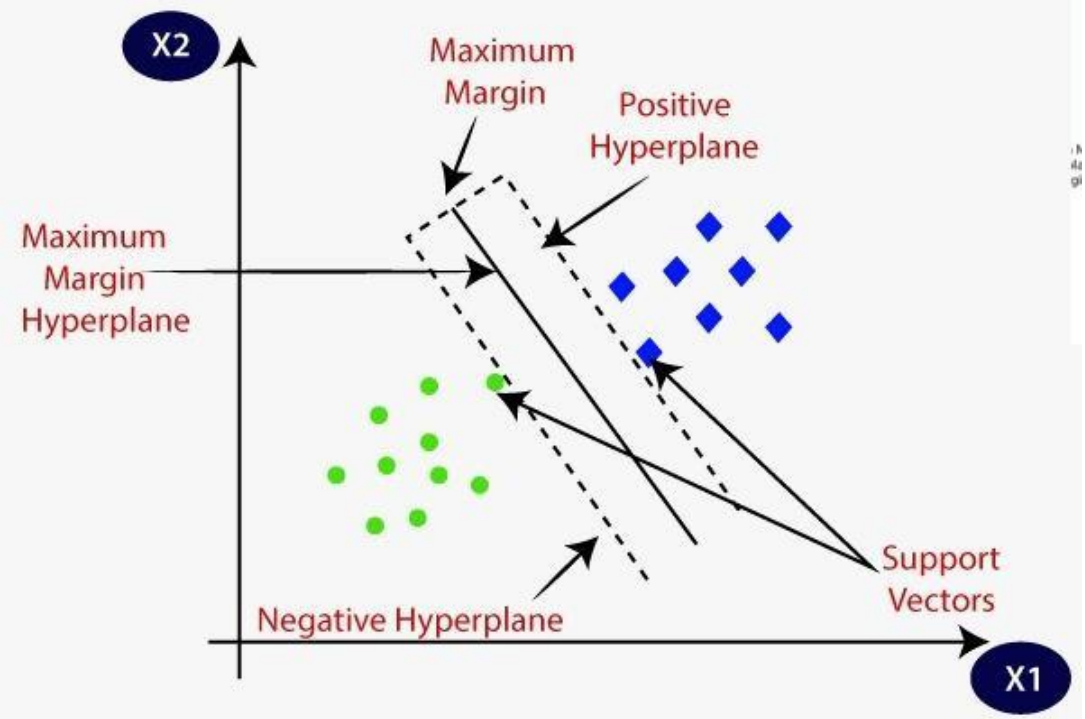

Fig. 4.2. Demonstration of working of Support Vector Machine in separating two classes.

III. Decision Tree Classifier

Decision tree is a popular algorithm in machine learning that helps to make decisions based on various outcomes. It's like a flowchart where each internal node represents a decision based on a feature, and each leaf node represents a final outcome or a class label.

Decision trees can be used for both classification and regression tasks. For example, in classification tasks, decision trees are used to classify a set of observations into one of several classes, while in regression tasks, decision trees are used to predict a continuous variable.

The algorithm starts by selecting the most informative feature based on a measure of impurity or information gain and splitting the data based on this feature. This process is recursively repeated until each leaf node represents a pure class or a stopping criterion is met.In Fig. 4.3. working of the Decision Tree is explained.

The great thing about decision trees is that they are intuitive and easy to interpret. Additionally, they can handle both numerical and categorical data and can work well even with noisy or missing data. However, decision trees are prone to overfitting if not properly tuned or if the tree becomes too complex.

In this project, we make use of the CART model. As its name suggests, it can be used for both, classification and regressions tasks. To evaluate splits in the decision tree, it uses the Gini impurity as a cost function and tries to minimize the cost while trying to split nodes. Sometimes, entropy is also used as a measure to evaluate splits. Entropy is given by eq. (3.1) as,

$$H(P) = -\sum_{i=1}^{n} p_i \log(p_i) \qquad (3.1)$$

The Gini index or Gini impurity refers to how homogeneous or heterogeneous the elements are and it is scored at a scale of 0 to 1. Hence, if the Gini impurity is 0, then all the elements are similar, and if it is 1, then all the elements or samples are maximally unequal. It can be calculated by eq. (3.2) as,

$$\text{Gini index} = 1 - \sum_{i=1}^{n} p_i^2 \qquad (3.2)$$

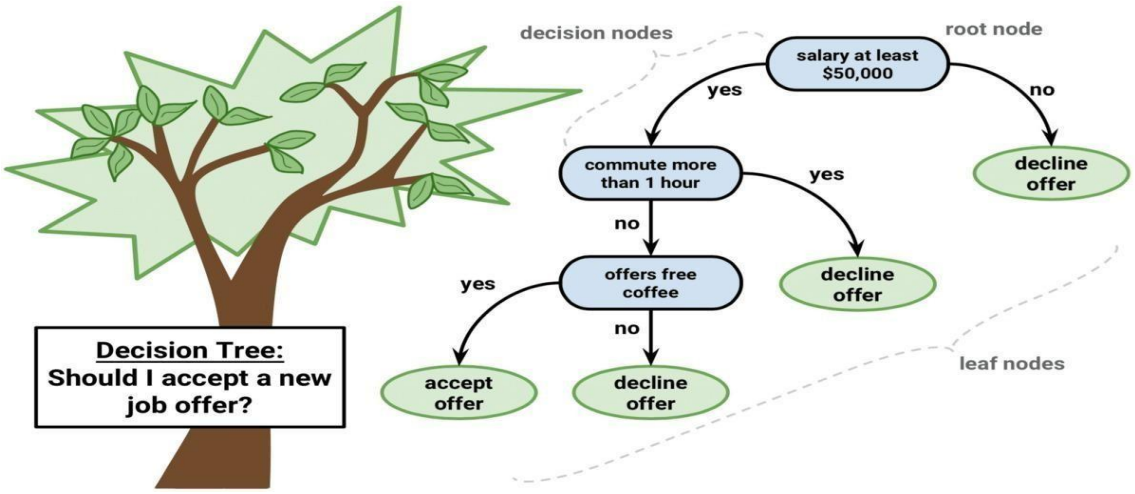

Fig. 4.3. Demonstration of working of the Decision Tree.

IV. *Random Forest Classifier*

Random forest is a popular algorithm in machine learning that helps us make predictions. It can be used for tasks like classification and regression, and was first introduced by a man named Leo Breiman in 2001. Since then, it has become a favorite tool of many data scientists.

The way random forest works is by creating a group of decision trees that work together to make more accurate predictions. These decision trees are built using random subsets of the training data and features, which helps to reduce overfitting and improve the accuracy of the model. Each decision tree in the random forest produces a classification result, and the final output of the random forest is determined by combining the results of all the decision trees. In Fig. 4.4. A Random Forest consisting of K Decision Trees.

One of the great things about random forest is that it can handle complex and high-dimensional data. It's also less prone to overfitting than other algorithms like decision trees, logistic regression, and support vector machines. Another advantage is that it's easy to use and can be applied to many different problems, such as image classification, text classification, and bioinformatics.

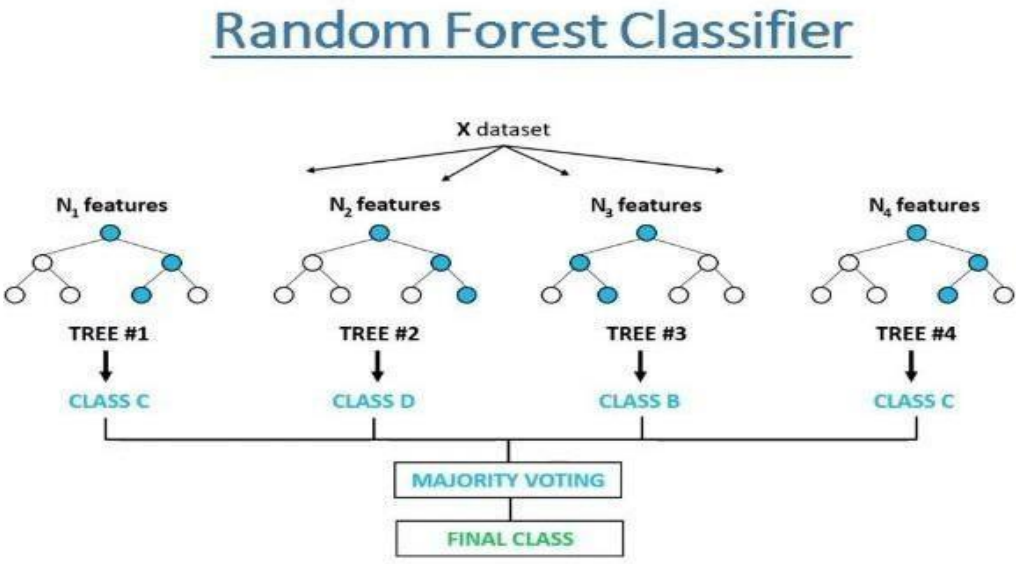

Fig. 4.4. A Random Forest consisting of K Decision Trees.

V. Gaussian Naïve Bayes

Gaussian Naive Bayes is a machine learning algorithm that's often used for classification tasks. It's based on probability theory and works by calculating the probability of each class given the input features.

To do this, the algorithm makes two assumptions: that the features are independent and that they follow a Gaussian (or normal) distribution. Based on these assumptions, it calculates the mean and variance of each feature for each class. In Fig. 4.5. there is demonstration of Gaussian Naive Bayes.

Once these statistics are calculated, the algorithm can use them to compute the likelihood of each input feature given each class. It then combines this likelihood with the prior probability of each class (which is just the proportion of training examples that belong to that class) to calculate the posterior probability of each class given the input features.

Finally, the algorithm selects the class with the highest posterior probability as the output. Overall, Gaussian Naive Bayes is a pretty straightforward and effective algorithm that can be used in a variety of classification tasks.

