2024

PROJ 518 FINAL DISSERTATION

HEART DISEASE DIAGNOSIS USING MACHINE LEARNING AND DATA MINING

Presented To

UNIVERSITY OF PLYMOUTH

Presented By

DANIEL TOLUWANI ADELEKE (10892938)

HEART DISEASE DIAGNOSIS USING MACHINE LEARNING AND DATA MINING

by

DANIEL TOLUWANI ADELEKE

Thesis submitted to University of Plymouth
in partial fulfilment of the requirements for the degree of

MSc Data Science & Business Analytics

University of Plymouth
Faculty of Science & Engineering

Supervised by

Dr. Raphael Stuhlmeier

September 2024

Copyright Statement

Abstract

This thesis, submitted to the University of Plymouth in partial fulfilment of the requirements for the MSc in Data Science & Business Analytics, explores the application of machine learning (ML) and data mining techniques in diagnosing heart disease. The research aims to develop accurate and reliable predictive models that address the challenges posed by imbalanced data and the complexity of medical datasets. The study involves meticulous data preparation, including cleaning, normalisation, and feature selection, to curate a robust dataset for analysis. Various classification algorithms are developed and trained using these processed datasets, ensuring each model is optimised for the task.

The research critically analyses existing studies, identifies gaps and limitations, and contributes to the field by proposing hybrid models and advanced data preprocessing techniques. The results demonstrate significant improvements in healthcare outcomes and early detection of heart disease, resonating with healthcare professionals and researchers. The thesis concludes with a discussion on the future scope of this research, emphasising the potential for further advancements in ML-based healthcare diagnostics.

Author: Daniel Toluwani Adeleke

Declaration of the Word Count

The dissertation contains a total of **9,000** words.

List of Contents

List of Figures

List of Tables

Acknowledgements

I am deeply indebted to my supervisor, Dr. Raphael Stuhlmeier, whose expert guidance, encouragement, and insightful feedback have been pivotal in shaping this research. His expertise and dedication have greatly contributed to the successful completion of this thesis.

I would also like to extend my sincere gratitude to the University of Plymouth and the Faculty of Science & Engineering. Their essential role, steadfast support, and provision of necessary resources and a conducive environment have been crucial to my research. Their contributions have not only enabled the completion of this thesis but have also elevated its quality. I am especially thankful to the Data Science & Business Analytics program for equipping me with the skills and knowledge needed to undertake this project.

I am particularly grateful to those who provided financial support, without which this research would not have been possible. The support from my parents has been instrumental in allowing me to dedicate myself fully to my studies and research.

My thanks also go to the IEEEDataport website for supplying the vital datasets, especially the heart disease dataset. This data, sourced from the Cleveland Clinic Foundation and the Hungarian Institute of Cardiology in Budapest, has been fundamental to the analysis and findings of this thesis.

Lastly, I am profoundly thankful to my family and friends for their unwavering support and encouragement throughout this journey. Their belief in me has been a constant source of strength and motivation. Thank you all for your invaluable contributions and unwavering support, which have been essential to the completion of this thesis.

1 INTRODUCTION

Heart disease, often referred to as cardiovascular disease, encompasses a broad range of conditions affecting the heart's functionality. These conditions include narrowed or blocked blood vessels, which can lead to heart attacks, angina (chest pain), or strokes. Other types of heart disease affect the heart muscles, valves, or rhythm, and these, too, fall under the umbrella of cardiovascular diseases. The symptoms of heart disease can vary significantly based on the specific condition and can differ between men and women. For instance, men are more likely to experience chest pain, whereas women often report symptoms such as shortness of breath, nausea, and extreme fatigue. (*Mayo Clinic, 2024*) The detection and diagnosis of heart disease at an early stage are crucial for effective treatment and management. Machine learning and data mining have emerged as powerful tools in the healthcare domain, capable of significantly enhancing the early detection and diagnosis of heart disease. These technologies, with their potential to analyse complex medical data, identify patterns and correlations that conventional diagnostics might overlook, and accurately predict disease outcomes, offer hope for the future of healthcare. Implementing machine learning in heart disease diagnosis involves training algorithms on large datasets of patient information, including medical history, lifestyle factors, and clinical data. These algorithms can then predict the likelihood of heart disease in new patients, potentially improving early detection rates and enabling timely interventions. Data mining complements this approach by extracting valuable insights from complex datasets, facilitating the identification of novel risk factors and enhancing the understanding of heart disease progression. This study employs supervised machine learning techniques to make predictions using four distinct classification algorithms: Random Forest, Decision Tree, Naïve Bayes, and Support vector machines. These algorithms are utilised to create a classification model during training using a heart dataset to predict heart disorders. The resulting model is adaptable and capable of forecasting various cardiac conditions. Integrating machine learning and data mining in heart disease detection holds great promise for improving patient outcomes through early diagnosis and personalised treatment plans.

1.1 AIM AND OBJECTIVES

This project research aims to leverage machine learning and data mining to develop a robust diagnostic tool that can assist healthcare professionals in creating a predictive model. It will use the most suitable supervised machine learning algorithms, namely Random Forest, Decision Tree, Naïve Bayes, and Support Vector Machine (SVM), to analyse extensive datasets to enhance the early detection and management of various heart diseases. This study seeks to enhance early detection and prevention strategies by leveraging large datasets and advanced algorithms, ultimately contributing to more personalised and accessible healthcare solutions.

The specific research objectives include:

- Performing a comprehensive literature review to map the current landscape of ML, DM, and their applications in cardiology.
- Executing meticulous data preparation, including cleaning, normalisation and feature selection, to curate a robust dataset for analysis.
- Developing and training the classification algorithms using the processed datasets, ensuring each model is optimised for the task.
- Assessing each model's performance through rigorous validation techniques, comparing their effectiveness in diagnosing heart conditions.

2 LITERATURE REVIEW

The prevalence of heart disease presents a significant health challenge and an economic burden, with direct and indirect costs impacting healthcare systems and patients. (*Benjamin et al., 2019*) The complexity of heart disease necessitates urgent and innovative approaches to enhance early detection, improve treatment strategies, and optimise patient outcomes. This research, which leverages the innovative power of machine learning (ML) and data mining, is an advancement in this direction. These tools have become successful and cutting-edge for improving the diagnosis of heart disease, potentially improving speed, accuracy, and cost-effectiveness.

Researchers have explored various data mining algorithms and machine learning models to predict heart diseases. These studies have employed techniques such as clustering and classification. However, gaps remain in effectively utilising healthcare data and developing robust prediction systems. While some studies demonstrate promising results, others highlight limitations. Agreement exists on the importance of feature selection, model interpretability, and data preprocessing. However, disagreements arise regarding the optimal algorithmic choices and the impact of imbalanced datasets. The use of several ML algorithms in diagnosing heart disease has been the subject of numerous studies.

A review by Detrano et al. (1989) employed logistic regression to predict coronary artery disease, setting a precedent for subsequent ML applications (*Detrano et al., 1989*). This foundation has been strengthened by more recent research that uses more complex algorithms, including ensemble methods, neural networks, and support vector machines (SVMs). For example, Johnson et al. (2016) demonstrated that ensemble methods, which combine multiple models, can significantly improve diagnostic accuracy compared to single algorithms. (*Johnson et al., 2017*)

In a thorough analysis, Dubey et al. (2021) used the Cleveland and Statlog datasets to examine the effectiveness of Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), SVM with grid search (SVMG), KNN, and Naïve Bayes.

He found that on the Cleveland dataset, Logistic Regression and SVM had the highest accuracy (89% accuracy), whereas on the Statlog dataset, LR had the best performance (93% accuracy). (*Dubey, Choudhary and Sharma, 2021*)

While there is consensus on the potential of ML in heart disease diagnosis, researchers diverge on the most effective algorithms and data preprocessing techniques. According to some research, deep learning models are superior because they can handle massive, complicated datasets with great accuracy. (*Rajkomar et al., 2018*) Conversely, others argue that simpler models like decision trees or logistic regression, combined with feature selection techniques, offer better interpretability and comparability across different patient populations. (*Chowdhury et al., 2019*) Studies consistently report high accuracy rates for models such as neural networks, random forests, and support vector machines.

A study by Khandaker et al. (2023) found that Decision Tree algorithms achieved an accuracy of 99.16% in predicting heart disease. (*Uddin et al., 2023*) Similarly, the w-SVM algorithm developed by Yahya et al. demonstrated over 90% accuracy in predicting heart disease based on angiographic tests. (*Banjoko and Abdulazeez, 2021*) Other studies have focused on optimising these algorithms.

“A review by Veisi et al. (2022) developed various ML models, including DT, RF, SVM, XGBoost, and Multilayer Perceptron (MLP), and achieved the highest accuracy of 94.6% using MLP”. (*Ahmad and Polat, 2023*) “Sahoo et al. (2022) reported that RF performed the best, with a classification accuracy of 90.16%”. (*Ahmad and Polat, 2023*)

Several factors, including data quality, feature selection, and algorithm optimisation, influence Machine Learning algorithms' effectiveness in heart disease diagnosis. For instance, *Karthick et al. (2022)* used the Chi-square statistical test for feature selection, which improved the accuracy of the RF classifier to 88.5% (*Ahmad and Polat, 2023*). Similarly, Sarra et al. (2022) applied the Chi-square statistical optimal feature selection method to enhance the prediction accuracy of their SVM-based model from 85.29% to 89.7%. (*Ahmad and Polat, 2023*)

Siddique highlighted the challenges associated with imbalanced data in heart disease diagnosis and various solutions proposed by researchers to address these issues. (*Ahsan and Siddique, 2021*)

The review underscores the importance of data preprocessing techniques like SMOTE (Synthetic Minority Over-sampling Technique) in balancing training data and enhancing model performance. Despite the ongoing debate on the best approach to handling imbalanced datasets, researchers like Siddique persevered in finding solutions. Some advocate for data-level solutions like SMOTE, while others propose algorithm-level modifications. Siddique's work is a testament to the enduring effort to overcome the challenge of imbalanced data in applying ML models for heart disease diagnosis. (*Ahsan and Siddique, 2021*)

Despite the advancement, several gaps remain in the existing literature. One prominent gap is the lack of standardised datasets, which hampers the reproducibility and generalizability of findings. Additionally, the issue of imbalanced data, common in medical datasets, has not been adequately addressed, leading to biased and suboptimal performance of ML models.

There is a pressing need for more research in the field of machine learning and healthcare. Most studies rely on publicly available datasets like Cleveland and Statlog, which may not fully represent the diversity of patient populations globally. (Nagavelli, Samanta and Chakraborty, 2022) Furthermore, there is a significant gap in research on integrating multi-modal data (e.g., combining ECG data with patient history) to improve diagnostic accuracy. This presents a crucial opportunity for researchers to contribute to the field by exploring the integration of ML models into clinical workflows and evaluating their impact on patient outcomes. Additionally, most studies are retrospective, relying on historical data, which may not fully capture the dynamic nature of heart disease. Prospective studies involving real-time data collection and monitoring could provide more robust insights.

Moreover, the potential impact of integrating multimodal data, such as medical images, genomic data, and electronic health records, is exciting. This could enhance the predictive power of ML models and revolutionise the field of heart disease diagnosis.

Recent studies have attempted to address these gaps by developing hybrid models and incorporating advanced data preprocessing techniques. For example, the survey by *Khandaker et al. (2023)* combined multiple datasets to train and test their ML models, thereby improving the generalizability of their findings. (*Uddin et al., 2023*)

Similarly, hybrid models, such as RF combined with linear models or XGBoost, have shown promise in improving diagnostic accuracy and robustness. (*Nagavelli, Samanta and Chakraborty, 2022*) Future research should also focus on developing more interpretable models or incorporating explainable AI techniques to enhance trust and adoption in clinical settings. Additionally, addressing the issue of imbalanced datasets through advanced techniques like synthetic data generation or cost-sensitive learning could improve model performance.

The theoretical foundation of using Machine Learning in heart disease diagnosis is based on the ability of these algorithms to identify complex patterns in large datasets. Techniques like SVM, RF, and neural networks are particularly effective in handling high-dimensional data and capturing non-linear relationships between variables. Integrating data mining techniques, such as frequent itemset mining, further enhances the ability to identify significant predictors of heart disease. (*Ilayaraja M., Meyyappan T., 2015*)

Several theoretical frameworks and concepts underpin the application of ML and data mining techniques for heart disease diagnosis. One fundamental theory is the Supervised Learning paradigm, which involves training ML models on labelled data to learn patterns and make predictions on unseen data. Popular supervised learning algorithms used in heart disease diagnosis include Support Vector Machines (SVMs), Decision Trees, Random Forests, and Neural Networks.

Feature selection is another relevant theory that seeks to enhance the performance and interpretability of machine learning models by identifying the most pertinent and informative characteristics from the given data. Numerous feature selection strategies, including filter, wrapper, and embedding approaches, have been investigated in relation to the detection of heart disease.

Data Imbalance is a critical issue in medical datasets, where one class (e.g., healthy individuals) is significantly more prevalent than the other (e.g., individuals with heart disease). This imbalance can lead to biased ML models favouring the majority class, resulting in poor performance for the minority class. Techniques such as oversampling, undersampling, and ensemble methods have been proposed to address this challenge.

This review's logical progression is guided by the need to develop accurate and reliable ML models for heart disease diagnosis while addressing the challenges posed by imbalanced data and the complexity of medical datasets.

This review aims to provide a solid foundation for future research in this field by critically analysing existing research and identifying gaps and limitations. The goal is to significantly contribute to improved healthcare outcomes and early detection of heart disease, a prospect that should resonate with all healthcare professionals and researchers.

The literature analysis above clarifies that data mining algorithms have successfully predicted heart diseases. It is also troubling since the model's reliability for predicting cardiac illnesses with various risk variables is questionable. However, SVM, Naïve Bayes, Decision Trees, Bagging and Boosting, and Random Forests have produced trustworthy findings for diagnosing heart disease. (*Jan et al., 2018*) Existing studies have demonstrated the potential of these approaches in improving diagnostic accuracy and efficiency. However, challenges such as imbalanced data, feature selection, and model interpretability remain to be addressed. The critical examination of past research highlights the need for integrating multiple algorithms, addressing biases, and ensuring model interpretability. The review has identified several promising research directions, including the development of ensemble methods, advanced data preprocessing techniques, and the integration of domain knowledge into ML models. These directions hold great promise for the future of healthcare, inspiring hope for developing more robust and reliable ML models for heart disease diagnosis, ultimately leading to improved patient outcomes and more effective healthcare delivery.

3 METHODOLOGY/PROCEDURE

This is a structured approach to conducting research, serving as a vital framework that ensures the study is organised, valid, and reliable. In this research, the system architecture follows a sequential process, beginning with data collection and concluding with evaluating the performance metrics. Data collection, the first phase, is crucial to obtaining the information needed to train the model. This may entail gathering data from various sources, including databases, scraped websites, or legally accessible datasets. (*Kotsiantis, 2007*) Once it has been gathered, preprocessing is required to guarantee that the data is in an appropriate format for analysis. In machine learning, it is described as the process of cleaning and converting unintelligible raw data into legible data. Data preprocessing is used for data reduction, cleansing, and transformation. Filling in missing numbers, reducing noise in the data, and eliminating outliers is known as data cleaning; normalisation and aggregation are examples of data transformation. Normalising the data is essential to ensure that distance-based algorithms such as SVM perform effectively. The data reduction process reduces the amount of data, but the result remains the same. Most publicly available datasets are already preprocessed, making them suitable for use without preprocessing. Subsequently, the data is collected in a format that can be used and divided into train and test sets. The model is trained on the train set, and its performance is assessed on the test set. A common split ratio is 70:30 for the train and test data. *(Kohavi, 1995)* During the phase where the training data is used to train the model, the machine-learning algorithm learns the patterns and relationships in the data. After the training, the next step is the feature selection. It involves selecting the dataset's most impactful features or variables from the dataset to improve the model's performance. Techniques like recursive feature elimination or principal component analysis can be used for this purpose. After the training, the model is validated on the test to assess its performance. The model's performance is evaluated using various evaluation metrics to check its efficiency. "Common metrics include accuracy, precision, recall and F1 score, confusion matrix and ROC curves". *(Powers, 2011)* Python is the language chosen for this application. It is widely regarded as the language of choice for computer models and has special tools that effectively work with machine learning systems.

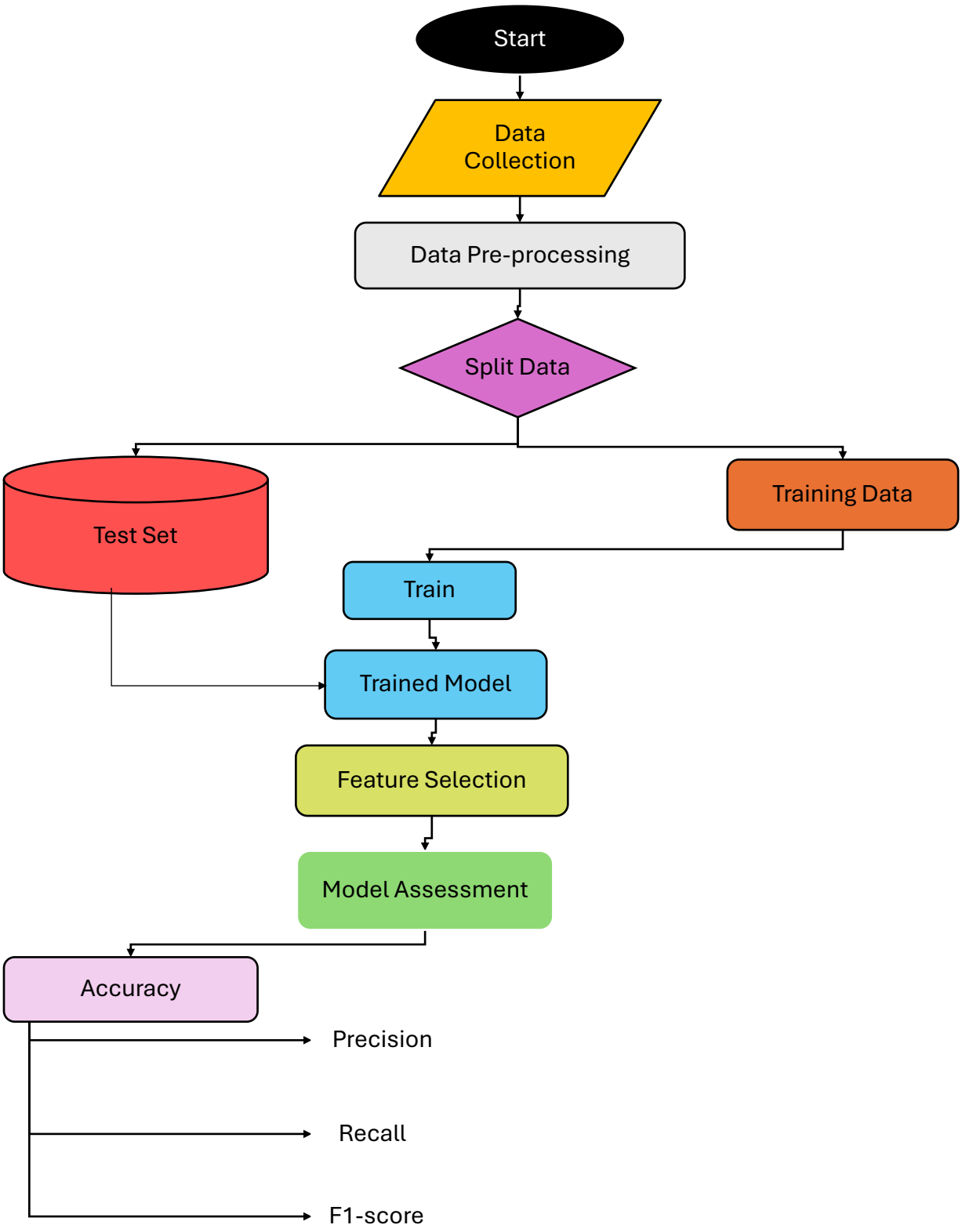


Figure 1: *System Architecture of the research methodology*

3.1 Heart Disease Dataset Description

A dataset is a collection of data. “This dataset has 1190 instances with 11 features, including age, sex, chest pain type, resting BP, cholesterol, fasting blood sugar, resting ECG, max heart rate, exercise angina, old peak ST slope, and target” *(Manu Siddhartha, 2020)*. “It is sourced from the IEEEDataport website, where the data was combined from the five popular heart disease datasets already available independently but not combined before”. *(Manu Siddhartha, 2020)*

- “Cleveland dataset (from Cleveland Clinic Foundation)
- Hungarian dataset (from the Hungarian Institute of Cardiology, Budapest)
- Switzerland dataset (from University Hospital, Zurich, Switzerland)
- Long Beach VA dataset (from V.A Medical Centre, Long Beach, CA)
- Stalog (Heart) dataset”. *(Manu Siddhartha, 2020)*

In the dataset, 629 patients (52.9%) present the presence of heart disease, while 561 patients (47.1%) are normal.