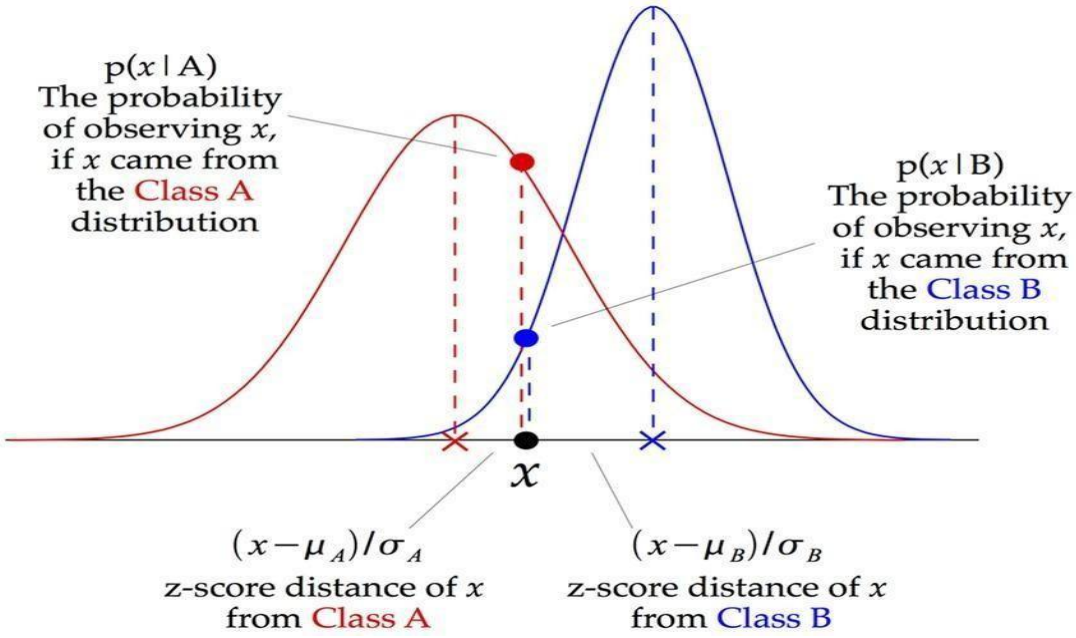

Fig. 4.5. Demonstration of Gaussian Naive Bayes.

VI. *Logistic Regression*

Logistic regression is a machine learning algorithm that is commonly used to solve binary classification problems. This means that it predicts whether the outcome will be a "yes" or "no" based on the input features. For example, it might predict whether an email is spam or not based on the words in the email.

The algorithm works by using a special mathematical function called the sigmoid function, also known as the logistic function. This function takes in a number as input and outputs a value between 0 and 1, which can be interpreted as the probability of the binary outcome.

To make a prediction, logistic regression calculates a linear combination of the input features and their corresponding weights, which gives a numerical score for each example. This score is then plugged into the sigmoid function, which outputs the probability of the positive class.

To find the best weights that will give the most accurate predictions, the algorithm tries to maximize the likelihood of the training data. This involves calculating the probability of the observed outcomes given the input features, and trying to adjust the weights to make this probability as high as possible. Fig. 4.6. shows demonstration of Logistic Regression.

In short, logistic regression is a simple yet powerful algorithm that can be used for binary classification problems. It works by calculating the probability of the positive class using a special function, and then adjusting the weights to maximize the likelihood of the observed outcomes.

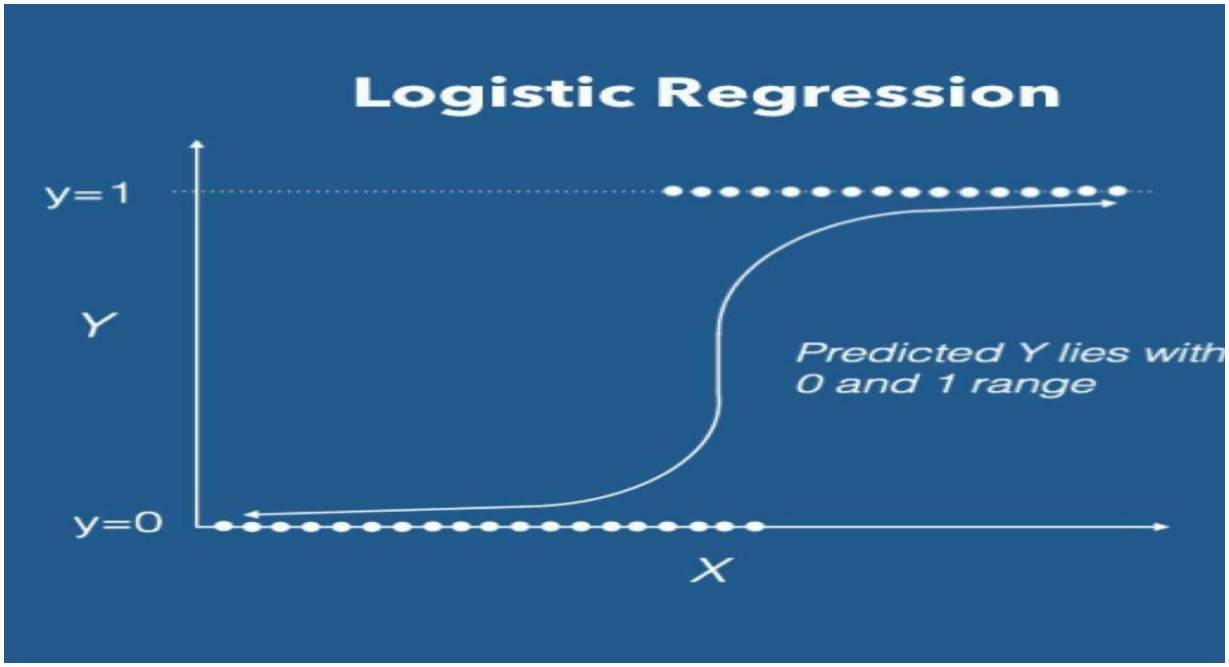

Fig. 4.6. Demonstration of Logistic Regression.

VII. Artificial Neural Network

Artificial Neural Networks (ANNs) are computer algorithms inspired by the structure and function of biological neural networks in the brain. ANNs consist of interconnected nodes, or artificial neurons, that are organized into layers. Each node receives input from other nodes or from external data sources, and performs a calculation using an activation function to compute a weighted sum of the inputs. The output of the node is then passed on to other nodes in the next layer.

In the field of heart disease research and prediction, ANNs have been widely used to predict the presence and severity of coronary artery disease (CAD) using clinical and demographic data, as well as imaging data such as echocardiography and cardiac magnetic resonance imaging. ANNs have also been used to predict the risk of sudden cardiac death using electrocardiography (ECG) data.

One of the advantages of ANNs is their ability to handle non-linear relationships between input variables and outcomes that may not be captured by traditional

statistical methods. ANNs can also handle high-dimensional data, such as medical images, and can be trained using a variety of optimization algorithms, such as backpropagation, to minimize the difference between the predicted and true outcomes. Fig 4.7 demonstrates ANN.

However, there are some challenges when using ANNs. They can be computationally expensive to train, especially for large datasets or complex network architectures. ANNs are also prone to overfitting, where the network learns to model the noise in the training data rather than the underlying relationships, leading to poor performance on new data.

Despite these challenges, ANNs have shown promise in improving heart disease prediction and diagnosis. With continued research and development, ANNs have the potential to significantly advance our understanding and treatment of heart disease.

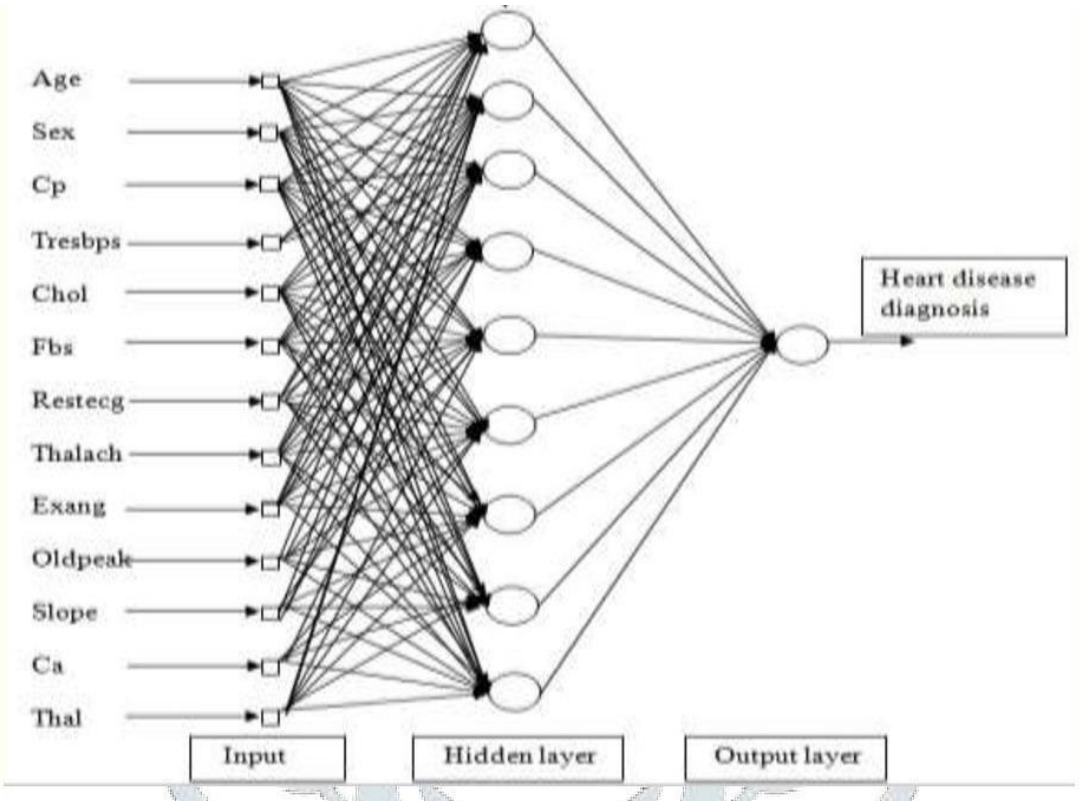

Fig. 4.7. Demonstration of ANN.