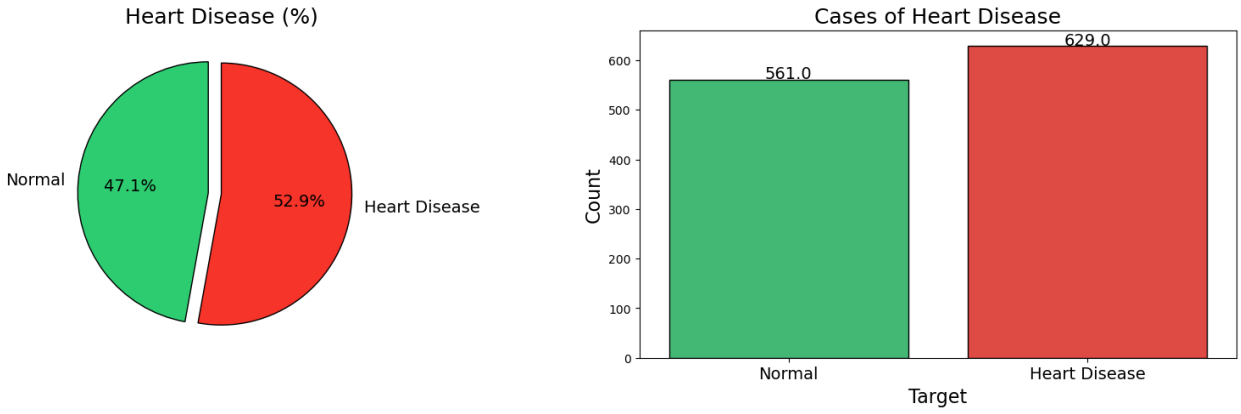


Figure 2: *Visualization of the target feature of the heart disease dataset*

Feature Name	Type	Detail
age	Integer	Age of the patient
sex	1: male, 0: female	Sex of the patient
chest pain type	1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic	Patient having which type of pain
resting bp s	Integer	Patient's blood pressure level in mmHg
cholesterol	Integer	Patient's cholesterol level in mg/dl
Fasting blood sugar	(Fasting bs >120 mg/dl) 1: true, 0: false	Patient's fasting blood sugar level.
resting ecg	0: normal, 1: ST-T wave abnormality, 2: left ventricular hypertrophy	Patients resting electrocardiogram result
max heart rate	71 – 202 (Integer)	Patient's maximum heart rate
exercise angina	1: yes, 0: no	Patient's induced angina
oldpeak	Numeric	Patient's ST value in depression
ST slope	1: upsloping, 2: flat, 3: downsloping	Slope of peak exercise ST segment
target	1: heart disease, 0: Normal	Diagnosis of heart diseases

Table 1: *Heart disease dataset attribute description (Manu Siddhartha, 2020)*

3.2 Data Mining

The process of going through massive data sets to find patterns and relationships that may be used to address business problems through data analysis is known as data mining. It is a vital component of data analytics and one of the main fields of data science, which uses sophisticated analytics methods to extract insights from data sets. (*Gillis, Stedman and Hughes, 2024*) These techniques have practical applications in data science, making them invaluable tools for solving real-world problems. The following categories of data mining techniques are commonly used:

- **Classification**: This technique involves finding a model or function that helps divide data into classes based on different attributes. (*Mitchell, 1997*)
- **Clustering:** In contrast to classification, clustering is the process of arranging a set of objects into groups based on their similarity to one another. Techniques like K-means and hierarchical clustering are widely used. (*Jain et al., 1999*)
- **Association Rule Learning**: This method is used to discover interesting relations between variables in large datasets. A famous example of association rule learning is the market basket analysis, which tries to find products that frequently co-occur in transactions. (*Agrawal et al., 1993*)
- **Regression**: Regression involves identifying the relationship between variables. It is used for forecasting and finding causal relationships among the variables. Linear regression is often used when the relationship between variables is assumed to be linear. (*Montgomery et al., 2012*)
- **Neural Networks**: Neural networks, which take their cues from the human brain, are used in data mining to simulate intricate interactions between inputs and outputs or to identify patterns in data. They are compelling for classification and regression tasks. (*Haykin, 2009*)
- **Anomaly Detection**: This technique involves identifying rare items, events, or observations that differ significantly from most data and raising suspicions. It is used in fraud detection and network security. (*Chandola et al., 2009*)

3.3 Machine Learning Models/Algorithms

Within artificial intelligence (AI), machine learning (ML) is a subset that enables systems to automatically learn from experience and get better at it without needing to be explicitly programmed. Mathematical models called machine learning models are used to categorise or forecast data depending on input data. Reinforcement learning, supervised, unsupervised, and semi-supervised models are the general categories into which these models can be divided. Each category includes several types, as shown in Figure 3.

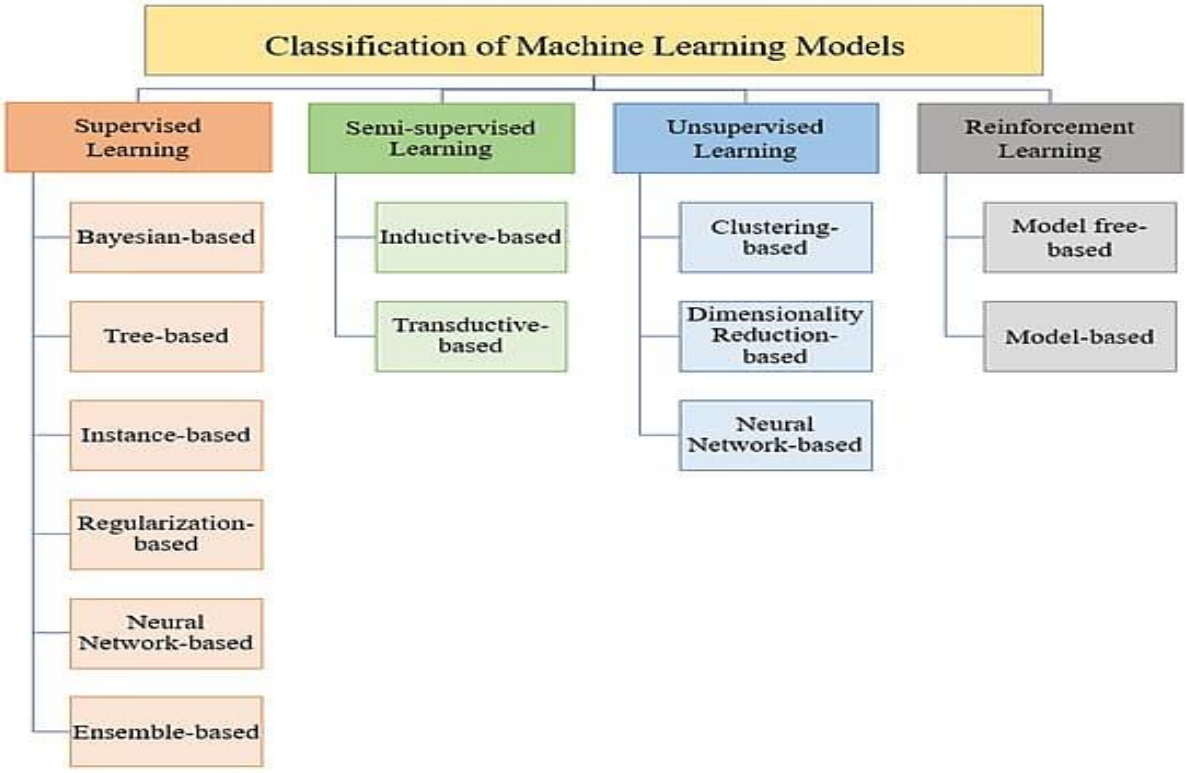


Figure 3: *Classification of machine learning models (Khoei and Kaabouch, 2023)*

- **Supervised Learning**: Labelled data is used to train supervised learning algorithms. This implies that an output label or result is associated with every training example. “Neural networks, decision trees, and support vector machines (SVM) are examples of common algorithms”. (*Murphy, 2012*) For instance, a linear regression model predicts a continuous output variable based on one or more input variables. They are used in spam detection, sentiment analysis, and predictive analytics applications.

- **Unsupervised Learning**: Unsupervised learning models are used when the training data is not labelled. The model tries to learn patterns and structures from the data. Common techniques include clustering (e.g., k-means clustering) and association (e.g., the Apriori algorithm). They are used in customer segmentation and anomaly detection.
- **Semi-supervised Learning**: Semi-supervised learning features a combination of both labelled and unlabeled data. These models fall between supervised and unsupervised learning and are typically used when acquiring a fully labelled dataset, which is expensive or time-consuming. They are used in robotics gaming and autonomous vehicles.
- **Reinforcement Learning**: Reinforcement learning models learn to make decisions by performing actions and receiving feedback via rewards or penalties. A common algorithm is Q-learning. (*Sutton and Barto, 2018*)

"A general structure of a machine learning-based predictive model has been shown in Figure 4, where the model is trained from historical data in phase 1, and the outcome is generated in phase 2 for the new test data". (*Sarker, 2021*)

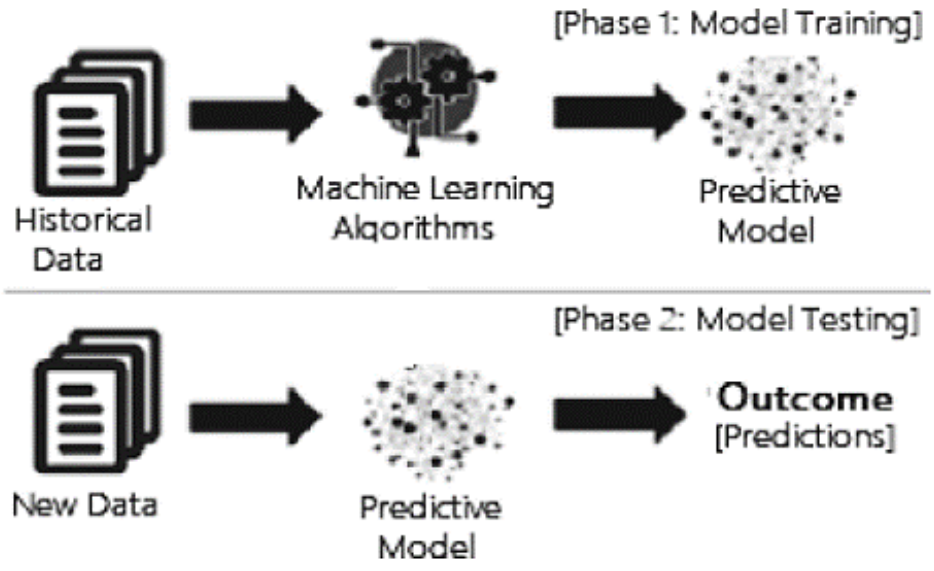


Figure 4: "*A general structure of a machine learning-based predictive model" (Sarker, 2021)*

3.3.1 Decision Tree

A decision tree is a non-parametric supervised machine learning algorithm with broad applicability due to its simplicity and interpretability. It is intuitive and easy to interpret, making it a valuable tool in various domains. (*Quinlan, 1986*) It involves splitting the data into subsets based on the value of input features, which helps make predictions for a target variable. There are nodes and edges in a decision tree. Every leaf node depicts a result, every edge denotes a decision rule, and every interior node represents a feature (or attribute). The decision tree process divides the dataset into subsets using an attribute value test. The splitting continues recursively, forming a tree structure. Decision trees are easy to understand and visualise, even for non-experts. They do not require scaling or normalisation. However, they can easily overfit the training data, especially with deep trees. Slight variations in the data can generate a completely different tree. If some classes dominate, the tree might become biased toward those classes. Its tendency to overfit and sensitivity to data variations necessitate careful tuning and techniques like pruning to ensure robust performance. “Decision tree classifies instances classified by checking the attribute defined by that node, starting at the tree's root node, and then moving down the tree branch corresponding to the attribute value. For splitting, the most popular criteria are “Gini” for the Gini impurity and “entropy” for the information gain that can be expressed”. (*Sarker, 2021*)

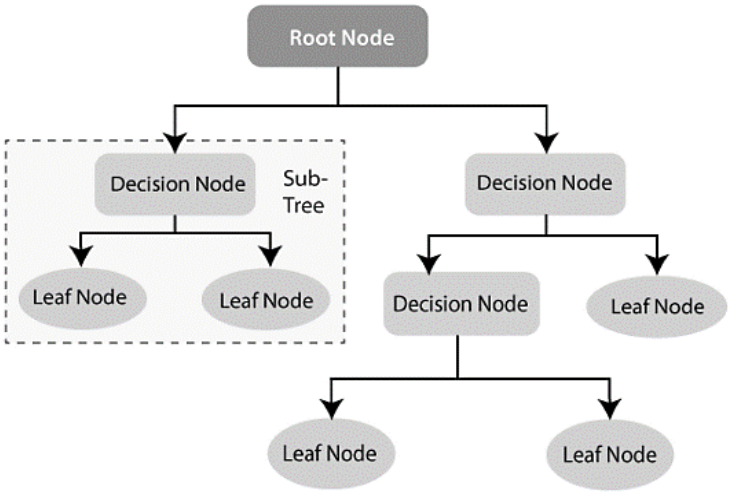


Figure 5: *A decision tree structure (Sarker, 2021)*

Entropy: Entropy measures the uncertainty or impurity in a dataset. It is used to determine how mixed the data is. The formula for entropy is given by:

$$\boldsymbol{H = -\sum_{i=1}^{n} (p_i) \log_2 (p_i)}$$

- $\boldsymbol{H}$**:** Entropy of the random variable.
- $\boldsymbol{p_i}$**:** The proportion of instances belonging to class $\boldsymbol{i}$**.**
- $\boldsymbol{n}$**:** The total number of possible outcomes.
- $\boldsymbol{\log_2}$**:** Logarithm to the base 2.

Entropy reaches its maximum when all outcomes are equally likely, indicating maximum uncertainty. Conversely, it is zero when there is no uncertainty (i.e., one outcome has a probability of 1, and all others have a probability of 0). (*Niranjan Appaji, 2024*)

Gini Impurity: It quantifies the probability that a new instance would be incorrectly classified if randomly labelled based on the dataset's label distribution. The formula for Gini impurity is:

$$\boldsymbol{G = 1 - \sum_{i=1}^{k} p_i^2}$$

- $\boldsymbol{G}$: Gini index.
- $\boldsymbol{p_i}$**:** The proportion of elements belonging to class $\boldsymbol{i}$ in the node
- k: The total number of classes

The Gini Index ranges from 0 to 0.5. A Gini Index of 0 indicates perfect purity (all elements belong to a single class). In contrast, a Gini Index of “0.5 indicates maximum impurity (elements are evenly distributed among all classes)”. (*Niranjan Appaji, 2024*)

3.3.2 Random Forest

Random Forest is a robust and flexible supervised classification algorithm within the larger class of ensemble learning techniques. It generates more accurate outcomes than any one model by combining predictions from several models. During training, the algorithm builds numerous decision trees and outputs the mode of the classes of each unique tree. (*Breiman, 2001*) An army of trees is trained in random forests rather than just one. Random forest is created using bootstrap samples (random sampling with replacement from the original dataset) of the training data and random subsets of features at each split. This method combines the “bagging” idea and the random selection of features. The unpruned decision trees in the ensemble contribute to the high variance reduction, leading to better model performance. (*Svetnik et al., 2003*) During the construction of each tree, random subsets of features are considered for splitting nodes, thus ensuring that the trees are diverse and reducing the likelihood of overfitting. It works well with large datasets and estimates feature importance, which can be useful for feature selection. However, training can be slow, especially with many trees and features. Also, the model can become complex and challenging to interpret. Random Forests are widely used for their high accuracy and ability to handle large datasets with higher dimensionality. They are particularly effective in applications involving classification problems, such as predicting medical diagnoses or categorising images. The algorithm's versatility and robustness suit various domains. (*Schonlau and Zou, 2020*) Random Forest classifiers have been proven to deliver superior performance compared to individual decision trees, but this comes at the cost of losing human interpretability. (*Caruana & Niculescu-Mizil, 2006*) Their accuracy and robustness against overfitting make them a preferred choice for many machine learning tasks. The ensemble approach allows random forests to generalise unseen data better, making them highly reliable. (*Fawagreh, Gaber and Elyan, 2014*) Several systematic reviews and meta-analyses have been conducted to evaluate the performance of Random Forest classifiers. These studies highlight the consistent and reliable performance of Random Forests across different datasets and applications, reinforcing their position as a powerful tool in machine learning. (*Sarica, Cerasa and Quattrone, 2017*)

3.3.2.1 Hyperparameter Tuning (n_estimators)

The hyperparameter n_estimators are crucial in ensemble learning methods, particularly in algorithms like Random Forest and Gradient Boosting. This parameter determines the number of trees (estimators) to be built in the ensemble model. The choice of the n_estimators value can significantly impact the performance of the model, influencing both its accuracy and computational efficiency. In the context of Random Forest, n_estimators specify the number of decision trees the algorithm will create. Each tree is trained on a bootstrap sample of the data, and their predictions are averaged (for regression) or voted on (for classification) to produce the final model output. A higher number of estimators generally leads to better performance due to reduced variance but also increases computational cost.