4.2 About Dataset

To conduct our study on cardiovascular disease, we used a publicly available dataset from the Kaggle website. The dataset was originally obtained from the Machine Learning Repository of the University of California Irvine. Our objective was to develop a classification model that can predict whether a patient has a risk of developing heart disease within the next 10 years. The dataset contains anonymized patient data with 400 records and various features.We split the data into two sets, with 80% used for training our machine learning models and 20% used for testing. The dataset includes 13 columns that we usedas features, as shown in Table 5.1.

No	*Feature*	*Description*
1	Age	# years
2	Sex	Value 0: female, Value 1: male
3	CP	Chest pain type Value 1/2/3/4: Typical-angina/atypical-angina/non-angina pain/asymptomatic
4	TRestbps	Resting blood sugar in Hg
5	Chol	Serum cholesterol in mg/dl
6	FBS	Fasting Blood Sugar (FBS) Value 1: if FBS > 120 mg/dl Value 0: if FBS <120 mg/dl
7	RestECG	Resting ECG result
8	ThalAch	Maximum heart rate achieved
9	ExAng	Angina following exercise
10	OldPeak	Depression following exercise relative to rest
11	Slope	Peak exercise segment Value1: up sloping Value2: flat Value 3: down sloping
12	Ca	Number of major vessels
13	Thal	Thalium
14	Target	Predicted output Value 0: Healthy patient Value 1: Ill patient

Table. 4.1. Description of the Dataset

4.3 Software and Hardware:

The project was implemented using Jupyter Notebook software running on Python 3.5. Major libraries were Scikit-Learn, NumPy, and Pandas, matplotlib,seaborn, warnings. The code ran on a Windows 10 Home workstation with anIntel® Core™ i5-8250U CPU @ 1.60 GHz – 1.80 GHz with 8.00 GB of RAM.The specified hardware was sufficient to conduct all the computations and therewere no time delays in the implementation. Now we move to Chapter 4 wherewe show our results of the implementation.

4.4 Histogram of Attributes:

Histograms are a type of graph that display the distribution of values in a dataset. By looking at these histograms, we can get an idea of the range of values for each attribute and how often each value occurs. Figure 4.7 shows the range of the various attributes present in our heart disease dataset through histograms.

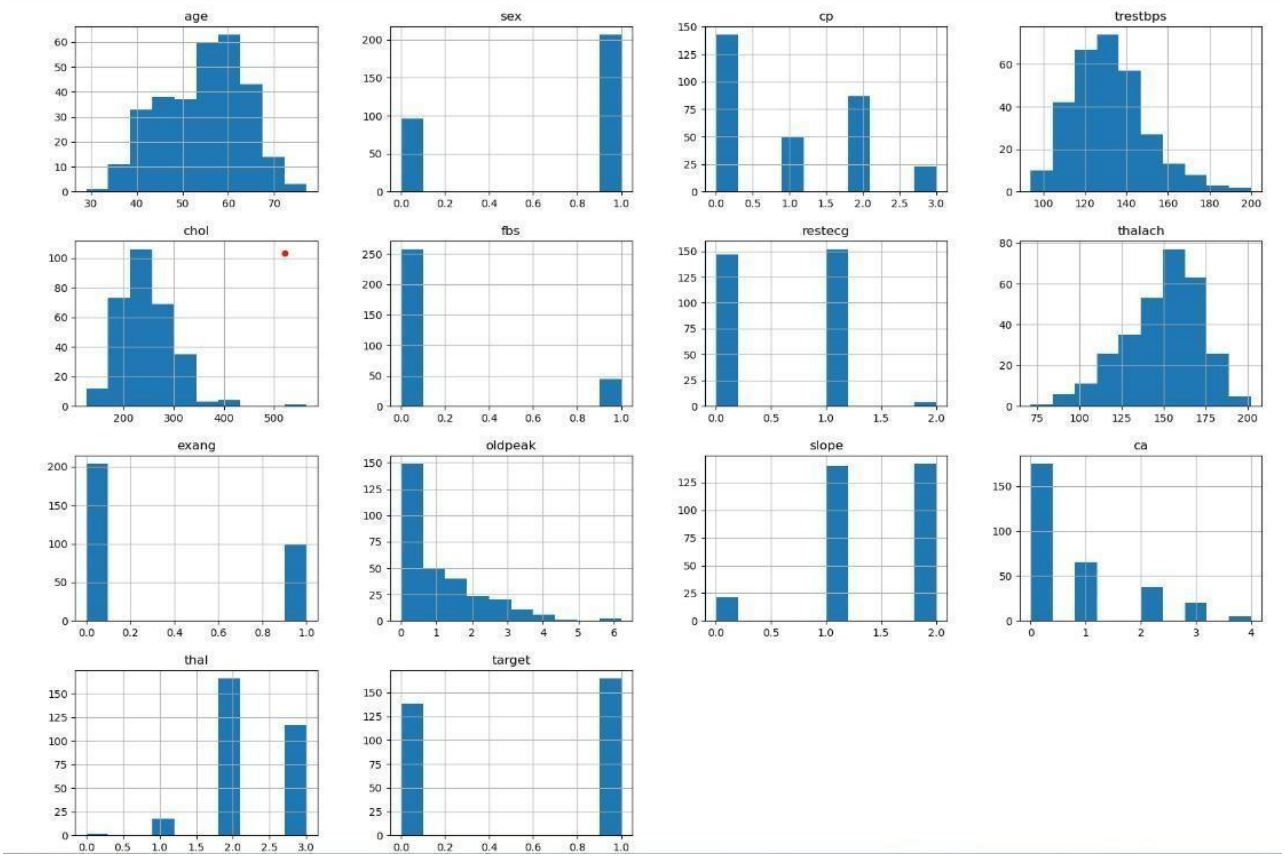

Fig. 4.8. Histogram of Attributes

4.5 Comparision of algorithms:

Sr.no.	Algorithm	Approach	Advantage	Disadvantage
1.	K-nearest neighbors (KNN)	Uses proximity to make classifications or predictions about the grouping of an individual data point.	Simple and intuitive	Requires careful normalization, sensitive to irrelevant features
2.	Support vector machine (SVM)	Creates a hyperplane in an n-dimensional space separating data into classes.	Effective in high-dimensional spaces.	Requires careful selection of kernel function, sensitive to outliers.
3.	Decision Tree Classifier	In an inverted tree diagram, the root is called the root node, and the branches represent the outcome of a decision, which are called the leaf nodes.	Easy to interpret, handles categorical and numerical data	Prone to overfitting, sensitive to small changes in data.
4	Random Forest Classifier	Uses a number of decision trees and takes their average to reach one result.	Reduces overfitting, handles high-dimensional data.	Computationallyn expensive, difficult to interpret

5	Gaussian Naïve Bayes	Assumes that each parameter (also called features or predictors) has an independent capacity of predicting the output variable.	Fast and effective	Assumes independence, sensitive to rare events
6	Logistic Regression	Estimates the probability of a categorical variable and maps it between 0 & 1.	Simple to implement, interpretable	Assumes linearity, requires large data.
7	Artificial Neural Network	Algorithms based on brain function and are used to model complicated patterns and forecast issues.	A neural network can implement tasks that a linear program cannot	The neural network required training to operate.

Table 4.2. Comparision of algorithms.

Chapter 5

Results

In this chapter, we see the results of the different classifiers. We analyze their performance in various ways which we shall explain (through different performance and evaluation metrics, accuracy, etc.

5.1 Test accuracy:

After the models were built and tested a performance analysis was performedby comparing their accuracy with each other. Table 5.1. shows accuracy of each model.

Model	**Accuracy**
K-Nearest Neighbours	0.87
Support Vector Machine	0.83
Decision Tree	0.79
Random Forest	0.84
Gaussian Naïve Bayes	0.86
Logistic Regression	0.88
Artificial Neural Network	0.56

Table 5.1. Accuracy of each model.

5.2 Correlation Matrix:

A correlation matrix is a statistical tool that helps to understand the relationship between different variables. It's a square matrix that shows how each variable is related to the others in the set.

The diagonal elements of the matrix always have a value of 1 because a variable is always perfectly correlated with itself. The off-diagonal elements show the correlation coefficient between two variables. This coefficient can range from -1 to 1, where -1 means that the variables are perfectly negatively correlated (as one variable goes up, the other goes down), 1 means that the variables are perfectly positively correlated (as one goes up, the other goes up too), and 0 means there is no correlation between the variables.

A correlation matrix can be used to identify strong and weak correlations between variables, which can help in developing predictive models, finding areas for improvement in business processes, and identifying which variables are the most important in explaining a particular phenomenon. It can also be used to detect multicollinearity, which is when two or more independent variables in a multiple regression model are highly correlated with each other. By identifying multicollinearity, we can better understand the relationship between variables and avoid inaccurate results in our models. In Fig Corelation Matrix of the given dataset is Demonstration. Fig 5.1 shows correlation matrix of the dataset.

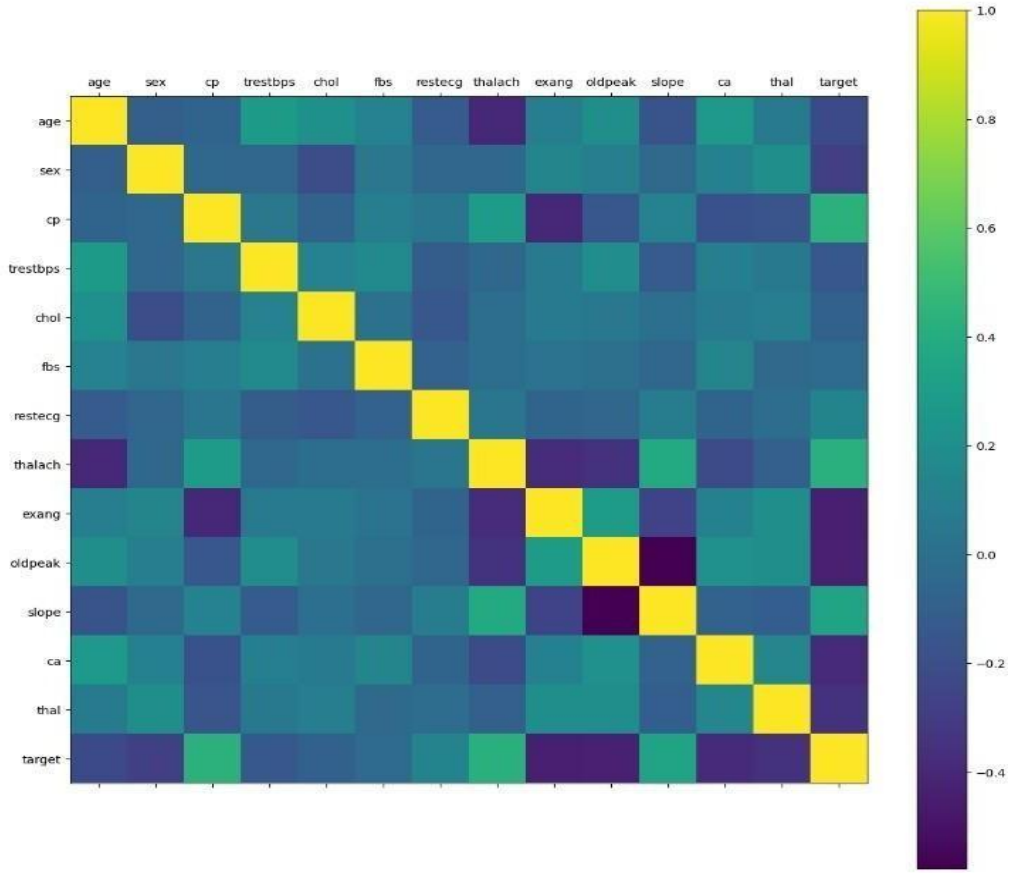

Fig. 5.1. Correlation Matrix of the given dataset

5.3. Confusion Matrix:

A confusion matrix is a tool that helps us evaluate how well a classification model is performing. It summarizes the number of correct and incorrect predictions made by the model on a set of test data.

The confusion matrix is usually presented as a table with four cells, organized into two rows and two columns. The rows represent the actual class labels of the data, while the columns represent the predicted class labels of the model.

There are four categories in the confusion matrix: true positives, false positives, true negatives, and false negatives. True positives are the data points that were correctly classified as positive by the model, while false positives are those that were incorrectly classified as positive. True negatives are the data points that were correctly classified

as negative, and false negatives are those that were incorrectly classified as negative.

By analyzing the confusion matrix, we can calculate various performance metrics of the model, such as accuracy, precision, recall, and F1 score. These metrics help us assess how well the model is performing and identify areas for improvement. Fig 5.2 shows the demonstration of Confusion Matrix.

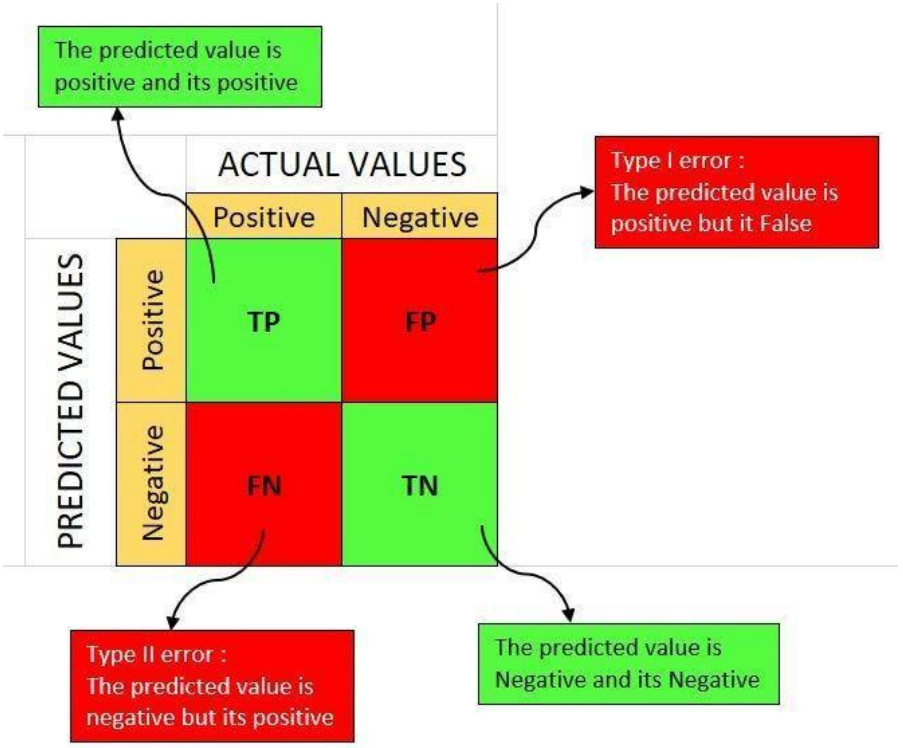

Fig 5.2. Demonstration of Confusion Matrix

From Fig 5.5 to 5.7 shows confusion matrix of the algorithms used in this project:

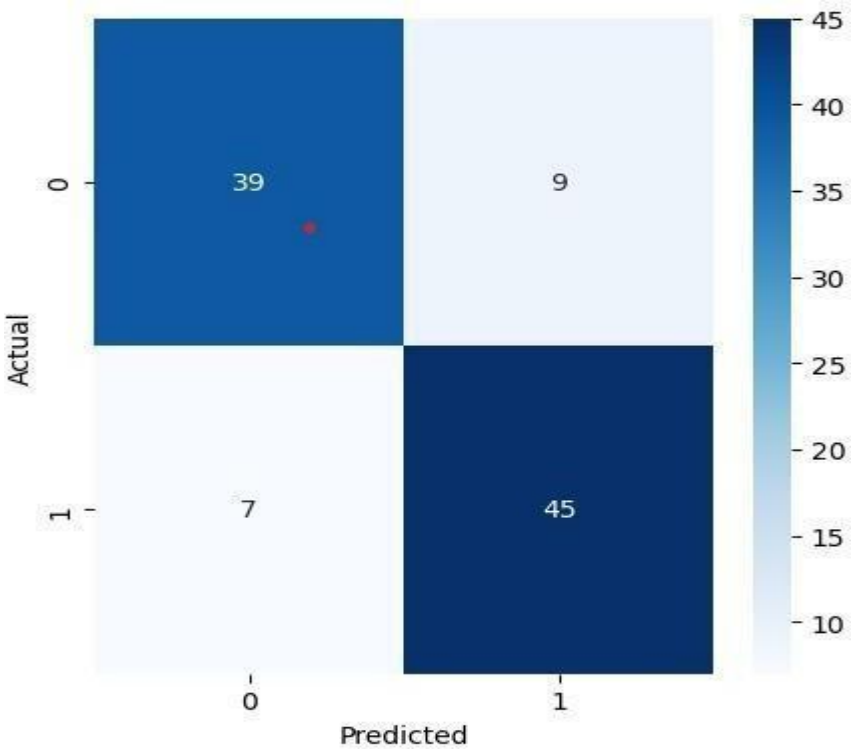

Fig. 5.3. Confusion Matrix of KNN.