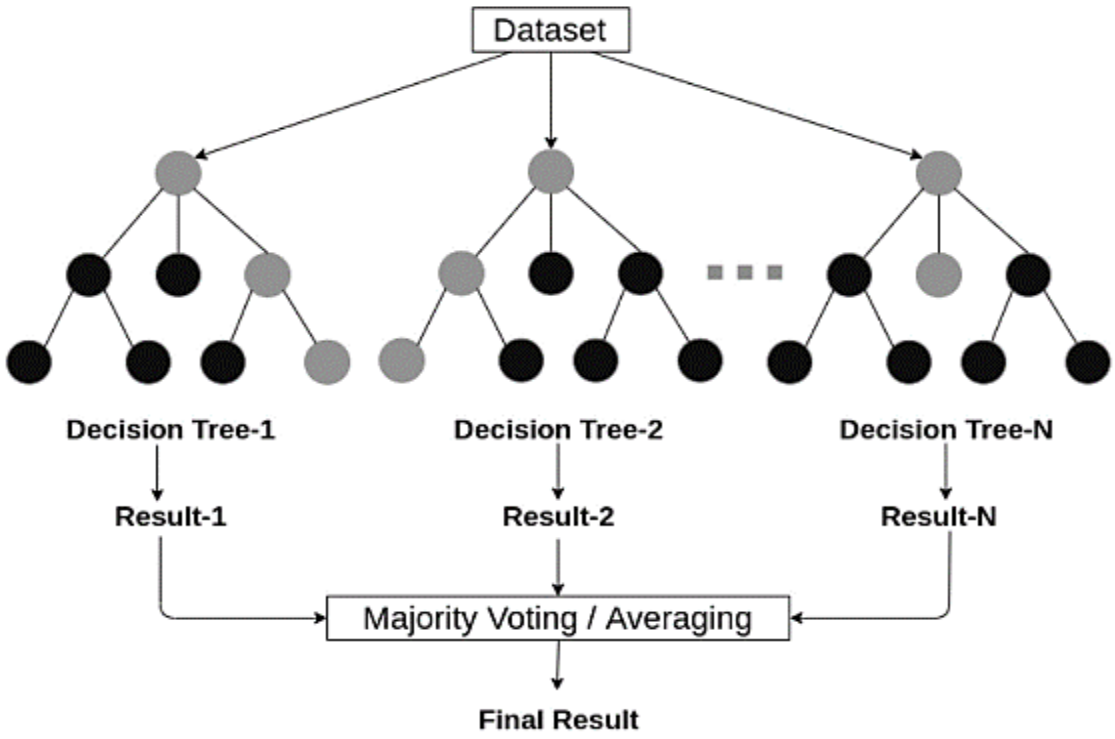


Figure 6: *"A random forest structure considering multiple decision trees"* *(Sarker, 2021)*

3.3.3 Naïve Bayes

Based on the feature independence assumption, the Naïve Bayes algorithm is a family of probabilistic algorithms that utilises the Bayes Theorem. Large datasets and text classification problems benefit significantly from their effectiveness. (*McCallum and Nigam, 1998*) It operates in the manner described below. Firstly, the probability of a class given a set of features is computed using the Bayes Theorem. Next, it assumes that a feature's existence in one class does not imply the existence of any other feature. Lastly, it calculates the likelihood of each feature inside each class and combines these estimates to create predictions. Naïve Bayes is easy to implement and computationally efficient. It scales well with the number of features and data points. However, if a feature is not present in the training data for a class, it assigns zero probability, which can be problematic. Naive Bayes is commonly used in spam filtering, sentiment analysis, and medical diagnosis. Given a new, unseen data point (input vector) $z = (z_1, \dots, z_M)$, "Naive Bayes classifiers, which are a family of probabilistic classifiers, classify z based on applying Bayes' theorem with the "naive" assumption of independence between the features (attributes) of $\mathbf{z}$ given the class variable t" (*Mahdavinejad et al., 2018*). "By applying Bayes' theorem, we have

$$p(t = c \mid z_1, \dots, z_M) = \frac{p(z_1, \dots, z_M \mid t = c)p(t = c)}{p(z_1, \dots, z_M)}$$

and by applying the naïve independence assumption and some simplifications, we have

$$p(t = c \mid z_1, \dots, z_M) \propto p(t = c) \prod_{j=1}^{M} p(z_j \mid t = c)$$

Therefore, the form of the classification task is

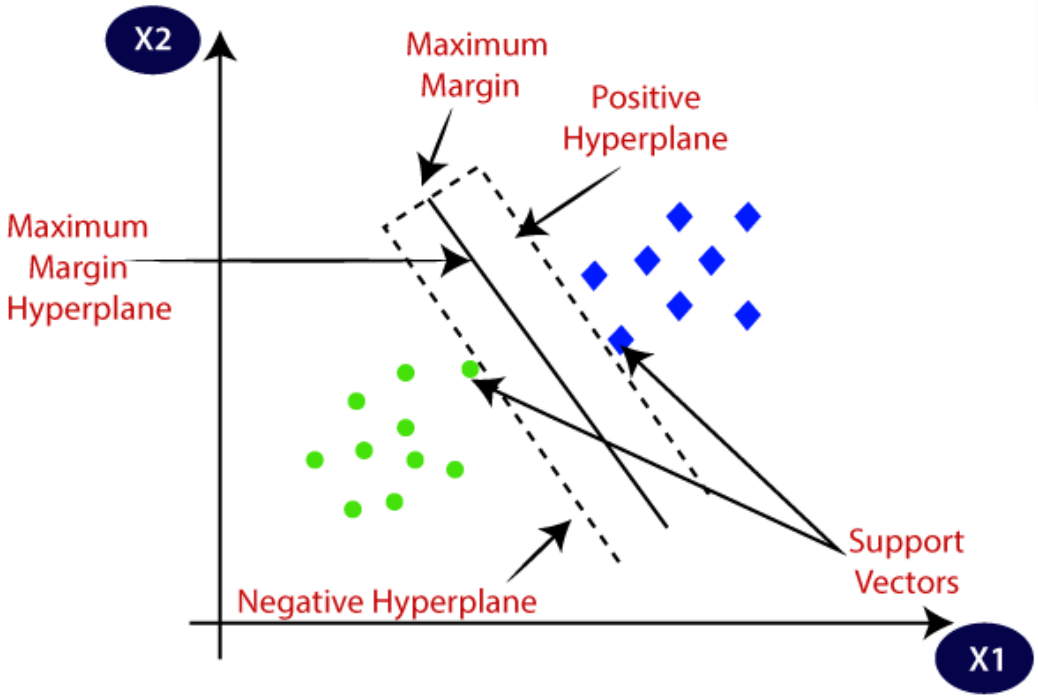
$$y = \arg\max_{c} p(t = c) \prod_{j=1}^{M} p(z_j \mid t = c)$$

Where y denotes the predicted class for z. Different naïve Bayes classifiers use various approaches and distributions to estimate $p(t = c)$ and $p(z_j \mid t = c)$ ". (*Zhang, 2004*)

3.3.4 Support Vector Machine (SVM)

A supervised learning, non-probabilistic binary classifier called Support Vector Machine (SVM) seeks to identify the dividing hyperplane with the most significant margin that separates the classes in both train sets. Then, depending on which side of the hyperplane a new, unseen data point falls, its anticipated label is ascertained. It functions by locating the ideal hyperplane in the feature space that best divides the classes. (*Cortes and Vapnik, 1995*) Support vectors are the data points that are closest to the hyperplane and are essential in determining the hyperplane's location and orientation. The margin is the distance between the hyperplane and the nearest N data point from either class, known as support vectors. The number of data points chosen for each heart disease dataset class is 183 and 179, respectively. For a linearly separable dataset, the objective of the SVM is to maximise this margin. The support vectors are the data points that lie closest to the decision surface, and they directly affect the position of the hyperplane. SVMs are inherently linear classifiers, but they can be adapted to work in a non-linear fashion using a method called the kernel trick. This involves transforming the original data into a higher-dimensional space where a linear separator can be found. Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid. SVM is highly effective and robust to overfitting, especially in high-dimensional spaces and datasets where the number of dimensions exceeds the number of samples. Different kernel functions can be specified for the decision function, providing flexibility in the choice of the model. However, training can be time-consuming and memory-intensive, especially with large datasets. SVM is widely used in bioinformatics (gene classification), image recognition, and text categorisation. With an SVM, the hyperplane with the most significant margin between the two classes is the optimal one. To do this, it first determines which hyperplane is the furthest from the data points or has the largest margin, and it then selects those hyperplanes that best classify the labels. The hyperplane with the maximum margin is called the optimal hyperplane. The primary goal of SVM is to find the optimal hyperplane which maximises the margin. This is done by solving the following optimisation problem where $w$is the weight vector, b is the bias, and x_i, y_i are the data points and their respective labels. (*Burges, 1998*)

$$min\frac{1}{2}\|w\|^2 \text{ subject to } y_i(w \cdot x_i + b) \geq 1$$



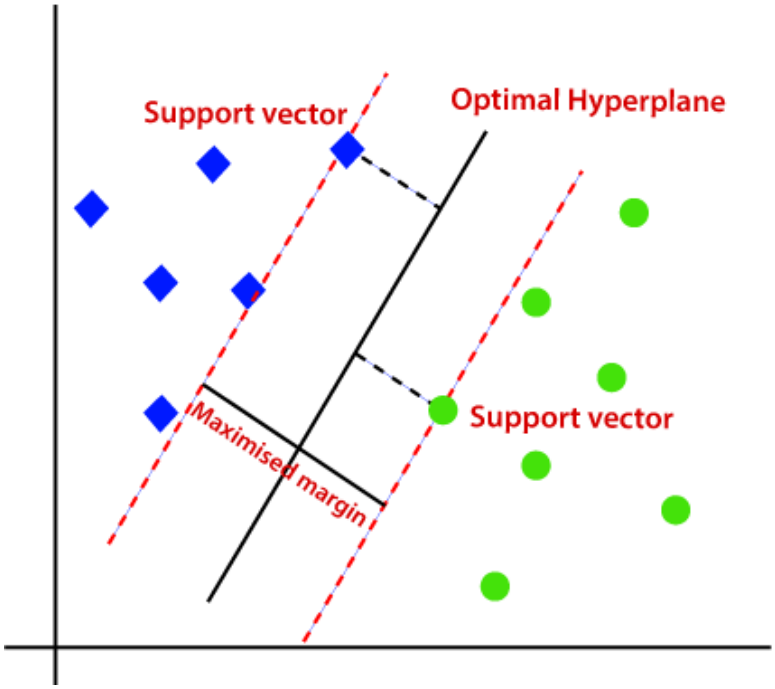


Figure 7: *Support Vector Machine (SVM) classification diagrams (Saini, 2024)*

4 RESULTS & DISCUSSION

All results were obtained on a computer with Core i7 10750h, 32 GB RAM, and a Nvidia RTX 3060 GPU. The interface used for the implementation was the Jupyter Notebook. The primary focus is to evaluate the performance of these models and understand the significance of different features in predicting the target variable (heart disease presence). The code for the dissertation was written in Python, a widely used programming language in data science and machine learning, due to its simplicity and the extensive range of libraries available for data manipulation, visualisation, and model building. The specific libraries used in the code include Numpy, Pandas, Matplotlib and Scikit-learn. The feature importance was calculated using all four models. The aggregate importance score was derived by averaging the importance across all models, highlighting the most significant features for heart disease prediction. Determining the feature importance in a dataset is crucial in understanding which variables significantly influence the target outcome in predictive modelling. The process uses several machine learning techniques, yielding a comprehensive knowledge of feature importance. It starts with the data preprocessing, where the dataset is first divided into features (X) and the target variable (y). This separation allows for a focused analysis of the predictors and their relationship with the target. The dataset is then split into training and testing subsets to evaluate model performance effectively. To ensure that the features contribute evenly to the model, they are standardised using StandardScaler. This scaling process transforms the data so that each feature has a mean of zero and a standard deviation of one, thus facilitating better model convergence and performance. Subsequently, several machine learning models are initialised. In this case, the models include Decision Tree, Random Forest, Naive Bayes, and Support Vector Machine (SVM). Each of these models has unique characteristics and methods for determining feature importance. For instance, Decision Tree and Random Forest models provide a feature_importances_ attribute, which quantifies the contribution of each feature in splitting data points. On the other hand, SVM utilises the coefficients (coef_) derived from the decision boundary it creates to denote feature importance. It is noteworthy that Naive Bayes does not inherently provide feature importance due to its probabilistic nature.

The next phase involves calculating feature importance across these models. Each model is fitted to the training data; where applicable, the feature importances are extracted and stored. This collection of importances forms the feature_importance_info, a crucial dataset for further analysis. Once these importances are gathered, they are aggregated to provide a holistic view of feature significance. This aggregation involves summing the importance provided by each model and normalising them by dividing them by the number of models. This step ensures that the crucial values are balanced and comparable across the different models. The features are sorted according to their aggregated importance values after the aggregation. This sorting provides a ranked list of features, highlighting those with the most significant impact on the target variable. These rankings are organised into a data frame to facilitate interpretation and decision-making, visually representing the feature's importance. The process culminates in visualising the results. Figure 8 shows a bar plot displaying the feature's importance, allowing for an intuitive understanding of which features are paramount. Additionally, another plot, as shown in Figure 9 below, illustrates model performance as the number of selected features increases. This visualisation helps identify the optimal number of features contributing to the best model performance without overfitting.

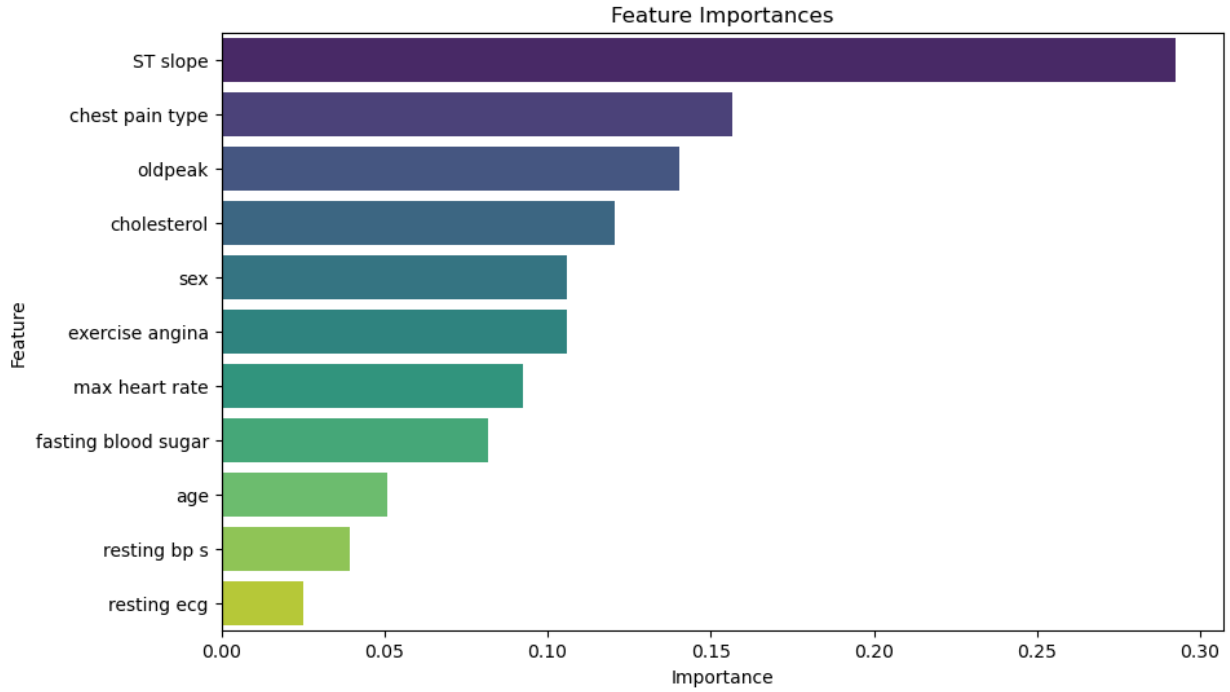


Figure 8: *Plot of increasing Feature Importance*

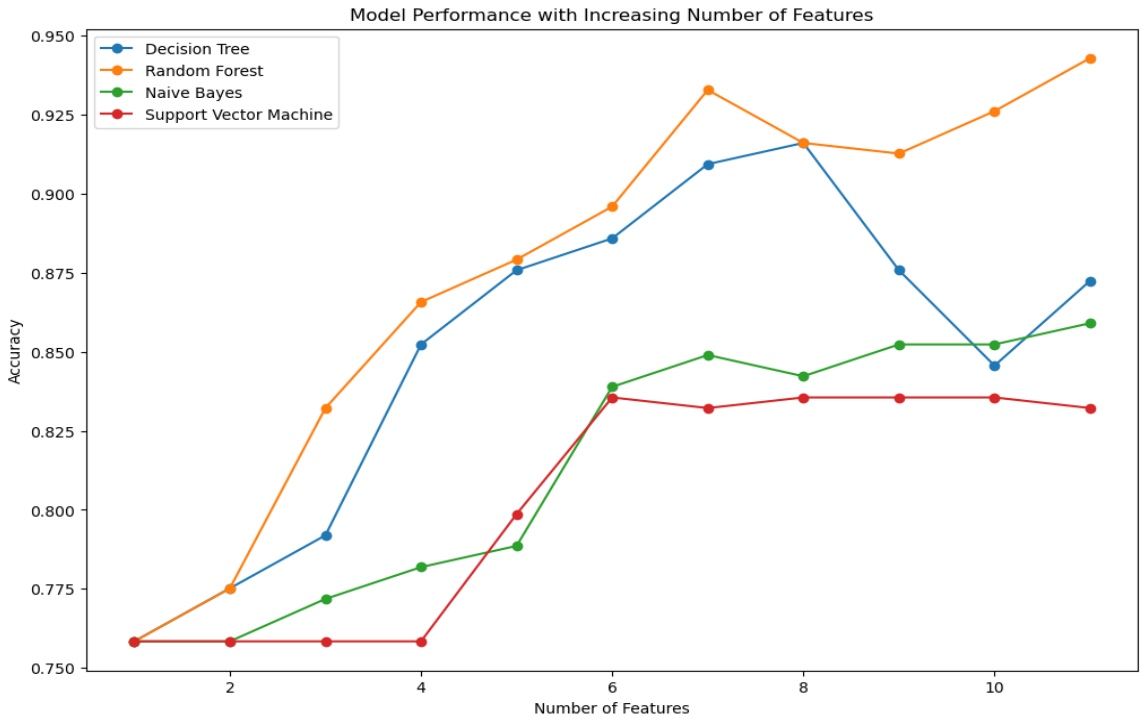


Figure 9: *Plot of Model Performance with increasing Number of Features*

```
Model: Decision Tree, Highest Accuracy: 0.9161073825503355, Number of Features: 8
Model: Random Forest, Highest Accuracy: 0.9429530201342282, Number of Features: 11
Model: Naive Bayes, Highest Accuracy: 0.8590604026845637, Number of Features: 11
Model: Support Vector Machine, Highest Accuracy: 0.8355704697986577, Number of Features: 6
```

The plot in Figure 9 above provides valuable insights into how different machine learning models benefit from including more features. It helps in understanding the importance of feature selection and how it impacts the performance of various models. In this case, Random Forest and Decision Tree significantly improve with more features, while SVM and Naive Bayes exhibit more stable performance. These findings have practical implications for predictive modelling in healthcare, as they highlight the importance of feature selection in improving model performance and the potential for different models to perform differently based on the number of features included.

4.1 Evaluation Metrics

“Evaluating machine learning models is crucial to understanding their performance. Common metrics include accuracy, precision, recall, F1-score, and area under the receiver operating characteristic (ROC) curve”. (*Fawcett, 2006*) These metrics help determine how well a model performs and where improvements can be made. These metrics differ depending on the application and the category of the models. Below are short descriptions of the most common metrics and their equation.

- **Accuracy**: “This is the ratio of correctly predicted instances to the total instances”. (*GeeksforGeeks, 2020*) It is a fundamental metric for evaluating classification models, especially when the class distribution is balanced. (*GeeksforGeeks, 2020*) FP: False Positives, FN: False Negatives, TP: True Positives, TN: True Negatives

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

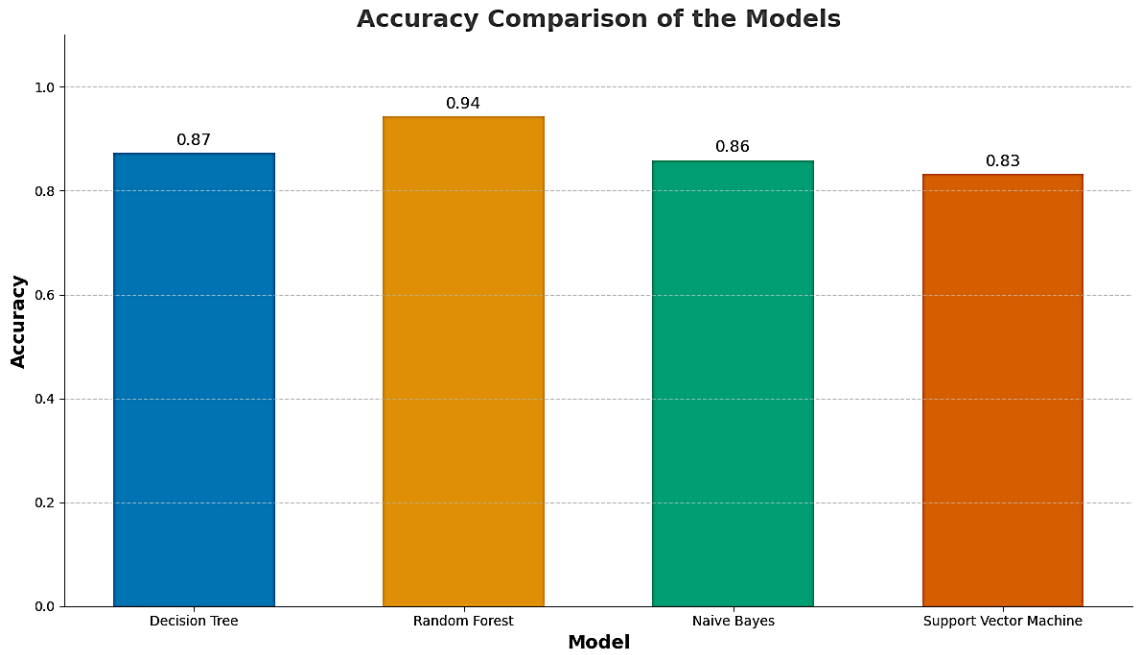


Figure 10: *Plot showing the comparison between the models' accuracy*

- **Precision, Recall and F1 Score**: “Precision is the ratio of true positive predictions to the total predicted positives. Recall (Sensitivity) is the ratio of true positive predictions to the actual positives. The F1 Score is the harmonic mean of precision and recall, providing a balance between the two”. (*GeeksforGeeks, 2020*)

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

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

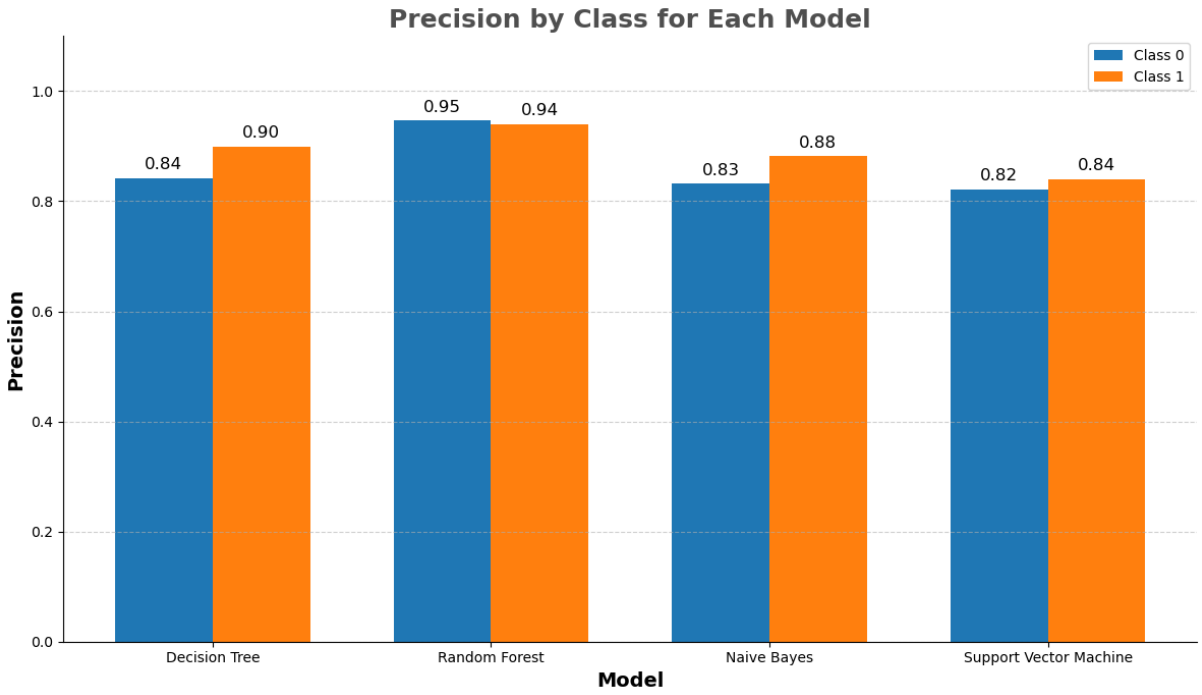


Figure 11: *Plot showing the comparison between the models' precision*

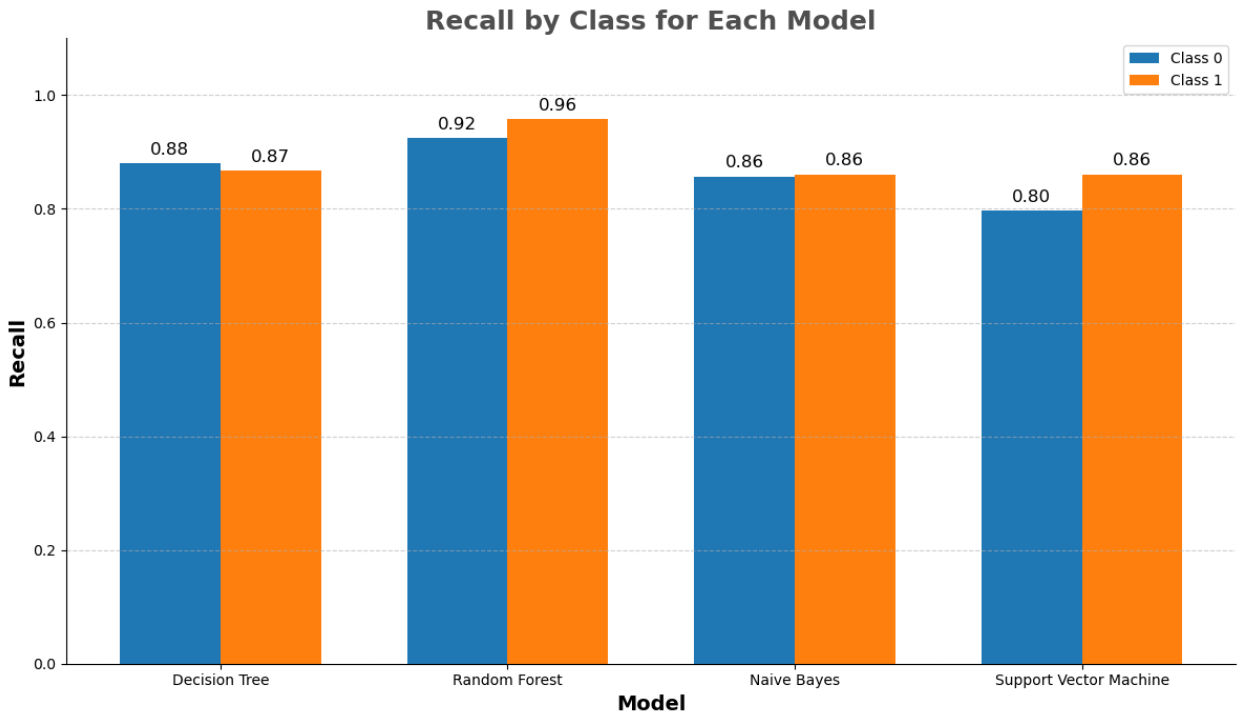


Figure 12: *Plot showing the comparison between the model's recall*

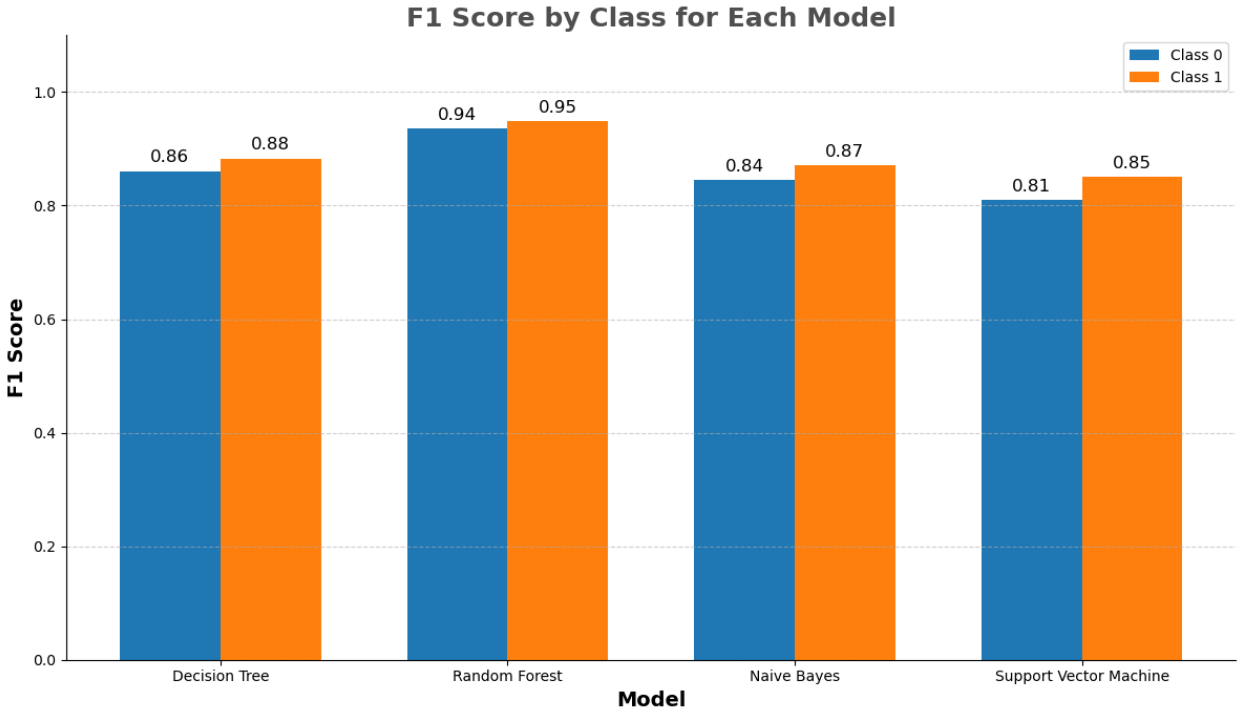


Figure 13: *Plot showing the comparison between the model's F1-Score*

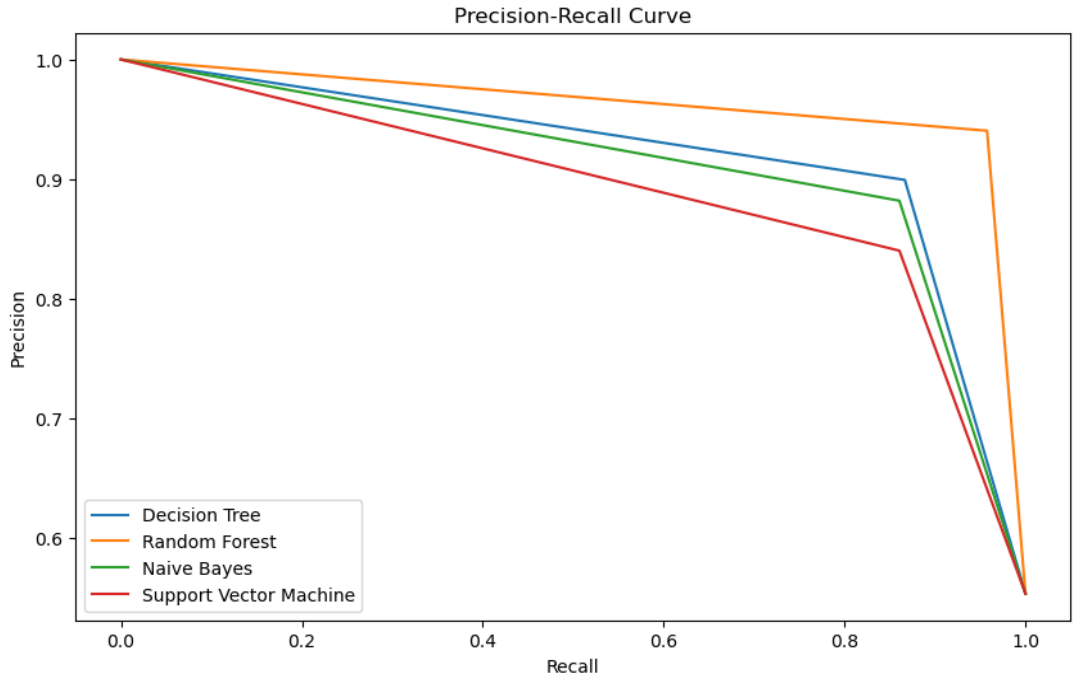


Figure 14: *Precision-Recall Curve for all the Model*

Figure 14 above shows a Precision-Recall Curve used to evaluate the performance of different machine learning models. The closer the curve is to the top-right corner, the better the model distinguishes between positive and negative classes. A high area under the curve (AUC) indicates a model with high precision and recall values.

The Random Forest (orange curve) generally shows the best performance with the highest precision and recall values, followed by the Decision Tree (blue curve), naive Bayes (green curve), and Support Vector Machine (red curve).

- **Confusion Matrix**: "A confusion matrix is a table used to describe the performance of a classification model. It contains information about actual and predicted classifications done by the model ". (GeeksforGeeks, 2020)

	Actual Positive	**Actual Negative**
Predicted Positive	True Positive TP	False Positive FP
Predicted Negative	False Negative FN	True Negative TN

Table 2: *Confusion Matrix*

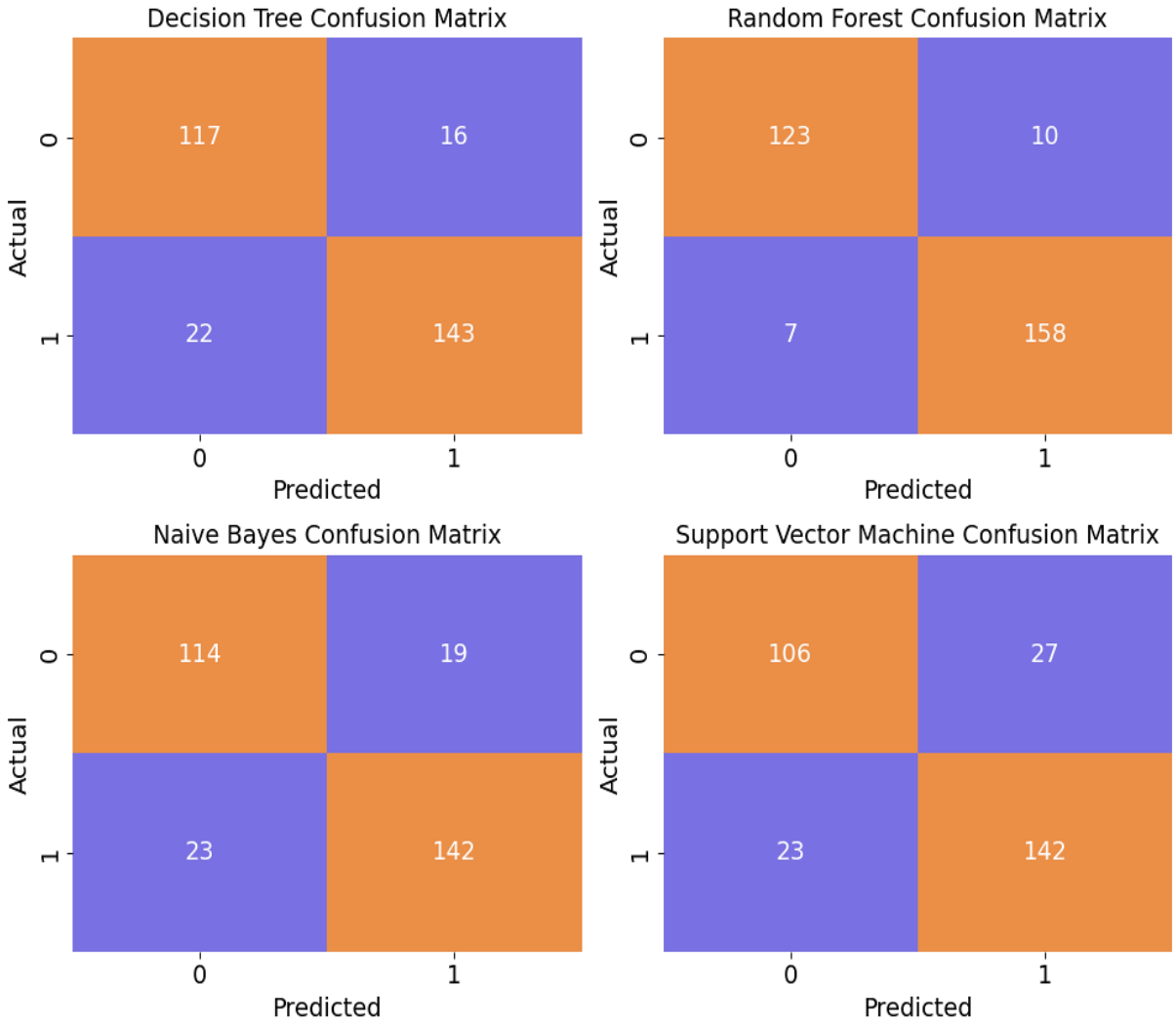


Figure 15: *Confusion Matrix for all the Models*

- **Area Under Curve (AUC) - Receiver Operating Characteristics (ROC): "**The ROC - AUC curve is a graphical representation of a classifier's performance. The ROC curve plots the true positive rate (recall) against the false positive rate (1-specificity). The AUC represents the degree or measure of separability achieved by the model. A higher AUC indicates a better-performing model". (*Kumar, 2024*)