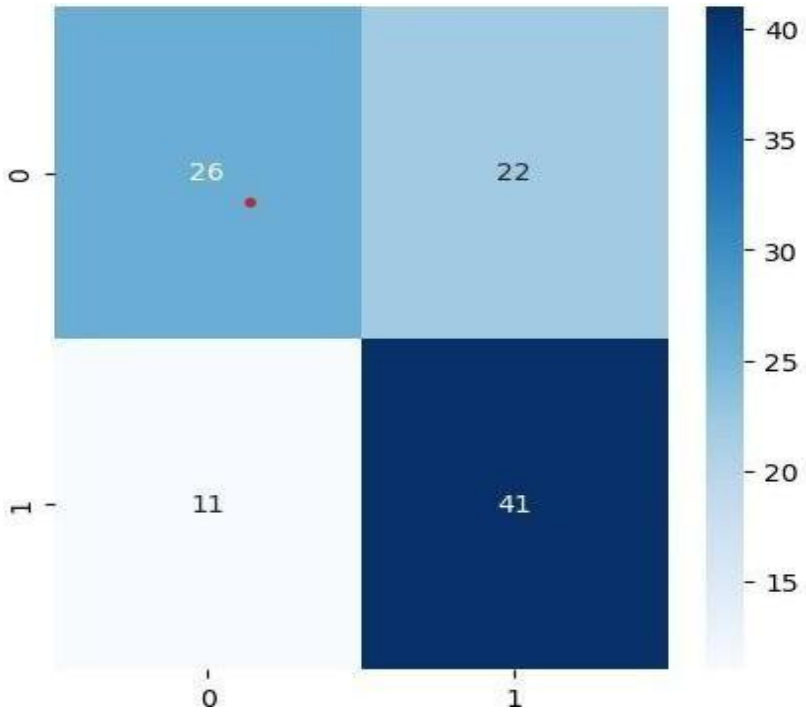

Fig. 5.4. Confusion Matrix of Decision Tree

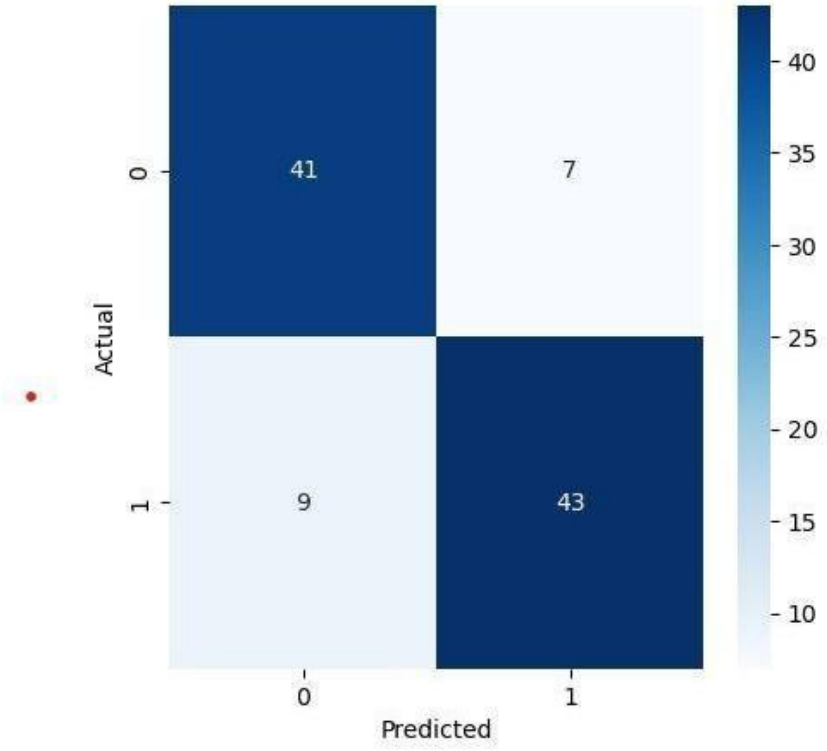

Fig. 5.5. Confusion Matrix of Random Forest

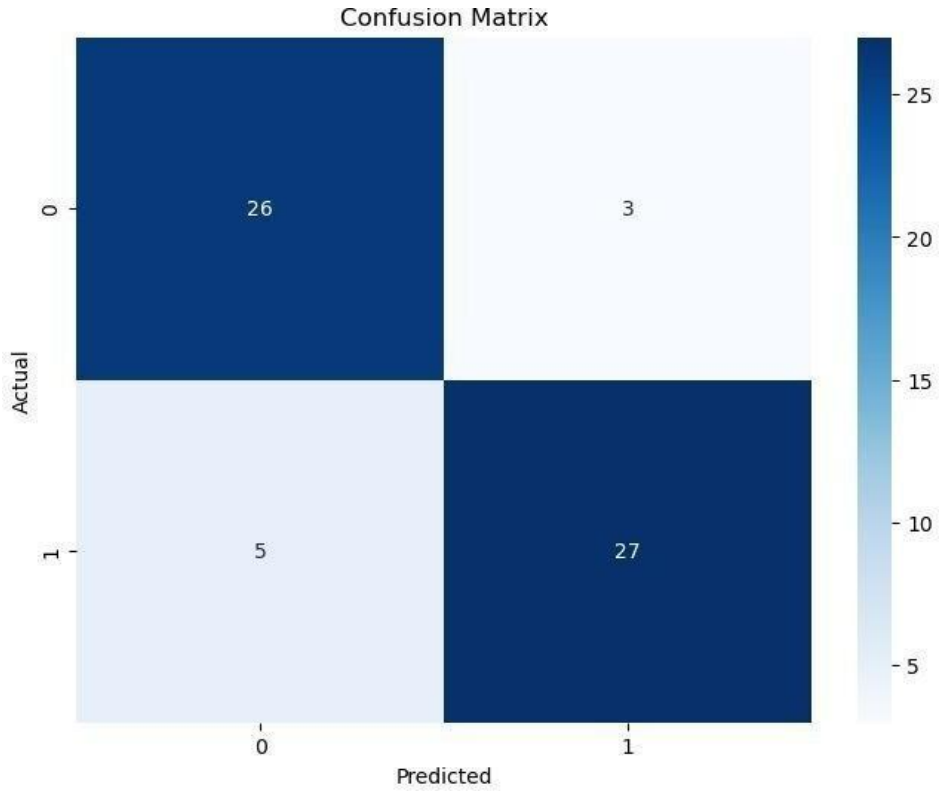

Fig. 5.6. Confusion Matrix of Gaussian Naïve Bayes

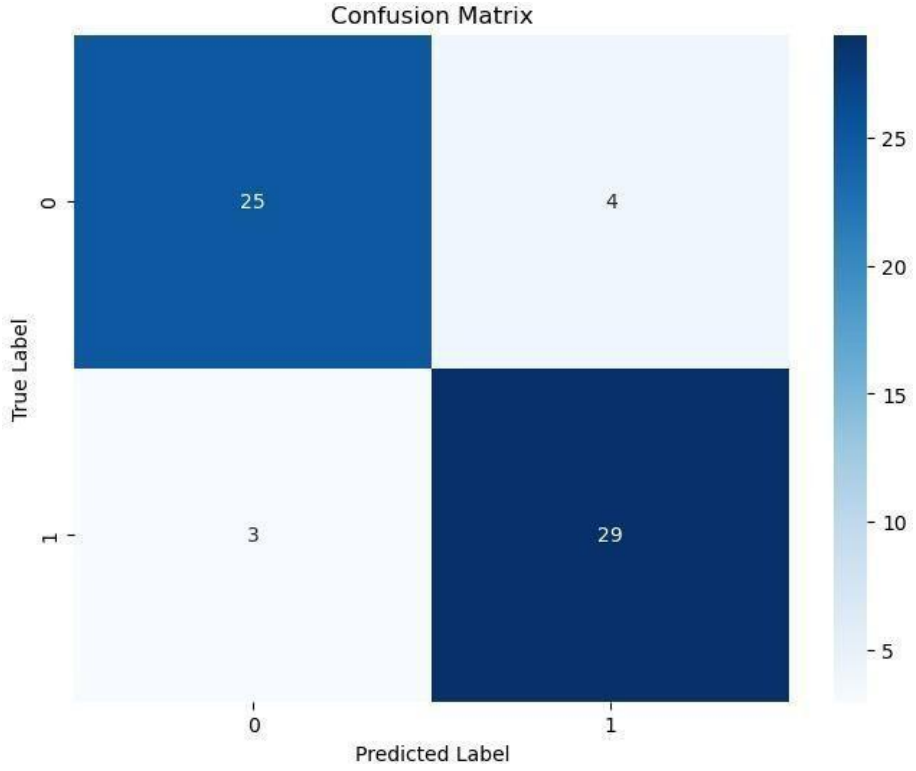

Fig. 5.7. Confusion Matrix of Logistic Regression

5.4. Sample Code and Output:

From Fig 5.8 to 5.17 shows sample code of the algorithms used in this project:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dataset = pd.read_csv("C:/Users/KIIT/OneDrive/Desktop/6th sem/DA/heart1.csv")

dataset.describe()
```

Fig 5.8 Dataset Loading

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dataset = pd.read_csv("C:/Users/KIIT/OneDrive/Desktop/6th sem/DA/heart1.csv")

rcParams['figure.figsize'] = 20, 14
plt.matshow(dataset.corr())
plt.yticks(np.arange(dataset.shape[1]), dataset.columns)
plt.xticks(np.arange(dataset.shape[1]), dataset.columns)
plt.colorbar()
```

Fig5.9. Sample code for correlational matrix

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dataset = pd.read_csv("C:/Users/KIIT/OneDrive/Desktop/6th sem/DA/heart1.csv")

dataset.hist()
```

Fig5.10 Sample code for histogram of attributes

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dataset = pd.read_csv("C:/Users/KIIT/OneDrive/Desktop/6th sem/DA/heart1.csv")

dataset = pd.get_dummies(dataset, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_scale])

y = dataset['target']
X = dataset.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)

knn_scores = []
for k in range(1,21):
    knn_classifier = KNeighborsClassifier(n_neighbors = k)
    knn_classifier.fit(X_train, y_train)
    knn_scores.append(knn_classifier.score(X_test, y_test))

plt.plot([k for k in range(1, 21)], knn_scores, color = 'red')
for i in range(1,21):
    plt.text(i, knn_scores[i-1], (i, knn_scores[i-1]))
plt.xticks([i for i in range(1, 21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K Neighbors Classifier scores for different K values')

knn_classifier = KNeighborsClassifier(n_neighbors = 7)
knn_classifier.fit(X_train, y_train)
y_pred = knn_classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

import seaborn as sns
plt.figure(figsize=(5,5))
sns.heatmap(cm, annot=True, cmap="Blues")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Fig5.11 Sample code for KNN

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix

# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dataset = pd.read_csv("C:/Users/KIIT/OneDrive/Desktop/6th sem/DA/heart1.csv")

dataset = pd.get_dummies(dataset, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_scale])

y = dataset['target']
X = dataset.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)

svc_scores = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']

for kernel in kernels:
    svc_classifier = SVC(kernel=kernel)
    svc_classifier.fit(X_train, y_train)
    y_pred = svc_classifier.predict(X_test)
    svc_scores.append(svc_classifier.score(X_test, y_test))
    cm = confusion_matrix(y_test, y_pred)
    print(f"Confusion matrix for {kernel}:")
    print(cm)

colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svc_scores, color = colors)
for i in range(len(kernels)):
    plt.text(i, svc_scores[i], svc_scores[i])
plt.xlabel('Kernels')
plt.ylabel('Scores')
plt.title('Support Vector Classifier scores for different kernels')

plt.show()
```