$$\text{AUC} = \int_0^1 \text{ROC}(x)dx$$

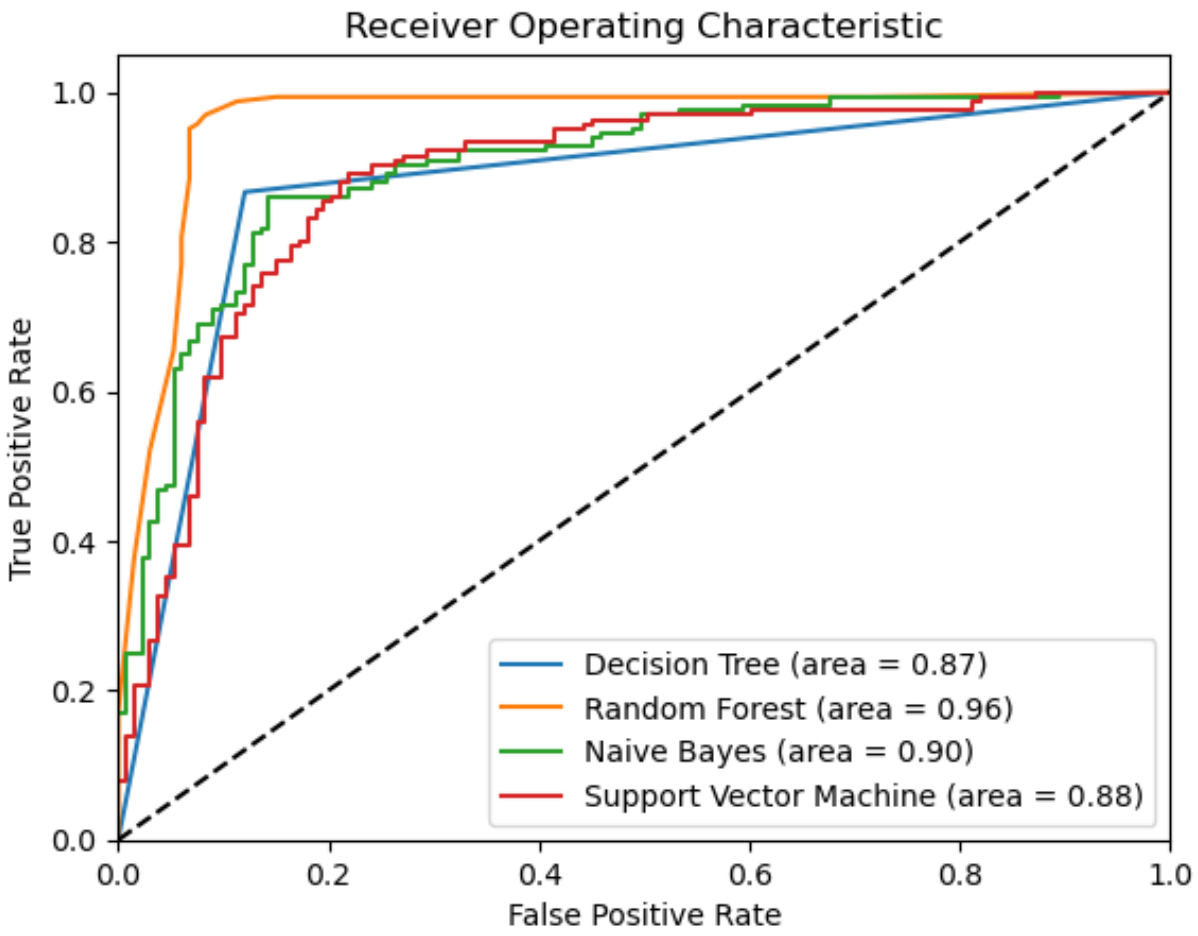


Figure 16: *ROC - AUC Curve for all the models*

Figure 16 above is a Receiver Operating Characteristic (ROC) curve. It is used to evaluate the performance of different classification models. A higher AUC indicates a better-performing model. The closer the curve follows the left-hand border and then the top border of the ROC space, the better the model. A lower AUC indicates a poorer-performing model. Curves closer to the diagonal line indicate models that are not performing better than random guessing. The Random Forest model shows the best performance with the highest AUC of 0.96, followed by Naive Bayes (0.90), Support Vector Machine (0.88), and Decision Tree (0.87).

4.2 Interpretation of Evaluation Metrics of Random Forest Model

The image in Figure 19 below demonstrates the Random Forest model's robust performance in classifying a given dataset, as evidenced by various evaluation metrics. The model achieves an overall accuracy of approximately 94.30%, indicating that it correctly predicts the class labels for 94.30% of the instances in the dataset. The classification report provides a detailed breakdown of the model's performance for each class. The precision, recall, and F1-score for class '0' are 0.95, 0.92, and 0.94, respectively, while for class '1', they are 0.94, 0.96, and 0.95, respectively. This suggests that the model performs slightly better in predicting class '1' than class '0'. The support values of 133 for class '0' and 165 for class '1' denote the number of true instances for each class in the test set. The overall accuracy of the model is reinforced by the macro and weighted averages of the precision, recall, and F1-score, all of which are calculated to be 0.94. These metrics provide an aggregated measure of the model's performance across both classes, confirming its reliability and consistency. The confusion matrix in Figure 15 shows that out of 298 predictions, the model correctly classified 123 instances of class '0' and 158 instances of class '1'. However, it incorrectly classified 10 instances of class '0' as class '1', and 7 instances of class '1' as class '0'. The ROC_AUC score of 0.96 indicates exceptional performance in distinguishing between the two classes, with a high true positive rate and a low false positive rate. This score underscores the model's capability to separate the classes effectively. Furthermore, the Random Forest Cross-Validation Scores, which range from 0.87 to 0.91, with a mean cross-validation accuracy of 0.89, illustrate the model's stability and generalisation ability across different subsets of the data. These scores suggest that the model maintains high performance even when evaluated on different portions of the dataset. In conclusion, the Random Forest model exhibits high proficiency in classification tasks, with solid accuracy, precision, recall, F1-scores, and ROC_AUC. The consistency of cross-validation scores further validates the model's robustness and reliability in real-world applications.

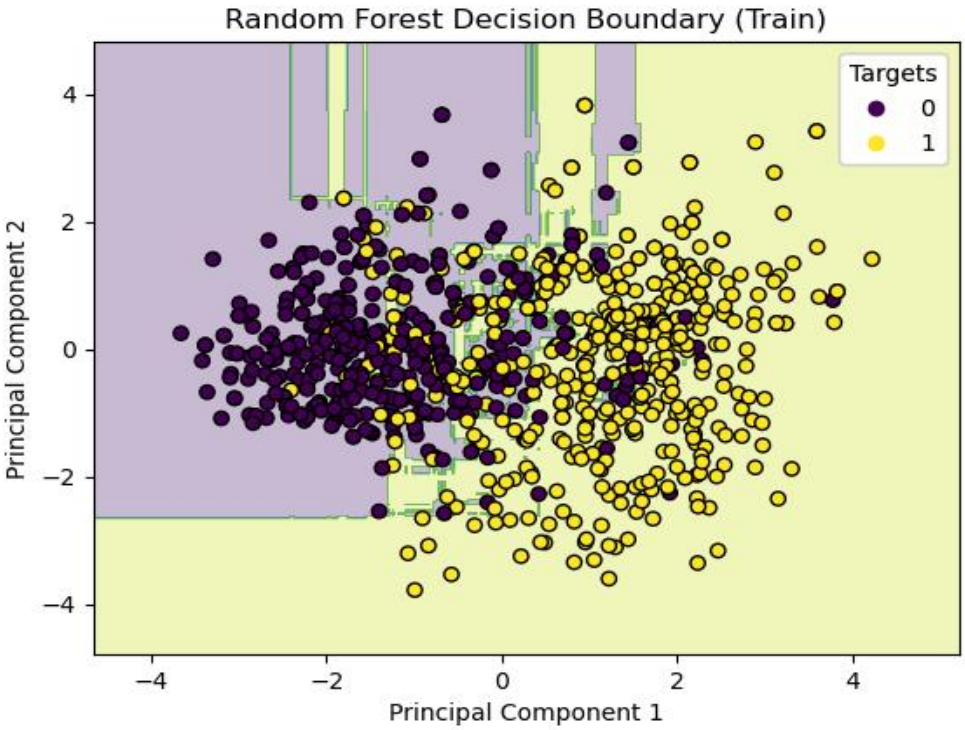


Figure 17: *Random Forest Decision Boundary diagram using the train set*

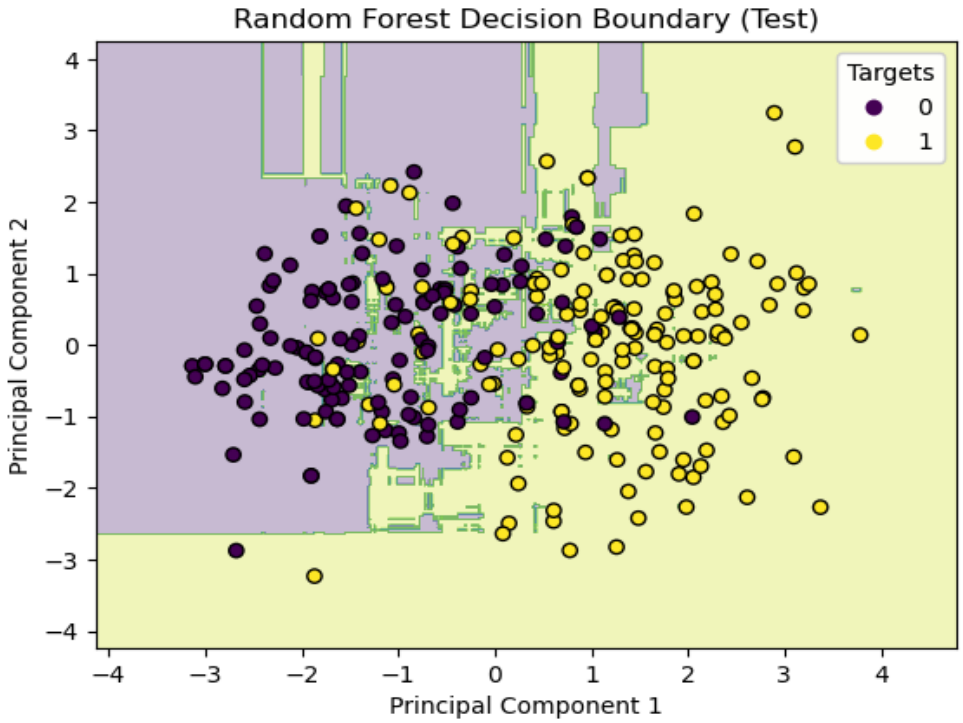


Figure 18: *Random Forest Decision Boundary diagram using the test set*

Decision boundaries are critical in understanding how machine learning models, such as Random Forest classifiers, separate different classes within a dataset. Visualising these boundaries can help us gain insights into the model's learning process, complexity, and ability to generalise to new data. The decision boundaries created by a Random Forest classifier, as illustrated in the train and test sets in Figures 17 and 18 above, provide a vivid depiction of the model's capacity to differentiate between two classes using Principal Component Analysis (PCA) components. The Random Forest model constructs a complex and non-linear decision boundary with the training set. The purple and yellow regions represent areas classified as class 0 and class 1, respectively. The intricacy of the boundary reflects the model's efforts to capture the underlying patterns in the data. The model effectively partitions the feature space, adapting to the distribution of the training points. This adaptability allows Random Forest to achieve high accuracy on training data by capturing even minor variations in the data structure. In contrast, the decision boundary in the test set provides insights into the model's generalisation capabilities. The boundary remains complex, yet the performance can vary as the model encounters unseen data. The test plot, with its scattered purple and yellow points, highlights areas where the model successfully predicts the class and areas where it might struggle. A well-generalized model will maintain similar accuracy on both train and test datasets, indicating it hasn't overfitted to the nuances of the train data. Comparing the two plots reveals Random Forest's performance in different scenarios. While the train boundary indicates the model's ability to learn from the data, the test boundary illustrates its predictive power. Ideally, the decision boundaries should be similar, showing that the model can generalise well to new data. However, discrepancies might indicate overfitting, where the model performs exceptionally on training data but less on testing data. The decision boundaries in the training and testing datasets underscore the balance between model complexity and generalisation. The Random Forest's strength lies in its ability to handle non-linear separations, yet its effectiveness is ultimately measured by how well it applies learned patterns to unseen data. These visualisations serve as a diagnostic tool, guiding adjustments to model parameters and training methods to achieve optimal performance.

```
Random Forest Model with Top 11 Features:
Accuracy: 0.9429530201342282
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.92      0.94       133
           1       0.94      0.96      0.95       165

    accuracy                           0.94       298
   macro avg       0.94      0.94      0.94       298
weighted avg       0.94      0.94      0.94       298

Confusion Matrix:
 [[123  10]
 [  7 158]]
ROC_AUC Score:
 0.9622693096377306
Random Forest Cross-Validation Scores: [0.87150838 0.90502793 0.91573034 0.90449438 0.89325843]
Random Forest Mean CV Accuracy: 0.8980038917833155
```

Figure 19: *Performance Metrics of a Random Forest Model with Top 11 Features*

Hyperparameter tuning for the Random Forest is a crucial step in optimising their performance. This parameter determines the number of trees (estimators) to be built in the ensemble model. The choice of the n_estimators value can significantly impact the model's performance, influencing its accuracy and computational efficiency. In the context of Random Forest, n_estimators specify the number of decision trees the algorithm will create. Each tree is trained on a bootstrap sample of the data, and their predictions are voted on to produce the final model output. A higher number of estimators generally leads to better performance due to reduced variance but also increases computational cost. Furthermore, to determine if hyperparameter tuning yielded a better model, we tested the code with different values of n_estimators from 10 to 50 and recorded the results. However, we found that the Random Forest classifier yielded the best accuracy of 94.30% when the n_estimator is 20.

Accuracy	N_Estimator
93.62%	10
94.30%	20
92.95%	30
93.29%	40
93.96%	50

Table 3: *Hyperparameter Tuning: N_Estimator Results*

4.3 Interpretation of the Evaluation Metrics of Decision Tree Model

The evaluation metrics of the decision tree model testify to its efficacy in classifying the heart disease dataset. The model achieves an overall accuracy of 87.25%, indicating that it correctly predicts the class labels for a substantial majority of the instances. The classification report delves deeper into the performance across the two classes, labelled 0 and 1. For class 0, the precision stands at 0.84, recall at 0.88, and the F1-score at 0.86, based on 133 instances. Class 1, on the other hand, exhibits a precision of 0.90, a recall of 0.87, and an F1-score of 0.88, calculated from 165 instances. These metrics highlight the model's balanced performance across both classes, with precision, recall, and F1-scores hovering around similar values, thus ensuring reliable classification. The overall metrics, encompassing macro and weighted averages, reiterate the model's robustness. The macro average and weighted average precision, recall, and F1-scores are 0.87, computed over 298 instances. These averages provide a holistic view of the model's performance, underscoring its consistent accuracy and reliability. The confusion matrix offers a granular view of the model's predictions. It reveals that out of 133 instances of class 0, 117 were correctly predicted, while 16 were misclassified as class 1. Similarly, for class 1, out of 165 instances, 143 were accurately predicted, with 22 misclassified as class 0. This matrix serves as a clear indicator of the model's ability to distinguish between the two classes effectively while also highlighting areas for potential improvement. Furthermore, the ROC_AUC score strongly indicates the model's discriminative power. This score, which measures the area under the receiver operating characteristic curve, underscores the model's capability to separate the classes accurately. Cross-validation ensured the model's robustness, yielding accuracy scores across five different folds. The mean cross-validation accuracy consolidates these findings, showcasing the model's consistent performance across various subsets of the data. In conclusion, the Decision Tree Model with the top 11 features demonstrates commendable performance in classifying the dataset. Its high accuracy, balanced precision, recall, and F1-scores, robust ROC_AUC score, and consistent cross-validation results collectively affirm its reliability and effectiveness as a classification tool.

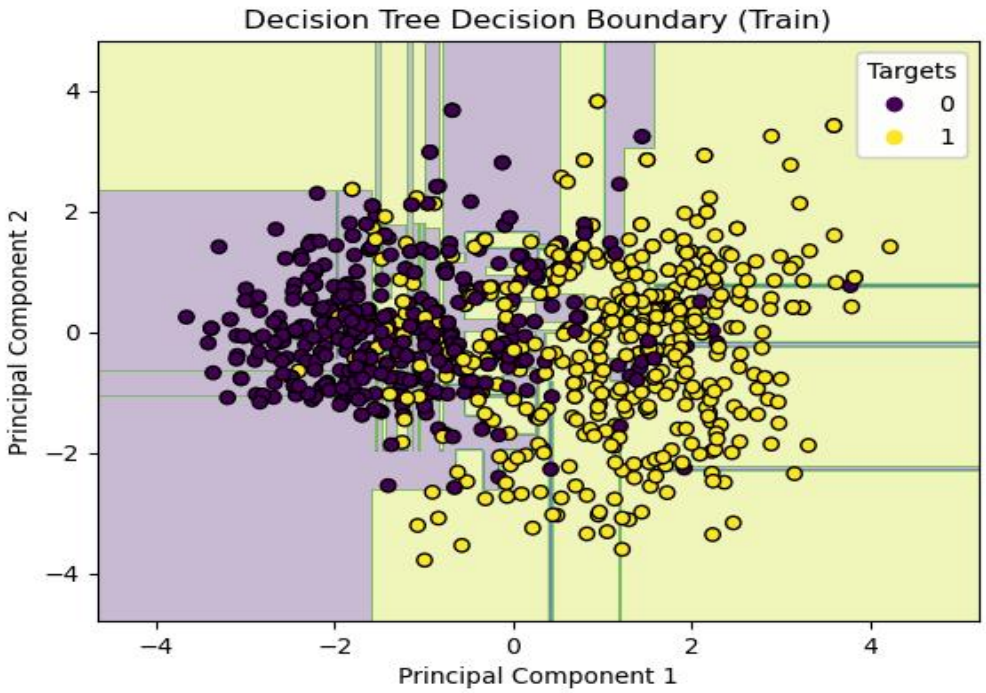


Figure 20: *Decision Tree decision boundary diagram using the train set*

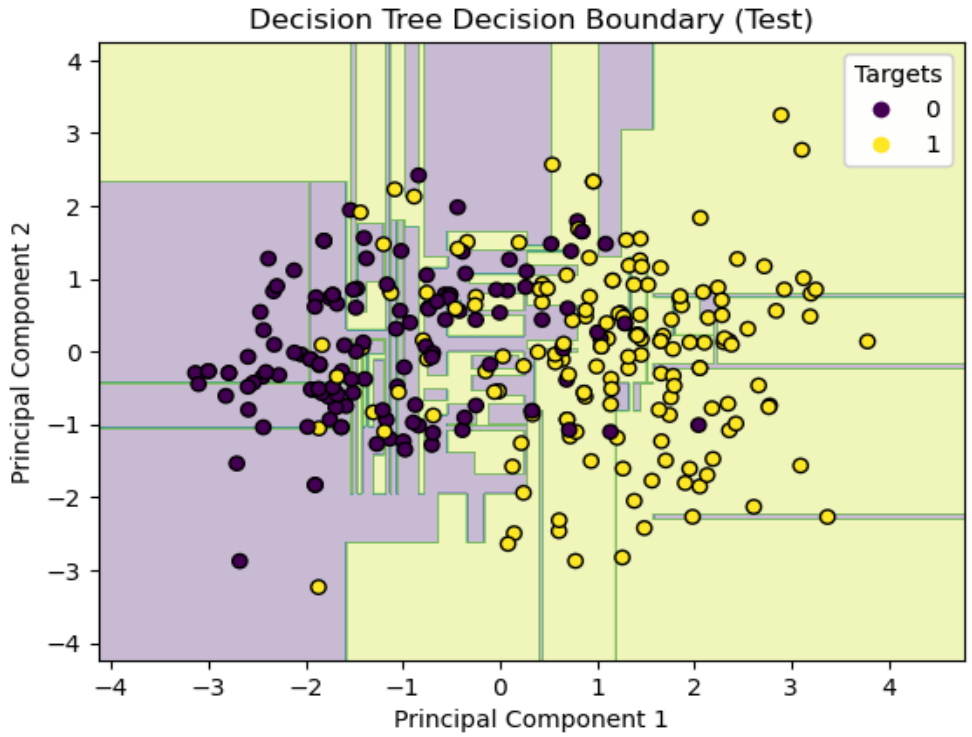


Figure 21: *Decision Tree decision boundary diagram using the test set*

The decision boundaries depicted in Figures 20 and 21 above are visual representations of the model's ability to distinguish between two target classes using principal components as features. In the train plot, the decision boundary is characterised by intricate, segmented regions. This complexity arises from the decision tree's nature, which divides the feature space into rectangular regions by recursively splitting the data based on feature values. The train plot shows how the model captures the nuances of the dataset, potentially overfitting by creating very specific boundaries that perfectly separate the train data. This can be seen in the many small, irregular regions that tightly wrap around data points, allowing the tree to classify almost all train samples correctly. However, the test plot reveals a different perspective. When applied to unseen data, the decision boundary remains complex but may not align perfectly with the true distribution of the test samples. This discrepancy highlights the decision tree's tendency to overfit, capturing noise in the train data rather than the underlying pattern. The test plot may show misclassified points where the decision boundary fails to generalise, indicating areas where the model's performance could be improved. The decision tree's boundaries reflect its hierarchical, rule-based learning approach. While it can achieve high accuracy on train data by creating detailed boundaries, the test plot underscores the importance of balancing complexity and generalisation to ensure robust performance on unseen data. This analysis suggests that while decision trees are powerful tools, careful tuning, such as pruning or feature selection, may be necessary to prevent overfitting.

```
Decision Tree Model with Top 11 Features:
Accuracy: 0.87248322147651
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       133
           1       0.90      0.87      0.88       165

    accuracy                           0.87       298
   macro avg       0.87      0.87      0.87       298
weighted avg       0.87      0.87      0.87       298

Confusion Matrix:
 [[117  16]
 [ 22 143]]
ROC_AUC Score:
 0.8731829573934837
Decision Tree Cross-Validation Scores: [0.84916201 0.8603352  0.88202247 0.84269663 0.85393258]
Decision Tree Mean CV Accuracy: 0.8576297784194338
```