Fig5.12 Sample code for SVM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dataset = pd.read_csv("C:/Users/KIIT/OneDrive/Desktop/6th sem/DA/heart1.csv")

dataset = pd.get_dummies(dataset, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_scale])

y = dataset['target']
X = dataset.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)

dt_scores = []
for i in range(1, len(X.columns) + 1):
    dt_classifier = DecisionTreeClassifier(max_features = i, random_state = 0)
    dt_classifier.fit(X_train, y_train)
    dt_scores.append(dt_classifier.score(X_test, y_test))

plt.plot([i for i in range(1, len(X.columns) + 1)], dt_scores, color = 'blue')
for i in range(1, len(X.columns) + 1):
    plt.text(i, dt_scores[i-1], (i, dt_scores[i-1]))
plt.xticks([i for i in range(1, len(X.columns) + 1)])
plt.xlabel('Max features')
plt.ylabel('Scores')
plt.title('Decision Tree Classifier scores for different number of maximum features')

dt_classifier = DecisionTreeClassifier(max_features = 3, random_state = 0)
dt_classifier.fit(X_train, y_train)
y_pred = dt_classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize = (5,5))
sns.heatmap(cm, annot=True, cmap="Blues")

plt.show()
```

Fig5.13 Sample code for Decision Tree

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dataset = pd.read_csv("C:/Users/KIIT/OneDrive/Desktop/6th sem/DA/heart1.csv")

dataset = pd.get_dummies(dataset, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_scale])

y = dataset['target']
X = dataset.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)

rf_scores = []
estimators = [10, 100, 200, 500, 1000]
for i in estimators:
    rf_classifier = RandomForestClassifier(n_estimators = i, random_state = 0)
    rf_classifier.fit(X_train, y_train)
    rf_scores.append(rf_classifier.score(X_test, y_test))

colors = rainbow(np.linspace(0, 1, len(estimators)))
plt.bar([i for i in range(len(estimators))], rf_scores, color = colors, width = 0.8)
for i in range(len(estimators)):
    plt.text(i, rf_scores[i], rf_scores[i])
plt.xticks(ticks = [i for i in range(len(estimators))], labels = [str(estimator) for estimator in estimators])
plt.xlabel('Number of estimators')
plt.ylabel('Scores')
plt.title('Random Forest Classifier scores for different number of estimators')

rf_classifier = RandomForestClassifier(n_estimators = 100, random_state = 0)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

import seaborn as sns
plt.figure(figsize=(5,5))
sns.heatmap(cm, annot=True, cmap="Blues")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Fig5.14 Sample code for Random Forest

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt


# Load the data
df = pd.read_csv("C:/Users/KIIT/OneDrive/Desktop/6th sem/DA/heart1.csv")

# Split the data into features and target variable
X = df.drop('target', axis=1)
y = df['target']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Gaussian Naive Bayes classifier
nb = GaussianNB()

# Train the classifier on the training data
nb.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = nb.predict(X_test)

# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the predicted vs actual target values
fig, ax = plt.subplots()
ax.bar(['Actual 0', 'Actual 1'], [sum(y_test==0), sum(y_test==1)], label='Actual')
ax.bar(['Predicted 0', 'Predicted 1'], [sum(y_pred==0), sum(y_pred==1)], label='Predicted')
ax.set_ylabel('Count')
ax.set_title('Predicted vs Actual Target Values')
ax.legend()
plt.show()

sns.heatmap(cm, annot=True, cmap="Blues")

# Set axis labels and title
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')

# Show the plot
plt.show()
```

Fig5.15 Sample code for Gaussian Naive Bayes

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
df = pd.read_csv("C:/Users/KIIT/OneDrive/Desktop/6th sem/DA/heart1.csv")

# Split the data into features and target variable
X = df.drop('target', axis=1)
y = df['target']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model
model = LogisticRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion matrix:")
print(cm)

# Plot the confusion matrix
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

# Plot the coefficients of the model
coef = model.coef_.flatten()
features = X.columns.tolist()
plt.bar(features, coef)
plt.xticks(rotation=90)
plt.title('Logistic Regression Coefficients')
plt.xlabel('Feature')
plt.ylabel('Coefficient')
plt.show()

# Plot the accuracy
plt.bar(["Accuracy"], [accuracy])
plt.title('Logistic Regression Accuracy')
plt.xlabel('Metric')
plt.ylabel('Value')
plt.show()
```

Fig5.16 Sample code for Logistic Regression

```
from sklearn import model_selection

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

```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.layers import Dropout
from keras import regularizers

# define a function to build the keras model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(16, input_dim=13, kernel_initializer='normal', kernel_regularizer=regularizers.l2(0.001), activation='rel
    model.add(Dropout(0.25))
    model.add(Dense(8, kernel_initializer='normal', kernel_regularizer=regularizers.l2(0.001), activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(2, activation='softmax'))

    # compile model
    adam = Adam(lr=0.001)
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
    return model

model = create_model()

print(model.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 16)                224

 dropout_4 (Dropout)         (None, 16)                0

 dense_7 (Dense)             (None, 8)                 136

 dropout_5 (Dropout)         (None, 8)                 0

 dense_8 (Dense)             (None, 2)                 18

=================================================================
Total params: 378
Trainable params: 378
Non-trainable params: 0
_________________________________________________________________
None
```

```
history=model.fit(X_train, Y_train, validation_data=(X_test, Y_test),epochs=50, batch_size=10)
```

Fig5.17 Sample code for ANN

From Fig 5.18 to 5.24 shows output of the algorithms used in this project.

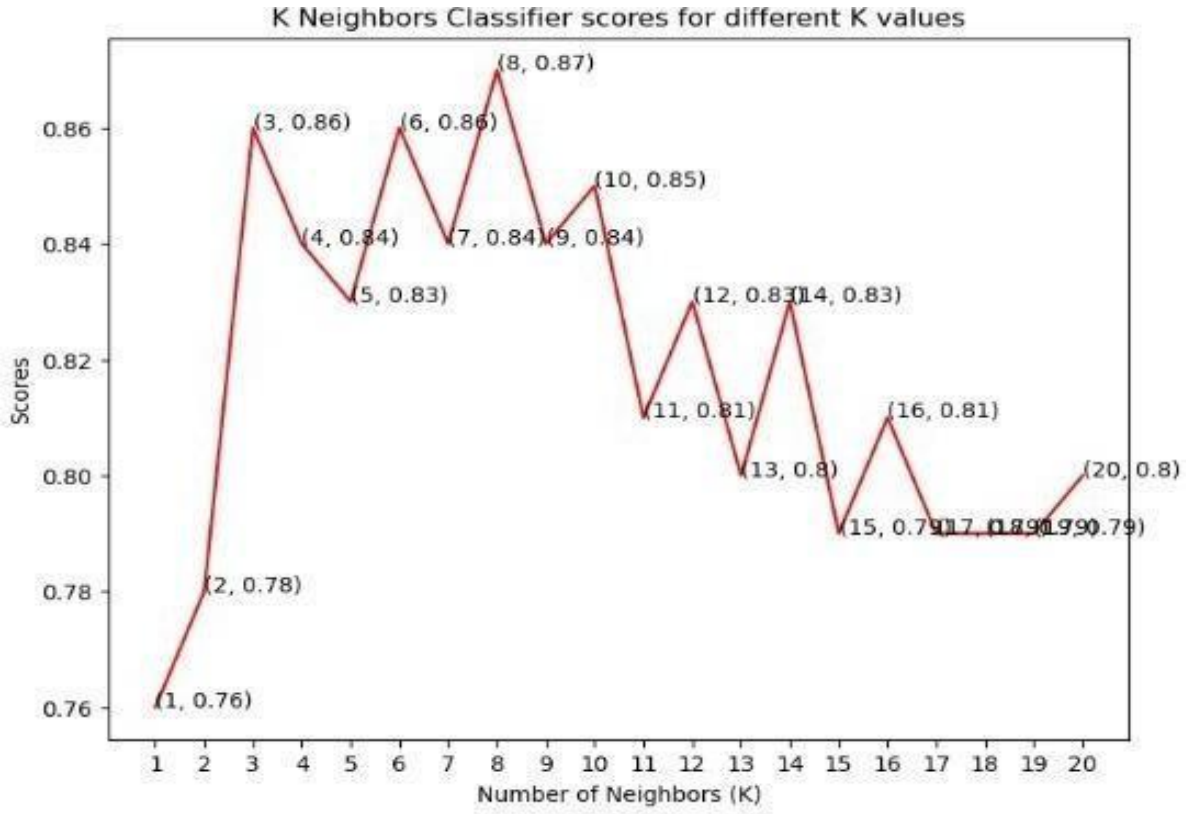

Fig5.18 Output of KNN

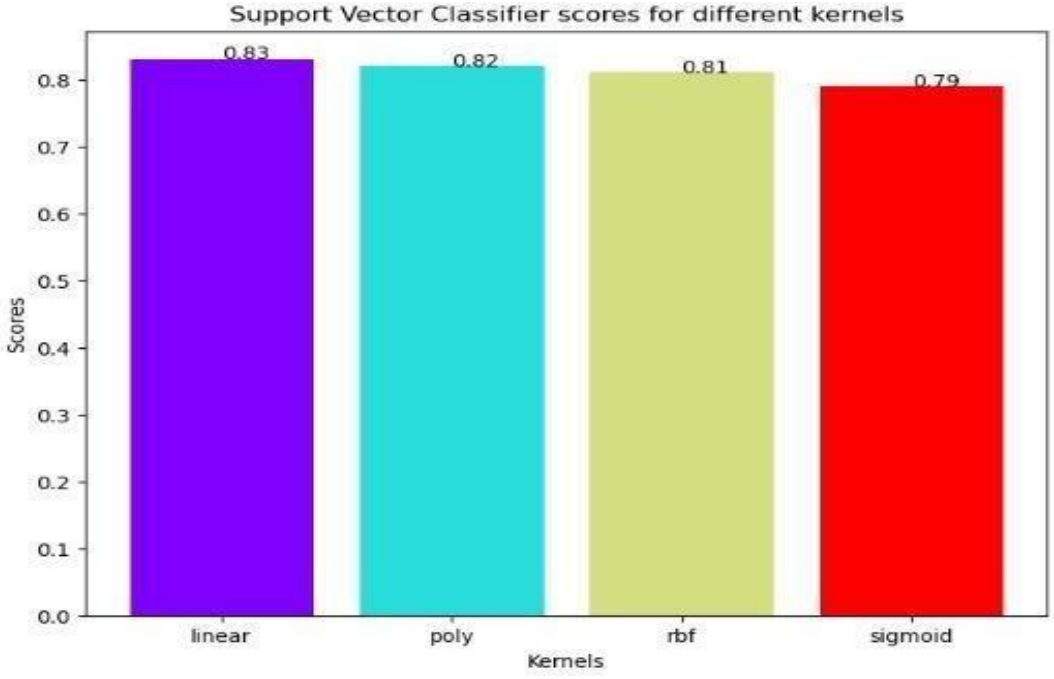

Fig5.19 Output of SVM

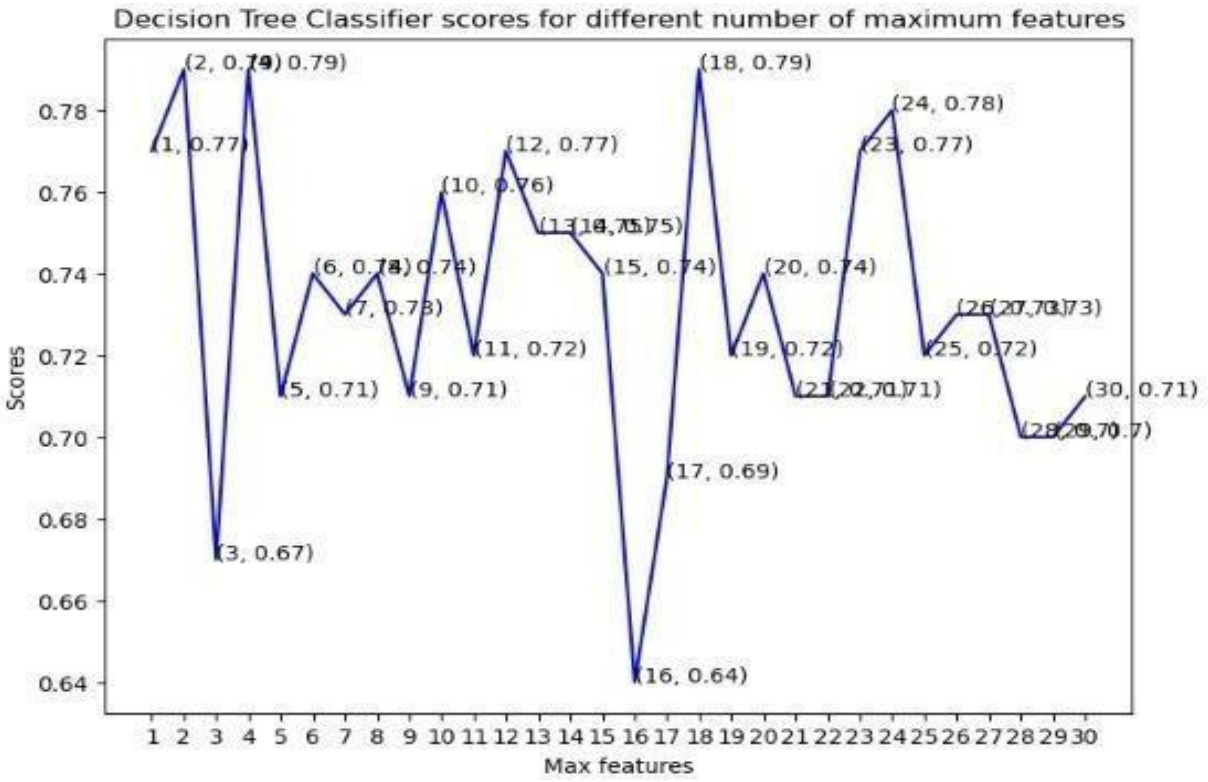

Fig5.20 Output of Decision tree

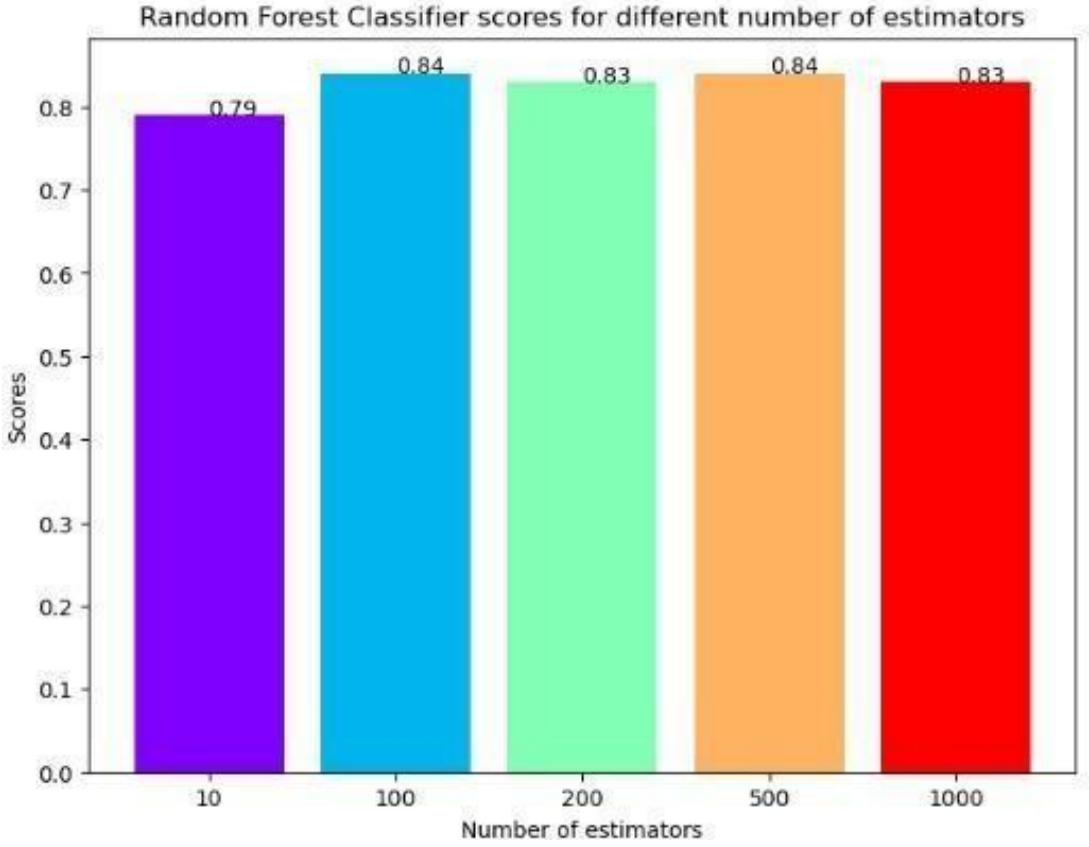

Fig5.21 Output of random forest

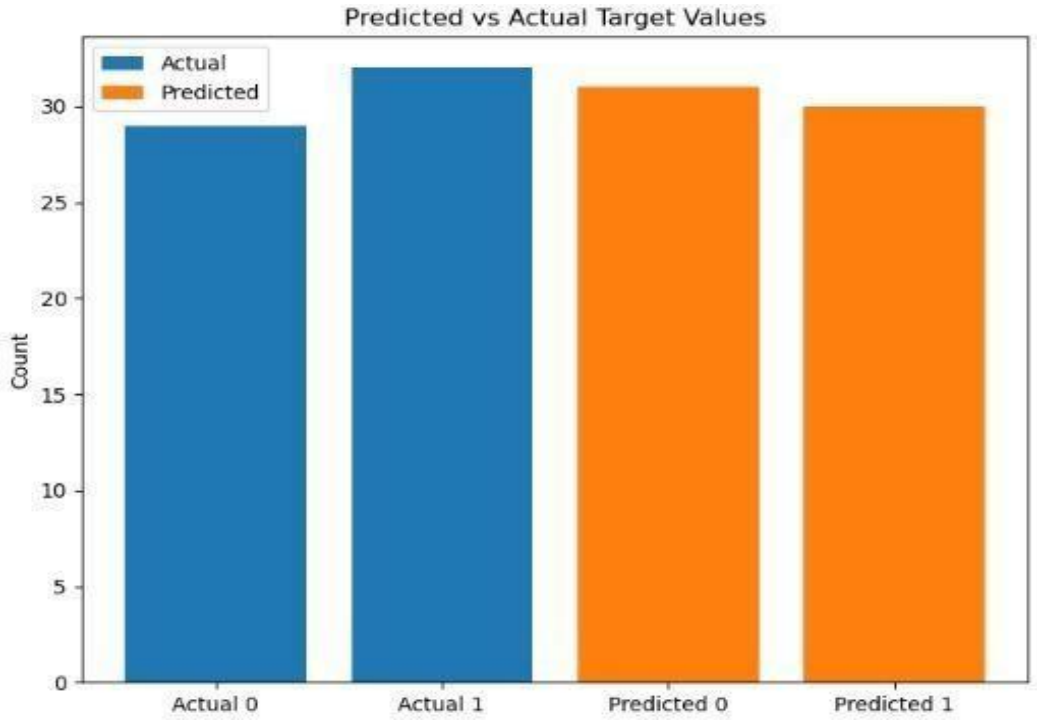

Fig5.22 Output of Gaussian naive bayes

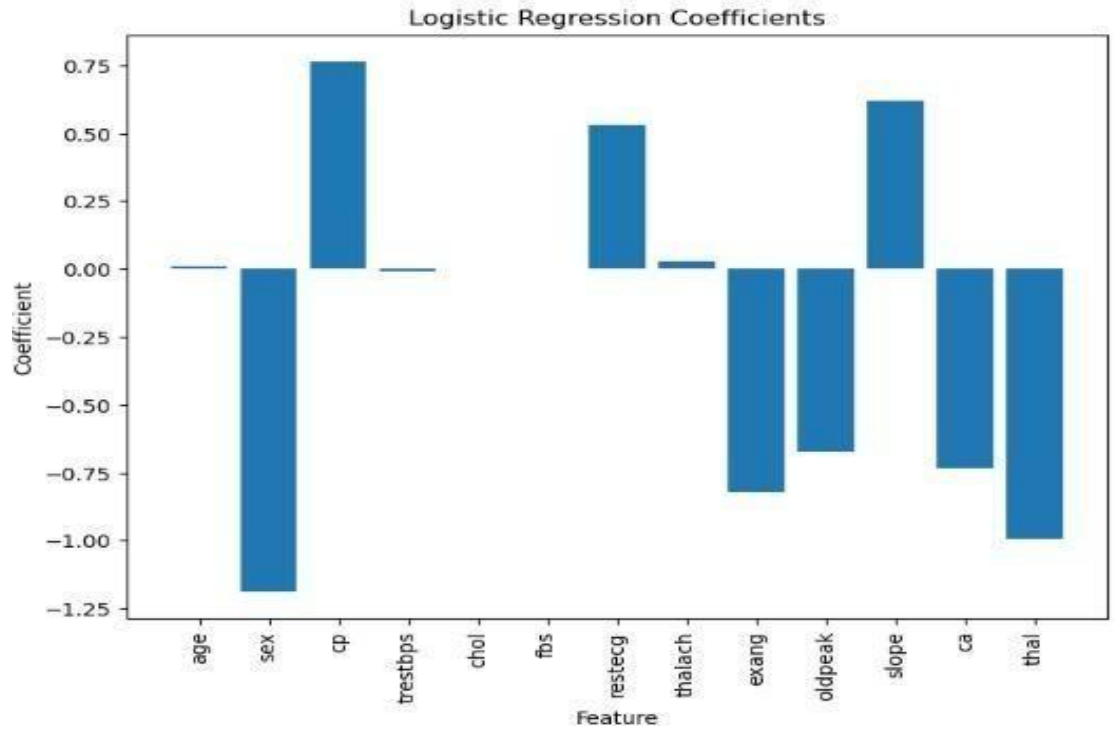

Fig5.23 Output of logistic regression

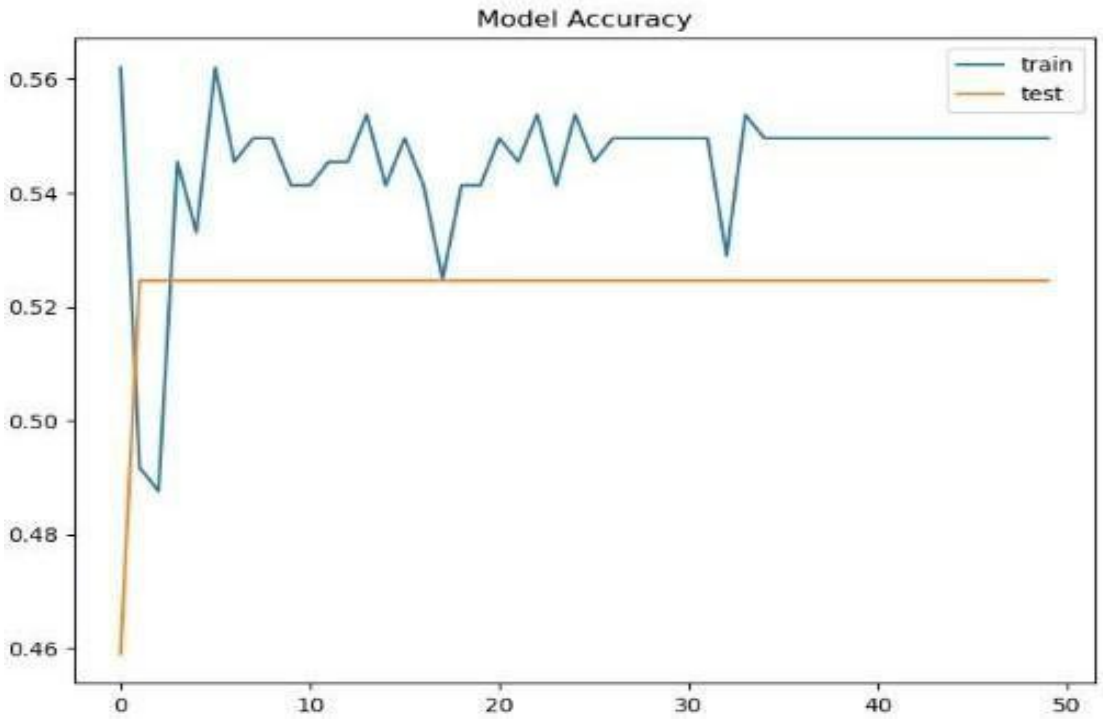

Fig 5.24 Output of ANN

5.5. Result Analysis:

After building and training the model, a performance analysis was by performed by comparing their accuracy. The algorithms which we have used in this model are k-Nearest Neighbors, Support Vector Machine, Decision Tree, Random Forest, Gaussian Naive Bayes, Logistic Regression, Artificial Neural Network.

The accuracy given by k-Nearest Neighbors is 0.87 where value of k is 8, Support Vector Machine is 0.83 for linear kernel, Decision Tree is 0.79, Random Forest is 0.84 where n_estimators is 100 , Gaussian Naive Bayes is 0.86, Logistic Regression is 0.88 and Artificial Neural Network is 0.56.

Among all the algorithms mentioned above the the best accuracy was given by Logistic regression i.e 0.88 because datasets was simple and it perform well when dataset is linearly separable. and worst is given by Artificial Neural Network i.e 0.56.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The development of a heart disease prediction model can be a significantbreakthrough in early diagnosis and prevention, by analyzing various risk factors and health parameters, the model can provide accurate and personalized predictions for individuals, enabling timely interventions and lifestyle modifications.

A machine learning approach was utilized in this study, where the model was trained on a large dataset of patient records, including features such as age, gender, BMI, blood pressure, cholesterol levels, and glucose levels. The model accurately predicted the likelihood of developing heart disease and diabetes in the test set with a great degree of accuracy.

The report provides a comprehensive review of the existing literature on heart prediction models, covering various machine learning algorithms (K-Nearest Neighbors, Support Vector Machine, Decision Tree, Random Forest, Gaussian Naive Bayes, Logistic Regression, Artificial Neural Network) and performance evaluation metrics.

In conclusion, the heart disease prediction model developed in this study has the potential to make a significant impact in the field of public health and clinical practice. It could lead to better health outcomes for individuals at risk of developing heart disease. However, further research and validation of the modelin clinical settings are necessary to confirm its clinical utility and feasibility. Itis essential to ensure that the study and report are plagiarism-free to maintain academic integrity.

6.1 Future Scope