Figure 22: *Performance Metrics of a Decision Model with Top 11 Features*

4.4 Interpretation of the Evaluation Metrics of Naïve Bayes Model

The evaluation metrics of the Naive Bayes Model provide a comprehensive understanding of its efficacy in classifying the heart disease dataset. The model attains an overall accuracy of 85.91%, signifying its capability to correctly predict the class labels for a substantial majority of the instances. A deeper analysis through the classification report reveals the model's performance across two classes, labelled 0 and 1. For class 0, the precision is 0.83, recall is 0.86, and the F1-score is 0.84, based on 133 instances. For class 1, the precision stands at 0.88, recall at 0.86, and the F1-score at 0.87, calculated from 165 instances. These metrics reflect the model's balanced performance, indicating reliable classification across both classes. The overall metrics, encompassing macro and weighted averages, further reinforce the model's robustness. The macro average and weighted average precision, recall, and F1-scores are 0.86, computed over 298 instances. These averages provide a holistic view of the model's performance, underscoring its consistent accuracy and reliability. The confusion matrix offers a granular perspective of the model's predictions. It reveals that out of 133 instances of class 0, 114 were correctly predicted, while 19 were misclassified as class 1. Similarly, for class 1, out of 165 instances, 142 were accurately predicted, with 23 misclassified as class 0. This matrix serves as a clear indicator of the model's ability to distinguish between the two classes effectively while also highlighting areas for potential improvement. The ROC_AUC score strongly indicates the model's discriminative power. This score, which measures the area under the receiver operating characteristic curve, underscores the model's capability to separate the classes accurately. Cross-validation ensured the model's robustness, yielding accuracy scores across five different folds. The mean cross-validation accuracy consolidates these findings, showcasing the model's consistent performance across various subsets of the data. In conclusion, the Naive Bayes Model with the top 11 features demonstrates commendable performance in classifying the dataset. Its high accuracy, balanced precision, recall, and F1-scores, robust ROC_AUC score, and consistent cross-validation results collectively affirm its reliability and effectiveness as a classification tool.

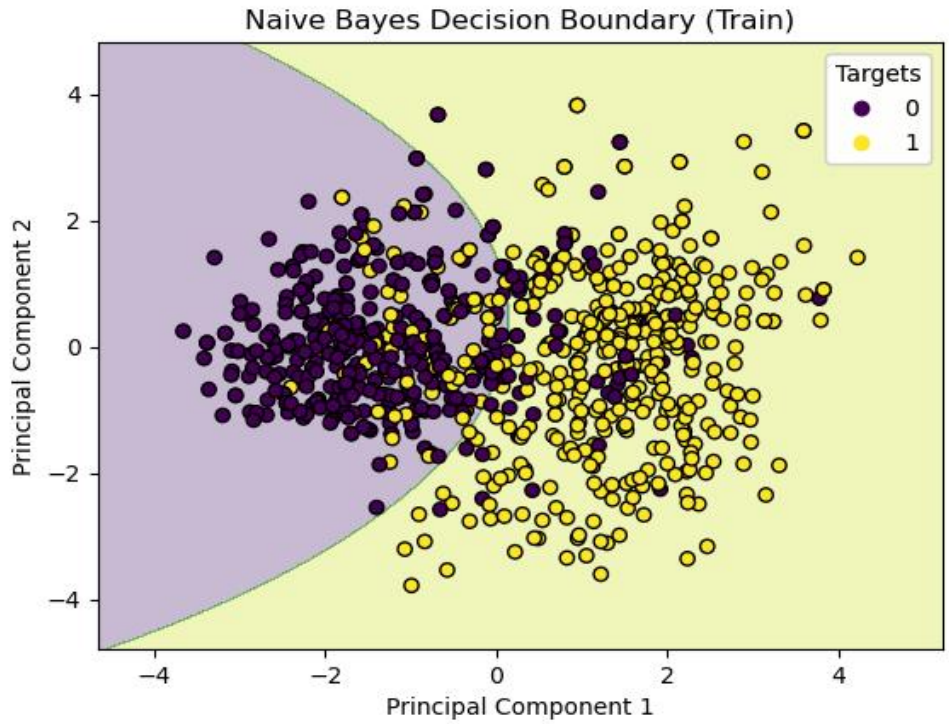


Figure 23: *Naïve Bayes decision boundary diagram using the train set*

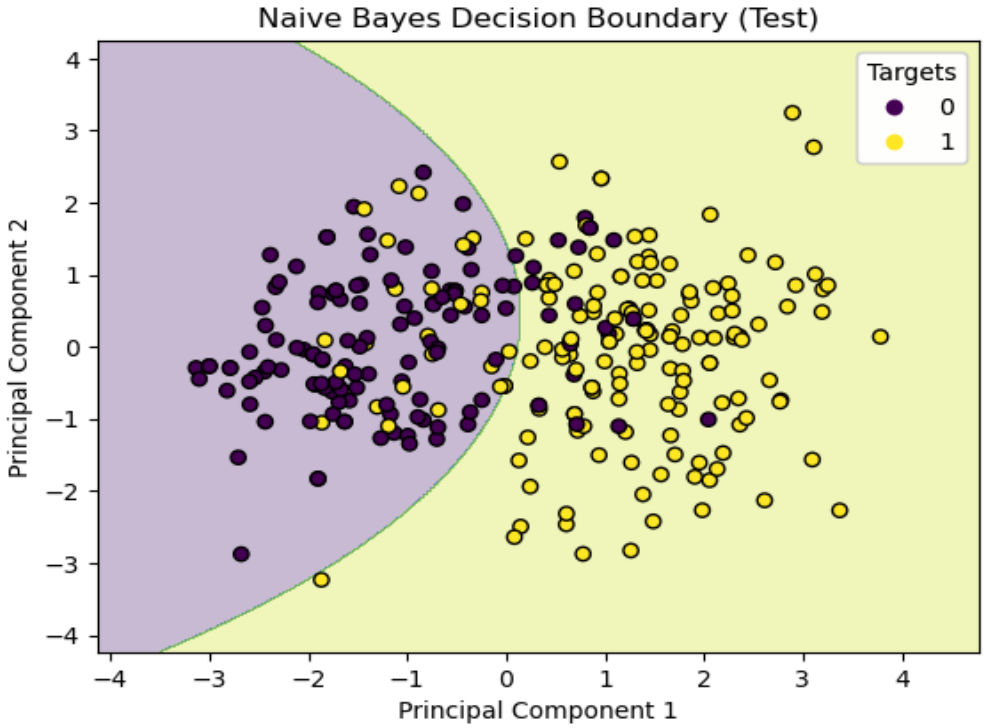


Figure 24: *Naïve Bayes decision boundary diagram using the test set*

The decision boundaries illustrated in Figures 23 and 24 above visually represent how the classifier distinguishes between two classes using principal component analysis (PCA). These plots are crucial for understanding the classifier's behaviour and effectiveness. In the train set plot shown in Figure 23, the decision boundary is depicted as a curved line that separates two regions, each associated with a different target class. The purple region corresponds to class 0, while the yellow region represents class 1. The boundary's curvature is a result of the Naive Bayes algorithm's assumption of feature independence, which allows it to model non-linear separations effectively. The distribution of data points—purple and yellow dots—indicates that the classifier generally captures the underlying patterns within the training data, with most points falling within their respective regions. Transitioning to the test set, the decision boundary maintains a similar shape, affirming the model's ability to generalise from trained to unseen data. The consistency between the train and test boundaries suggests that the classifier has learned a robust model of the data's distribution without overfitting. However, some misclassified points are visible, particularly near the boundary, highlighting areas where the model's assumptions may not perfectly align with the test data's characteristics. These decision boundaries underscore the Naive Bayes classifier's strengths and limitations. While capable of handling non-linear separations, the model's performance is contingent on the validity of its assumptions about data distribution. The visualisation aids in assessing the classifier's ability to generalise, offering insights into potential areas for model refinement.

```
Naive Bayes Model with Top 11 Features:
Accuracy: 0.8590604026845637
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       133
           1       0.88      0.86      0.87       165

    accuracy                           0.86       298
   macro avg       0.86      0.86      0.86       298
weighted avg       0.86      0.86      0.86       298

Confusion Matrix:
 [[114  19]
 [ 23 142]]
ROC_AUC Score:
 0.9005696058327636
Naive Bayes Cross-Validation Scores: [0.81564246 0.86592179 0.79775281 0.80898876 0.87078652]
Naive Bayes Mean CV Accuracy: 0.8318184671395393
```

Figure 25: *Performance Metrics of Naïve Bayes Model with Top 11 Features*

4.5 Interpretation of the Evaluation Metrics of SVM Model

The evaluation metrics of the Support Vector Machine (SVM) Model, as shown in Figure 28, offer a comprehensive insight into its capabilities in classifying the heart disease dataset. The model achieves an overall accuracy of 83.22%, reflecting its competence in correctly predicting class labels for a significant majority of the instances. The classification report provides a detailed breakdown of the model's performance across the two classes, labelled 0 and 1. For class 0, the precision is 0.82, recall is 0.80, and the F1-score is 0.81, based on 133 instances. For class 1, the precision stands at 0.84, recall at 0.86, and the F1-score at 0.85, derived from 165 instances. These metrics indicate that the model performs reliably across both classes, with slightly better precision and recall for class 1. The overall metrics, encompassing macro and weighted averages, further affirm the model's robustness. The macro average and weighted average precision, recall, and F1-scores are 0.83, computed over 298 instances. These averages provide a holistic view of the model's performance, emphasising its consistent accuracy and reliability. The confusion matrix offers a granular perspective of the model's predictions. It reveals that out of 133 instances of class 0, 106 were correctly predicted, while 27 were misclassified as class 1. Similarly, for class 1, out of 165 instances, 142 were accurately predicted, with 23 misclassified as class 0. This matrix serves as a clear indicator of the model's ability to distinguish between the two classes effectively while also highlighting areas for potential improvement. The ROC_AUC score is a testament to the model's discriminative power. This score, which measures the area under the receiver operating characteristic curve, underscores the model's ability to separate the classes accurately. Cross-validation ensured the model's robustness, yielding accuracy scores across five different folds. The mean cross-validation accuracy consolidates these findings, showcasing the model's consistent performance across various subsets of the data. In conclusion, the Support Vector Machine Model with the top 11 features demonstrates commendable performance in classifying the dataset. Its high accuracy, balanced precision, recall, and F1-scores, robust ROC_AUC score, and consistent cross-validation results collectively affirm its reliability and effectiveness as a classification tool.

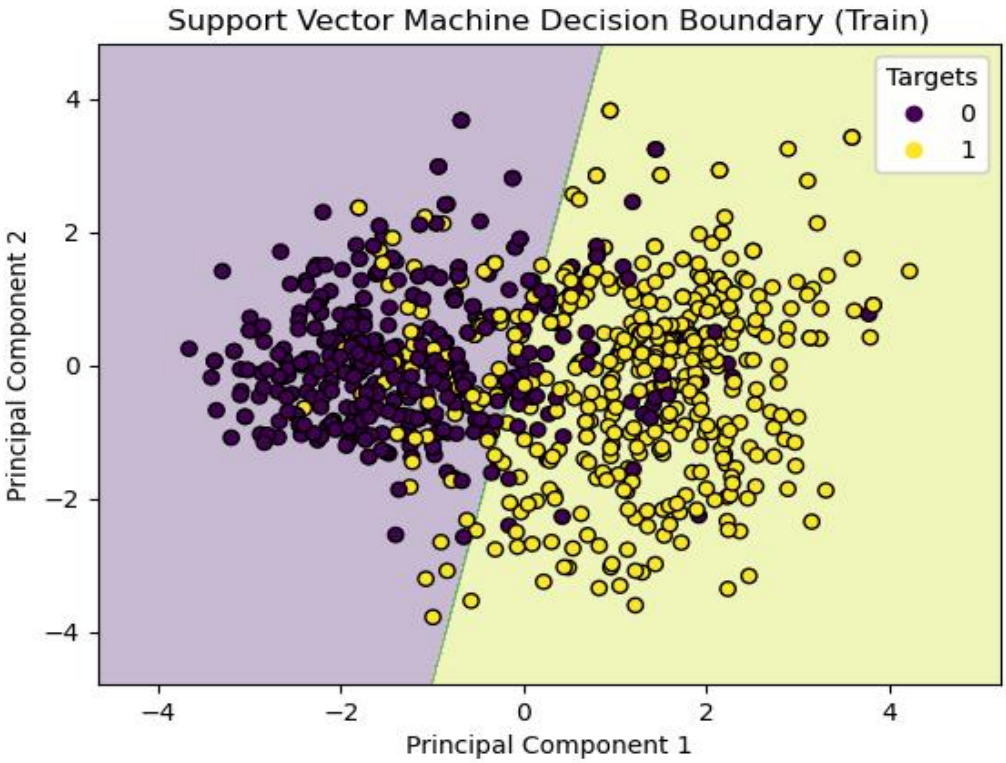


Figure 26: *SVM decision boundary diagram using the train set*

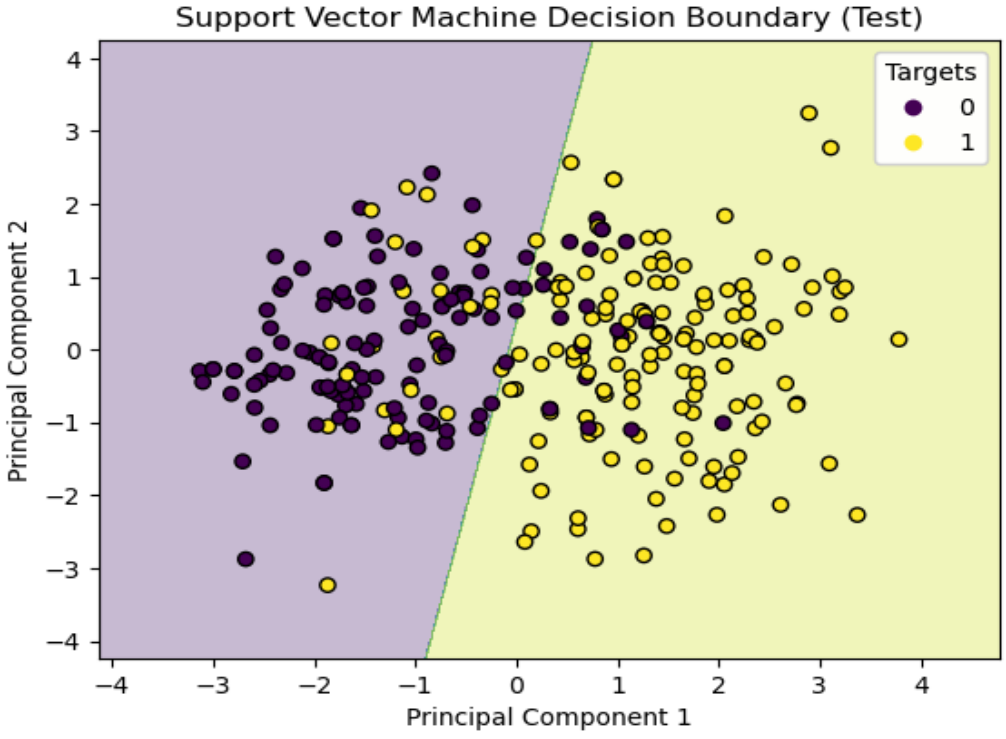


Figure 27: *SVM decision boundary diagram of the test set*

The decision boundary of a Support Vector Machine (SVM) is a critical component in understanding how this machine learning model separates data into different classes. The SVM constructs a decision boundary in the train set to maximise the margin between two classes, represented by differently coloured dots. The boundary is a linear separator that distinctly divides the two regions corresponding to the target classes 0 and 1. The SVM's objective is to find the optimal hyperplane that not only separates the classes but also is as far away from the closest data points (support vectors) of each class as possible. This margin maximisation ensures the model is robust and minimises overfitting to the train data. The decision boundary created during training is applied to unseen data for the test set. The test set's boundary closely resembles that of the training set, yet it serves the purpose of evaluating the model's generalisation ability. One can assess the model's performance in real-world scenarios by observing how well the boundary separates the test data. The alignment and separation of the data points in the test set indicate the model's effectiveness and capability to maintain a similar separation pattern as seen in the training phase. The train and test decision boundaries provide insights into the SVM's performance. While the training boundary is optimised for known data, the test boundary reflects the model's adaptability to new data. A consistent separation in both datasets suggests the model is well-trained and not overly fitted to the training data. Conversely, any significant discrepancies between the two boundaries might indicate potential overfitting or underfitting issues.

```
Support Vector Machine Model with Top 11 Features:
Accuracy: 0.8322147651006712
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.80      0.81       133
           1       0.84      0.86      0.85       165

    accuracy                           0.83       298
   macro avg       0.83      0.83      0.83       298
weighted avg       0.83      0.83      0.83       298

Confusion Matrix:
 [[106  27]
 [ 23 142]]
ROC_AUC Score:
 0.8822510822510823
Support Vector Machine Cross-Validation Scores: [0.83240223 0.86592179 0.82022472 0.82022472 0.86516854]
Support Vector Machine Mean CV Accuracy: 0.8407883999748916
```

Figure 28: *Performance Metrics of SVM Model with Top 11 Features*

4.6 Graphical User Interface (GUI)

The image in Figure 29 below is a graphical user interface (GUI) for the “Heart Disease Prediction System” designed to test user input. It was built using Tkinter in the Python library. It allows users to input specific medical and personal data required to diagnose the risk of heart disease. The fields include age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting ECG, max heart rate, exercise-induced angina, oldpeak, and ST slope. A "Predict" button is provided to process this information and predict heart disease risk. The GUI uses the Random Forest Algorithm, the model with the highest accuracy, to diagnose heart disease.

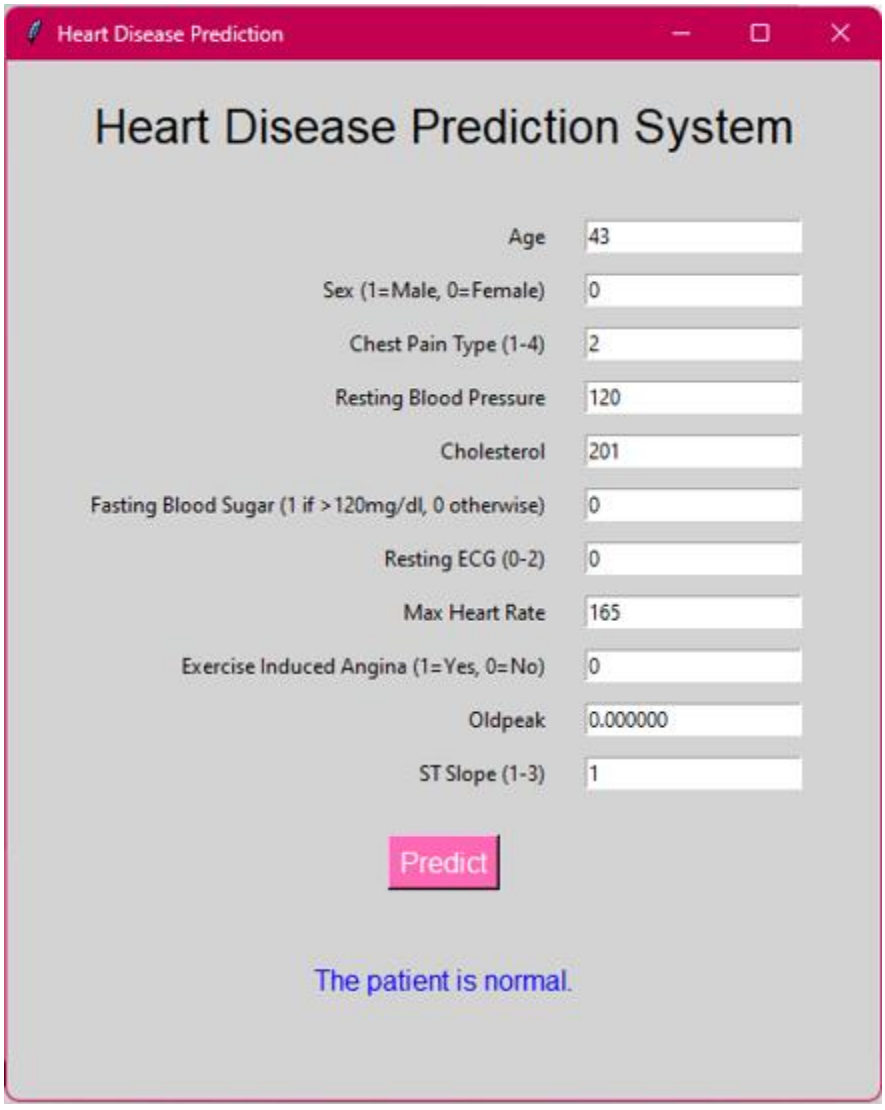


Figure 29: *GUI for the heart disease prediction system*

5 CONCLUSION & FUTURE SCOPE

5.1 CONCLUSION

This project aimed to explore the application of machine learning models in predicting heart disease, an area of immense significance given the global prevalence and impact of cardiovascular conditions. By leveraging a comprehensive dataset and employing a robust methodological framework, this study sought to identify effective predictive models and elucidate the role of various clinical features in heart disease diagnosis. The dataset utilised in this project was sourced from multiple established heart disease datasets, ensuring a rich and diverse collection of instances. Comprising 1,190 samples with 11 features, the dataset included critical variables such as age, sex, chest pain type, resting blood pressure, cholesterol levels, and maximum heart rate. A meticulous data preprocessing phase involving data cleaning, transformation, and standardisation was essential to enhance the dataset's reliability. These steps ensured the data was suitable for model training, facilitating accurate and consistent predictions. Four machine learning models were employed in this study: Random Forest, Decision Tree, Naive Bayes, and Support Vector Machine (SVM). Each model was rigorously evaluated based on its ability to classify instances of heart disease accurately. The Random Forest model emerged as the most effective, achieving an impressive accuracy of 94.30%. This model's superior performance was further corroborated by high precision, recall, F1-scores, and an exceptional ROC_AUC score, underscoring its robustness and suitability for heart disease classification. The Decision Tree model followed closely with an accuracy of 87.25%, demonstrating balanced performance across various metrics. Naive Bayes and SVM models also exhibited reliable classification capabilities, with accuracies of 85.91% and 83.22%, respectively. A critical aspect of this study was the analysis of feature importance across the different models. By aggregating feature importance scores, critical predictors of heart disease were identified, offering valuable insights for clinical research. Features such as chest pain type, cholesterol levels, and maximum heart rate were highlighted as significant contributors to model predictions. This analysis not only aids in understanding the data's underlying patterns but also suggests potential areas of focus for future clinical investigations. These findings have profound implications, particularly in the context of clinical decision support.

Machine learning models have demonstrated the potential to assist healthcare professionals in the early detection and diagnosis of heart disease, which could revolutionise patient care. Identifying high-risk individuals through predictive analytics allows for timely interventions, potentially reducing the incidence and severity of cardiovascular events. However, the study also acknowledges certain limitations and areas for future work. Despite the dataset's comprehensiveness, more extensive and diverse samples could enhance the generalisability of the findings. Additionally, while machine learning models offer high accuracy, improving their interpretability remains challenging, especially for complex models like Random Forest and SVM. Future research should focus on refining these models and addressing issues related to model transparency and integration with real-time clinical data systems. This project highlights the significant potential of machine learning in the healthcare domain, particularly for predicting heart disease. The Random Forest model, with its exceptional performance, stands out as a promising tool for clinical application. Moving forward, continued research and development in this field are essential to fully harness the benefits of advanced analytics in healthcare, ultimately improving patient outcomes and advancing our understanding of cardiovascular health. This study contributes to the growing body of evidence supporting the integration of machine learning into healthcare decision-making processes, paving the way for more personalised and effective medical care.

5.2 FUTURE SCOPE

The future scope of machine learning in heart disease prediction is expansive, offering numerous opportunities to improve patient care and outcomes significantly. The field can advance towards more accurate, reliable, and personalised healthcare solutions by integrating diverse datasets, enhancing model interpretability, applying real-time solutions, fostering interdisciplinary collaborations, and addressing ethical concerns. As technology continues to evolve, embracing these opportunities will be crucial in transforming heart disease prediction and management, ultimately reducing the global burden of cardiovascular diseases.

LIST OF REFERENCES

Agrawal, R., Imieliński, T. and Swami, A. (1993) 'Mining association rules between sets of items in large databases,' ACM SIGMOD Record, 22(2), pp. 207–216. https://doi.org/10.1145/170036.170072.

Ahmad, A.A. and Polat, H. (2023) 'Prediction of heart disease based on machine learning using Jellyfish optimization algorithm,' *Diagnostics*, 13(14), p. 2392. https://doi.org/10.3390/diagnostics13142392.

Ahsan, M.M. and Siddique, Z. (2021) 'Machine Learning-Based Heart Disease Diagnosis: A Systematic Literature Review,' *ResearchGate* [Preprint]. https://www.researchgate.net/publication/357013400_Machine_Learning-Based_Heart_Disease_Diagnosis_A_Systematic_Literature_Review.

Banjoko, A.W. and Abdulazeez, K.O. (2021) 'Efficient Data-Mining Algorithm for predicting heart disease based on an angiographic test,' *The Malaysian Journal of Medical Sciences/the Malaysian Journal of Medical Science*, 28(5), pp. 118–129. https://doi.org/10.21315/mjms2021.28.5.12.

Benjamin, E. J., Muntner, P., Alonso, A., et al. (2019). "Heart Disease and Stroke Statistics—2019 Update: A report from the American Heart Association." Circulation, 139(10) https://doi.org/10.1161/cir.0000000000000659.

Breiman, L. (2001) 'Random Forests', Machine Learning, 45(1), pp. 5-32. Available at: https://doi.org/10.1023/A:1010933404324.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2), 121-167.

Caruana, R. and Niculescu-Mizil, A., 2006, June. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning* (pp. 161-168).

Chandola, V., Banerjee, A. and Kumar, V. (2009) 'Anomaly detection,' ACM Computing Surveys, 41(3), pp. 1–58. https://doi.org/10.1145/1541880.1541882.

Chowdhury, M.E.H. *et al.* (2019) 'Real-Time Smart-Digital Stethoscope System for heart diseases monitoring,' *Sensors*, 19(12), p. 2781. https://doi.org/10.3390/s19122781.

Cortes, C. and Vapnik, V. (1995) 'Support-vector networks,' *Machine Learning*, 20(3), pp. 273–297. Available at: https://doi.org/10.1007/bf00994018.

Detrano, R. *et al.* (1989) 'International application of a new probability algorithm for the diagnosis of coronary artery disease,' ˜ *the œAmerican Journal of Cardiology*, 64(5), pp. 304–310. https://doi.org/10.1016/0002-9149(89)90524-9.

Dubey, A.K., Choudhary, K. and Sharma, R. (2021) 'Predicting Heart Disease Based on Influential Features with Machine Learning,' *Intelligent Automation and Soft Computing/Intelligent Automation & Soft Computing*, 30(3), pp. 929–943. https://doi.org/10.32604/iasc.2021.018382.

Esposito, F. *et al.* (1997) 'A comparative analysis of methods for pruning decision trees,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5), pp. 476–493. Available at: https://doi.org/10.1109/34.589207.

Fawagreh, K., Gaber, M.M. and Elyan, E. (2014) 'Random forests: from early developments to recent advancements,' *Systems Science & Control Engineering*, 2(1), pp. 602–609. https://doi.org/10.1080/21642583.2014.956265.

Fawcett, T. (2006) 'An introduction to ROC analysis,' Pattern Recognition Letters, 27(8), pp. 861–874. https://doi.org/10.1016/j.patrec.2005.10.010.

GeeksforGeeks (2020). Evaluation Metrics in Machine Learning. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/metrics-for-machine-learning-model/ [Accessed 4 Aug. 2024].

Gillis, A.S., Stedman, C. and Hughes, A. (2024). data mining. [online] Business Analytics. Available at: https://www.techtarget.com/searchbusinessanalytics/definition/data-mining [Accessed 1 Sep. 2024].

Haykin, S. (2009). Neural networks and learning machines. Prentice Hall.

Ilayaraja M. and Meyyappan T. (2015) 'Efficient data mining method to predict the risk of heart diseases through frequent itemsets,' *Procedia Computer Science*, 70, pp. 586–592. https://doi.org/10.1016/j.procs.2015.10.040.

Jain, A.K., Murty, M.N. and Flynn, P.J. (1999) 'Data clustering,' ACM Computing Surveys, 31(3), pp. 264–323. https://doi.org/10.1145/331499.331504.

Jan, M. *et al.* (2018) 'Ensemble approach for developing a smart heart disease prediction system using classification algorithms,' *Research Reports in Clinical Cardiology*, Volume 9, pp. 33–45. https://doi.org/10.2147/rrcc.s172035.

Johnson, K.W. *et al.* (2017) 'Enabling precision cardiology through multiscale biology and systems medicine,' *JACC. Basic to Translational Science*, 2(3), pp. 311–327. https://doi.org/10.1016/j.jacbts.2016.11.010.

Kohavi, R. (1995). *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*. Available at: https://www.ijcai.org/Proceedings/95-2/Papers/016.pdf.

Kumar, V.V. (2024) Evaluating machine learning models-metrics and techniques. https://www.aiacceleratorinstitute.com/evaluating-machine-learning-models-metrics-and-techniques/.

Mahdavinejad, M.S. *et al.* (2018) 'Machine learning for Internet of things data analysis: a survey,' *Digital Communications and Networks*, 4(3), pp. 161–175. https://doi.org/10.1016/j.dcan.2017.10.002.

Manu Siddhartha (2020). Heart Disease Dataset (Comprehensive). [online] IEEE DataPort. Available at: https://ieee-dataport.org/open-access/heart-disease-dataset-comprehensive [Accessed 1 Jan. 2024].

Mayo Clinic. (2024). *Heart disease - Symptoms and causes*. [online] Available at: https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118 [Accessed 27 Aug. 2024].

Mitchell, T. M. (1997). Machine learning. McGraw-Hill.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Introduction to linear regression analysis. John Wiley & Sons.

Murphy, K. P. (2012) Machine Learning: A Probabilistic Perspective. Cambridge, MA: MIT Press

Nagavelli, U., Samanta, D. and Chakraborty, P. (2022) 'Machine Learning Technology-Based Heart Disease Detection Models,' *Journal of Healthcare Engineering*, 2022, pp. 1–9. https://doi.org/10.1155/2022/7351061.

Niranjan Appaji (2024). A Tale of Two Metrics: Gini Impurity and Entropy in Decision Tree Algorithms. [online] Medium. Available at: https://medium.com/@niranjan.appaji/a-tale-of-two-metrics-gini-impurity-and-entropy-in-decision-tree-algorithms-7f61d46baacc [Accessed 20 Aug. 2024].

Powers, D 2011, 'Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation', *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37-63.

Quinlan, J. R. (1986) 'Induction of Decision Trees', Machine Learning, 1(1), pp. 81-106. Available at: https://doi.org/10.1023/A:1022643204877.

Rajkomar, A. *et al.* (2018) 'Scalable and accurate deep learning with electronic health records,' *Npj Digital Medicine*, 1(1). https://doi.org/10.1038/s41746-018-0029-1.

Saini, A. (2021). *Guide on Support Vector Machine (SVM) Algorithm*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/ [Accessed 21 Aug. 2024].

Sarica, A., Cerasa, A. and Quattrone, A. (2017) 'Random Forest Algorithm for the Classification of neuroimaging data in Alzheimer's Disease: a Systematic review,' *Frontiers in Aging Neuroscience*, 9. https://doi.org/10.3389/fnagi.2017.00329.

Sarker, I.H. (2021) 'Machine learning: algorithms, Real-World applications and research directions,' *SN Computer Science/SN Computer Science*, 2(3). https://doi.org/10.1007/s42979-021-00592-x.

Schonlau, M. and Zou, R.Y. (2020) 'The random forest algorithm for statistical learning,' *The Stata Journal Promoting Communications on Statistics and Stata*, 20(1), pp. 3–29. https://doi.org/10.1177/1536867x20909688.

Sutton, R. S., and Barto, A. G. (2018) Reinforcement Learning: An Introduction. 2nd edn. Cambridge, MA: MIT Press.

Svetnik, V. *et al.* (2003) 'Random Forest: a classification and regression tool for compound classification and QSAR modeling,' *Journal of Chemical Information and Computer Sciences*, 43(6), pp. 1947–1958. https://doi.org/10.1021/ci034160g.

Uddin, K.M.M. *et al.* (2023) 'Machine learning-based approach to the diagnosis of cardiovascular vascular disease using a combined dataset,' *Intelligence-based Medicine*, 7, p. 100100. https://doi.org/10.1016/j.ibmed.2023.100100.

Zhang, H., 2004. The optimality of naive Bayes. Aa, 1(2), p.3.

APPENDICES

The analysis in Figure 30 below reveals the key factors, including sex, chest pain type, fasting blood sugar, resting ECG results, exercise-induced angina, and ST slope, to elucidate their relationships with heart disease prevalence. There is a significant disparity between males and females concerning heart disease prevalence. Males exhibit a higher incidence of heart disease compared to females. This trend highlights the need for targeted interventions and awareness campaigns, mainly focusing on males who appear to be at a greater risk. Chest pain type is a critical indicator in diagnosing heart disease. The data indicates that individuals experiencing asymptomatic chest pain have a markedly higher association with heart disease. In contrast, those presenting with atypical angina or non-anginal pain tend to have a lower incidence of heart disease. These findings underscore the importance of thorough medical evaluations of chest pain types, even when symptoms are not overtly severe, as they could signify underlying heart conditions. Elevated fasting blood sugar levels, specifically those exceeding 120 mg/dl, are strongly associated with heart disease. Individuals with high fasting blood sugar levels show a significantly higher occurrence of heart disease compared to those with normal levels. This relationship emphasises the importance of maintaining healthy blood sugar levels through diet, exercise, and, when necessary, medication to mitigate the risk of developing heart disease. Resting ECG readings provide valuable insights into heart health. The presence of ST-T wave abnormalities is linked with a higher count of heart disease cases, whereas normal ECG readings are more frequently observed in individuals without heart disease. Exercise-induced angina is another critical factor associated with heart disease. Individuals who experience angina during physical exertion are more likely to have heart disease. Conversely, those without exercise-induced angina are predominantly found to be free of heart disease. The ST slope observed on an ECG is a vital diagnostic parameter. A flat ST slope is associated with a higher prevalence of heart disease, whereas an upsloping ST slope is more common among individuals without the condition. Analysing these factors provides valuable insights into the risk and prevalence of heart disease. Identifying high-risk groups based on sex, chest pain type, fasting blood sugar levels, resting ECG results, exercise-induced angina, and ST slope can lead to more personalised and effective preventive measures.

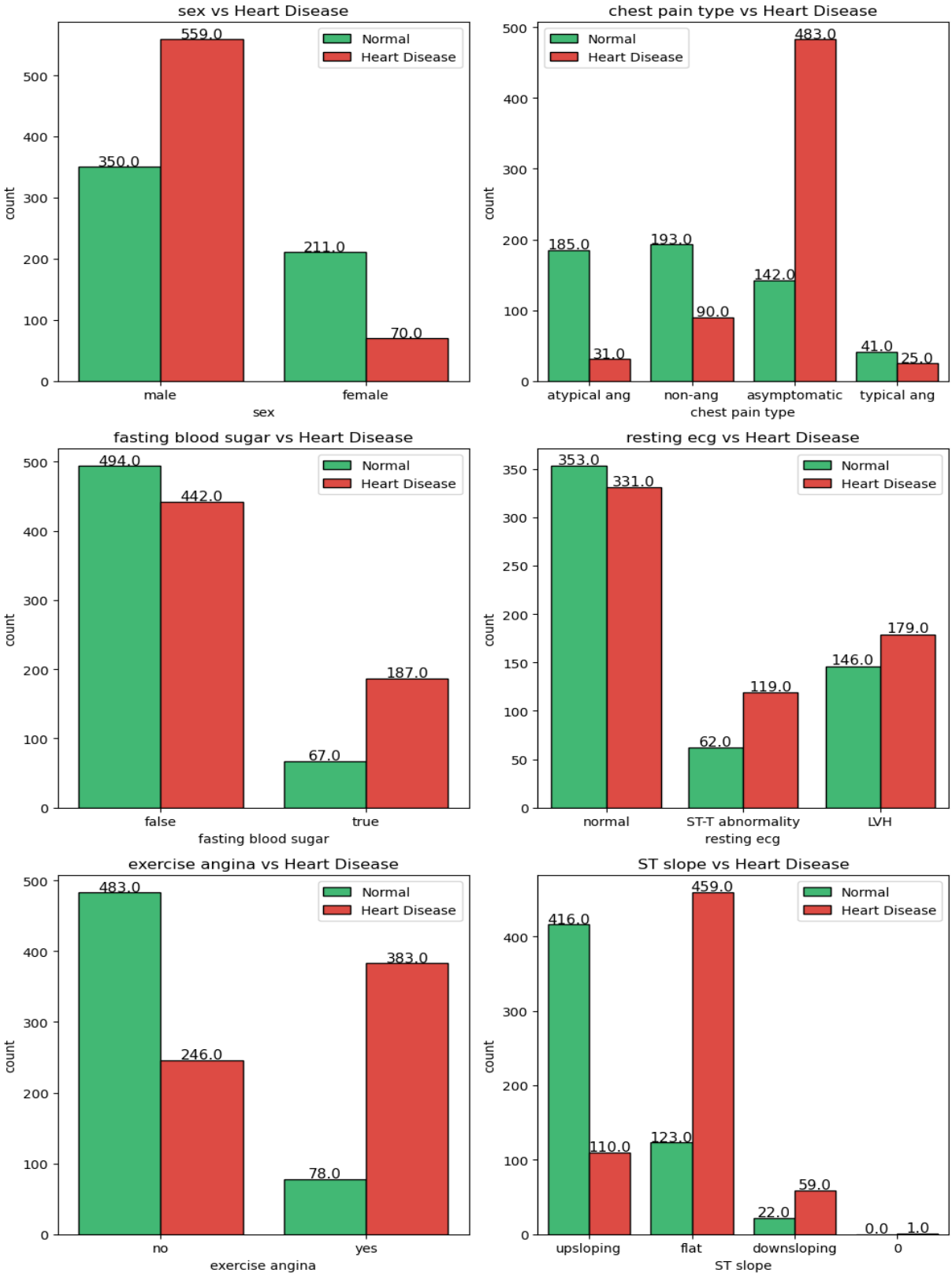


Figure 30: *Visualisation of the categorical features in the dataset*

Scenario:

Imagine we have a dataset of weather conditions and whether a sports event should be played.