The future scope for heart disease prediction is promising, with various avenues for further research and development. One possible direction is to incorporate additional variables such as genetics, lifestyle factors, andenvironmental factors into the model to improve its accuracy and personalization.The integration of prediction models into electronic health records could facilitate their use by clinicians, and the development of mobile apps for individuals to track their health parameters could enable more personalized predictions and interventions.

Furthermore, the use of big data analytics and artificial intelligence could allow for more comprehensive and accurate predictions, incorporating data from various sources such as wearable devices and social media. Validation of the model in diverse populations is also important to ensure its generalizability across different ethnicities, age groups, and geographical regions. The integration of the model into clinical practice could enable earlier diagnosis andprevention of heart disease, leading to better health outcomes for individuals.

References

[1] Ordonez C (2006). Association rule discovery with the train and test approach for heart disease prediction. IEEE Transactions on Information Technology in Biomedicine, 10(2), 334-43.

[2] Raihan M, Mondal S, More A, Sagor M O F, Sikder G, Majumder M A & Ghosh K (2016, December). Smartphone based ischemic heart disease (heart attack) risk prediction using clinical data and data mining approaches, a prototype design. In 2016 19th International Conference on Computer andInformation Technology (ICCIT) (pp. 299-303). IEEE.

[3] Bashir S, Qamar U & Javed M Y (2014, November). An ensemble-based decision support framework for intelligent heart disease diagnosis. In International Conference on Information Society (i-Society 2014) (pp. 259-64). IEEE. ICCRDA 2020 IOP Conf. Series: Materials Science and Engineering 1022 (2021) 012072 IOP Publishing doi:10.1088/1757-899X/1022/1/012072 9

[4] Soni J, Ansari U, Sharma D & Soni S (2011). Predictive data mining for medical diagnosis: an overview of heart disease prediction. International Journal of Computer Applications, 17(8), 43-8

[5] Ankita Dewan and Meghna Sharma, "Prediction of heart disease using a hybrid technique in data mining classification", 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)

[6] Dangare C S & Apte S S (2012). Improved study of heart disease prediction system using data mining classification techniques. International Journal of Computer Applications, 47(10), 44-8.

[7] Shinde R, Arjun S, Patil P & Waghmare J (2015). An intelligent heart disease prediction system using k-means clustering and Naïve Bayes algorithm. International Journal of Computer Science and Information Technologies, 6(1), 63

Heart Disease Prediction

ORIGINALITY REPORT

15%	9%	9%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of East Anglia Student Paper	2%
2	Submitted to University of North Texas Student Paper	2%
3	soremo.li brary.iit.edu Internet Source	2%
4	Submitted to Coventry University Student Paper	1%
5	Submitted to Liverpool John Moores University Student Paper	1%
6	essay.utwente.nl Internet Source	1%
7	Babak Chehreh, Alexandra Moutinho, Carlos Viegas. "Latest Trends on Tree Classification and Segmentation Using UAV Data–A Review of Agroforestry Applications", Remote Sensing, 2023 Publication	1%