Weather	Play
Sunny	Yes
Rainy	No
Sunny	Yes
Overcast	Yes
Rainy	No

Table 4: *Dataset of sports events in different weather conditions*

Entropy and Gini Impurity Calculation for Root Node

Probability:

Yes: 3/5 = 0.6

No: 2/5 = 0.4

Entropy:

$\boldsymbol{H} = -[0\,6\log_2 0.6 + 0\,4\log_2 0.4]$

$\boldsymbol{H} \approx 0.971$

Gini Index:

$\boldsymbol{G} = 1 - (0.6^2 + 0.4^2)$

$\boldsymbol{G} = 1 - (0.36 + 0.16)$

$\boldsymbol{G} = 0.48$

These calculations help decide the best attribute to split on at each step of building the decision tree, aiming to reduce impurity (Entropy or Gini Index) and improve classification accuracy.

Principal Component Analysis (PCA)

Principal Component Analysis is a powerful statistical technique used for dimensionality reduction. PCA is applied to the heart disease dataset to reduce complexity while retaining the most critical features for analysis and visualisation. The process begins by importing the PCA module from sklearn.decomposition, part of the Scikit-learn library. This library provides various machine-learning tools, among which PCA is beneficial for simplifying complex datasets. The code initialises PCA with n_components=2, indicating that the data will be reduced to two dimensions. This choice is often made to facilitate visualisation, as two-dimensional data can be easily plotted and understood. The PCA model is then fitted to the training data, X_train. During this fitting process, PCA identifies the directions, or principal components, that maximise the variance in the dataset. These components are orthogonal, meaning they are uncorrelated and provide a new set of axes to represent the data. This is a crucial aspect of PCA, as it ensures that each principal component captures unique information about the variance in the dataset. Once the principal components are determined, the original training data is projected onto these new axes through the fit_transform method. This transformation results in a new dataset, X_train_pca, which has reduced dimensions while preserving the most significant variance. The test data, X_test, is similarly transformed using the same PCA model. This ensures consistency in the data representation and allows for an accurate evaluation of predictive models. The reduced dimensionality provided by PCA not only simplifies the dataset but also aids in visualising the decision boundaries of these models.

```
Aggregate Feature Importances Across Models:
                Feature  Importance
10            ST slope    0.291303
2       chest pain type    0.157825
9               oldpeak    0.139602
4           cholesterol    0.120273
1                   sex    0.106994
8       exercise angina    0.106546
7        max heart rate    0.089306
5   fasting blood sugar    0.083005
0                   age    0.051875
3         resting bp s     0.040011
6          resting ecg     0.024991
```

Code Implementation

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import tkinter as tk
from tkinter import messagebox, ttk
import numpy as np
import joblib
from sklearn.model_selection import train_test_split, RepeatedStratifiedKFold, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,
 precision_recall_curve, roc_auc_score, roc_curve
```

```
dataset = pd.read_csv('heart_statlog_cleveland_hungary_final.csv')
print('The shape of the data is ', dataset.shape)
```

```
col = list(dataset.columns)
categorical_features = []
numerical_features = []
for i in col:
    if len(dataset[i].unique()) > 6:
        numerical_features.append(i)
    else:
        categorical_features.append(i)

print('Categorical Features :',*categorical_features)
print('Numerical Features :',*numerical_features)
```

```
df1 = dataset.copy()
df1['sex'] = df1['sex'].replace({1: 'male', 0: 'female'})
df1['chest pain type'] = df1['chest pain type'].replace({1: 'typical ang', 2: 'atypical ang',
 3: 'non-ang', 4: 'asymptomatic'})
df1['fasting blood sugar'] = df1['fasting blood sugar'].replace({1: 'true', 0: 'false'})
df1['resting ecg'] = df1['resting ecg'].replace({0: 'normal', 1: 'ST-T abnormality', 2: 'LVH'})
df1['exercise angina'] = df1['exercise angina'].replace({1: 'yes', 0: 'no'})
df1['ST slope'] = df1['ST slope'].replace({1: 'upsloping', 2: 'flat', 3: 'downsloping'})
df1['target'] = df1['target'].replace({1: 'heart disease', 0: 'no heart disease'})
```

```
d = list(df1['target'].value_counts())
circle = [d[1] / sum(d) * 100, d[0] / sum(d) * 100]
colors = ['#2ecc71', '#f7342a']
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20, 5))

plt.subplot(1, 2, 1)
plt.pie(circle, labels=['Normal', 'Heart Disease'], autopct='%1.1f%%', startangle=90,
 explode=(0.1, 0), colors=colors,
        wedgeprops={'edgecolor': 'black', 'linewidth': 1, 'antialiased': True},
         textprops={'fontsize': 14})
plt.title('Heart Disease (%)', fontsize=18)

# Count plot for heart disease cases
plt.subplot(1, 2, 2)
ax = sns.countplot(x='target', data=df1, palette=colors, edgecolor='black')
for rect in ax.patches:
    ax.text(rect.get_x() + rect.get_width() / 2, rect.get_height() + 2, rect.get_height(),
     horizontalalignment='center', fontsize=14)
ax.set_xticklabels(['Normal', 'Heart Disease'], fontsize=14)
ax.set_xlabel('Target', fontsize=16)
ax.set_ylabel('Count', fontsize=16)
plt.title('Cases of Heart Disease', fontsize=18)

plt.show()
```

```
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(10, 15))
for i in range(len(categorical_features) - 1):
    plt.subplot(3, 2, i + 1)
    ax = sns.countplot(x=categorical_features[i], data=df1, hue="target", palette=colors,
     edgecolor='black')
    for rect in ax.patches:
        ax.text(rect.get_x() + rect.get_width() / 2, rect.get_height() + 2, rect.get_height(),
         horizontalalignment='center', fontsize=11)
    title = categorical_features[i] + ' vs Heart Disease'
    plt.legend(['Normal', 'Heart Disease'])
    plt.title(title)

plt.tight_layout()
plt.show()
```

```python
# Data Preprocessing function
def data_preprocessing(dataset):
    # Handle missing values
    missing_values = dataset.isnull().sum()

    # Replace the outlier with the mode of the column
    mode_st_slope = dataset['ST slope'].mode()[0]
    dataset['ST slope'] = dataset['ST slope'].replace(0, mode_st_slope)
    print("Outlier replaced with mode:", mode_st_slope)

    # Split the data into features and target
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

    # Scale the features
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)

    return X_train, X_test, y_train, y_test, missing_values

# Perform data preprocessing
X_train, X_test, y_train, y_test, missing_values = data_preprocessing(dataset)

# Print dataset information and missing values
print("The dataset information:")
print(dataset.info())
print("\nMissing values in each column:")
print(missing_values)
```

```
# Initialize models to evaluate
models = {
    'Decision Tree': DecisionTreeClassifier(criterion='entropy', random_state=0),
    'Random Forest': RandomForestClassifier(n_estimators=20, criterion='entropy',
     random_state=0),
    'Naive Bayes': GaussianNB(),
    'Support Vector Machine': SVC(kernel='linear', random_state=0, probability=True)
}

# Dictionary to store feature importances for each model
feature_importance_info = {model_name: None for model_name in models.keys()}

# Evaluate models and get feature importances
for model_name, model in models.items():
    model.fit(X_train, y_train)

    if hasattr(model, 'feature_importances_'):
        importances = model.feature_importances_
    elif hasattr(model, 'coef_'):  # For models like SVM with coefficients
        importances = np.abs(model.coef_).mean(axis=0)
    else:
        importances = None

    feature_importance_info[model_name] = importances
```

```
# Aggregate feature importances across models
feature_names = dataset.columns[:-1]
aggregate_importances = np.zeros(len(feature_names))
for model_name, importances in feature_importance_info.items():
    if importances is not None:
        aggregate_importances += importances

# Normalize aggregated importances
aggregate_importances /= len(models)

# Sort indices based on aggregated importances
indices = np.argsort(aggregate_importances)[::-1]

# Create DataFrame to display aggregated feature importances
aggregate_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance':
 aggregate_importances})
aggregate_importance_df = aggregate_importance_df.sort_values(by='Importance', ascending=False)

# Display aggregated feature importances
print("\nAggregate Feature Importances Across Models:")
print(aggregate_importance_df)
```

```
# Function to evaluate model performance with selected features
def evaluate_model_with_selected_features(model, X_train, X_test, y_train, y_test, indices,
 num_features):
    top_features = indices[:num_features]
    X_train_selected = X_train[:, top_features]
    X_test_selected = X_test[:, top_features]

    model.fit(X_train_selected, y_train)
    y_pred = model.predict(X_test_selected)

    accuracy = accuracy_score(y_test, y_pred)
    return accuracy

# Evaluate models with increasing number of features
results = {model_name: [] for model_name in models.keys()}
for num_features in range(1, len(indices) + 1):
    for model_name, model in models.items():
        accuracy = evaluate_model_with_selected_features(model, X_train, X_test, y_train,
         y_test, indices, num_features)
        results[model_name].append(accuracy)

# Plotting the feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=aggregate_importance_df, palette='viridis')
plt.title('Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```

```
# Plotting model performance with increasing number of features
plt.figure(figsize=(12, 8))
for model_name, accuracies in results.items():
    plt.plot(range(1, len(indices) + 1), accuracies, label=model_name, marker='o')
plt.title('Model Performance with Increasing Number of Features')
plt.xlabel('Number of Features')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
# Dictionary to store highest accuracy and corresponding number of features for each model
best_accuracy_info = {model_name: {'accuracy': 0, 'num_features': 0} for model_name in
 models.keys()}

# Evaluate models with increasing number of features
for num_features in range(1, len(indices) + 1):
    for model_name, model in models.items():
        accuracy = evaluate_model_with_selected_features(model, X_train, X_test, y_train,
         y_test, indices, num_features)
        if accuracy > best_accuracy_info[model_name]['accuracy']:
            best_accuracy_info[model_name]['accuracy'] = accuracy
            best_accuracy_info[model_name]['num_features'] = num_features

# Print the results
for model_name, info in best_accuracy_info.items():
    print(f'Model: {model_name}, Highest Accuracy: {info["accuracy"]}, Number of Features:
     {info["num_features"]}')
```

```
# Select top 11 features
top_n = 11
top_features = indices[:top_n]
X_train_selected = X_train[:, top_features]
X_test_selected = X_test[:, top_features]

# Store model results
model_names = []
accuracies = []
conf_matrices = []
class_reports = []
precision_recall_curves = []
roc_auc_scores = []

# Train and evaluate each model with selected features
for name, model in models.items():
    model.fit(X_train_selected, y_train)
    y_pred = model.predict(X_test_selected)

    print(f'{name} Model with Top {top_n} Features:')
```

```
    # Accuracy
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
    print('Accuracy:', accuracy)

    # Classification Report
    class_report = classification_report(y_test, y_pred, output_dict=True)
    class_reports.append(class_report)
    print('Classification Report:\n', classification_report(y_test, y_pred))

    # Confusion Matrix
    conf_matrix = confusion_matrix(y_test, y_pred)
    conf_matrices.append(conf_matrix)
    print('Confusion Matrix:\n', conf_matrix)

    # Precision-Recall Curve
    precision, recall, _ = precision_recall_curve(y_test, y_pred)
    precision_recall_curves.append((precision, recall))

    # ROC AUC Score
    if hasattr(model, "predict_proba"):
        y_scores = model.predict_proba(X_test_selected)[:, 1]
    else:
        y_scores = model.decision_function(X_test_selected)

    roc_auc = roc_auc_score(y_test, y_scores)
    roc_auc_scores.append(roc_auc)
    print('ROC_AUC Score:\n', roc_auc_score(y_test, y_scores))
```

```
    # Cross-validation
    cv_scores = cross_val_score(model, X_train_selected, y_train, cv=5, scoring='accuracy')
    print(f'{name} Cross-Validation Scores:', cv_scores)
    print(f'{name} Mean CV Accuracy: {np.mean(cv_scores)}')
    print('\n' + '-'*30 + '\n')

    # Plot ROC curve
    fpr, tpr, _ = roc_curve(y_test, y_scores)
    plt.plot(fpr, tpr, label=f'{name} (area = {roc_auc:.2f})')

    model_names.append(name)  # Append the model name to the list

plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()
```

```python
for name, model in models.items():
    if hasattr(model, 'support_'):
        num_support_vectors_per_class = model.n_support_
        print(f'Number of support vectors for each class: {num_support_vectors_per_class}')
    else:
        print(f'{name} does not use support vectors.')
```

```python
# Apply PCA to reduce to 2 dimensions
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

# Plot decision boundary function
def plot_decision_boundary(model, X, y, title):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                         np.arange(y_min, y_max, 0.01))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.3, cmap='viridis')
    scatter = plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', cmap='viridis')
    plt.title(title)
    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    legend1 = plt.legend(*scatter.legend_elements(), title="Targets")
    plt.gca().add_artist(legend1)
    plt.show()

# Train and plot decision boundaries for each model
for name, model in models.items():
    model.fit(X_train_pca, y_train)
    plot_decision_boundary(model, X_train_pca, y_train, f'{name} Decision Boundary (Train)')
    plot_decision_boundary(model, X_test_pca, y_test, f'{name} Decision Boundary (Test)')
```

```python
# Plot the Precision-Recall curves
plt.figure(figsize=(10, 6))
for i, (precision, recall) in enumerate(precision_recall_curves):
    plt.plot(recall, precision, label=model_names[i])

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='best')
plt.show()
```

```
# Set up Seaborn color palette for better color harmony
colors = sns.color_palette("colorblind", len(model_names))

# Increase figure size
plt.figure(figsize=(12, 7))

# Plot the accuracy comparison
bars = plt.bar(model_names, accuracies, width=0.6, color=colors)

# Add values on top of the bars for better readability
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval + 0.01, f'{yval:.2f}',
             ha='center', va='bottom', fontsize=12, color='black')

# Add labels, title, and customize appearance
plt.xlabel('Model', fontsize=14, fontweight='bold')
plt.ylabel('Accuracy', fontsize=14, fontweight='bold')
plt.title('Accuracy Comparison of the Models', fontsize=18, fontweight='bold', color='#4f4f4f')

# Set y-axis limit and add gridlines
plt.ylim([0, 1.1])
plt.gca().yaxis.grid(True, linestyle='--', alpha=0.6)

# Remove top and right spines for a cleaner look
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Adjust layout to prevent clipping
plt.tight_layout()

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

```
# Plot the confusion matrix for each model
fig, axes = plt.subplots(2, 2, figsize=(10, 8))
axes = axes.ravel()
colors = ['#7971e3', '#eb8e46']
font_size = 14

for i, (name, conf_matrix) in enumerate(zip(model_names, conf_matrices)):
    sns.heatmap(conf_matrix, annot=True, fmt='d', ax=axes[i], cmap= colors, cbar=False,
     annot_kws={"size": font_size})
    axes[i].set_title(f'{name} Confusion Matrix', fontsize=font_size)
    axes[i].set_xlabel('Predicted', fontsize=font_size)
    axes[i].set_ylabel('Actual', fontsize=font_size)
    axes[i].tick_params(axis='both', which='major', labelsize=font_size)

plt.tight_layout()
plt.show()
```

```python
# Extract precision for each class from the class reports
precision_class_0 = [
    report['0']['precision'] for report in class_reports
]
precision_class_1 = [
    report['1']['precision'] for report in class_reports
]

# Plotting function for precision by class
def plot_precision_by_class(model_names, precision_class_0, precision_class_1):
    bar_width = 0.35
    index = np.arange(len(model_names))

    plt.figure(figsize=(12, 7))
    bars1 = plt.bar(index, precision_class_0, bar_width, label='Class 0')
    bars2 = plt.bar(index + bar_width, precision_class_1, bar_width, label='Class 1')

    plt.xlabel('Model', fontsize=14, fontweight='bold')
    plt.ylabel('Precision', fontsize=14, fontweight='bold')
    plt.title('Precision by Class for Each Model', fontsize=18, fontweight='bold',
     color='#4f4f4f')
    plt.xticks(index + bar_width / 2, model_names)
    plt.ylim([0, 1.1])
    plt.legend()

    for bars in [bars1, bars2]:
        for bar in bars:
            yval = bar.get_height()
            plt.text(bar.get_x() + bar.get_width() / 2, yval + 0.01, f'{yval:.2f}',
                     ha='center', va='bottom', fontsize=12, color='black')

    plt.gca().yaxis.grid(True, linestyle='--', alpha=0.6)
    plt.gca().spines['top'].set_visible(False)
    plt.gca().spines['right'].set_visible(False)
    plt.tight_layout()
    plt.show()
```

```python
# Model names for labeling
model_names = ["Decision Tree", "Random Forest", "Naive Bayes", "Support Vector Machine"]

# Plot precision by class
plot_precision_by_class(model_names, precision_class_0, precision_class_1)
```

```
# Extract recall for each class from the class reports
recall_class_0 = [
    report['0']['recall'] for report in class_reports
]
recall_class_1 = [
    report['1']['recall'] for report in class_reports
]

# Plotting function for recall by class
def plot_recall_by_class(model_names, recall_class_0, recall_class_1):
    bar_width = 0.35
    index = np.arange(len(model_names))

    plt.figure(figsize=(12, 7))
    bars1 = plt.bar(index, recall_class_0, bar_width, label='Class 0')
    bars2 = plt.bar(index + bar_width, recall_class_1, bar_width, label='Class 1')

    plt.xlabel('Model', fontsize=14, fontweight='bold')
    plt.ylabel('Recall', fontsize=14, fontweight='bold')
    plt.title('Recall by Class for Each Model', fontsize=18, fontweight='bold', color='#4f4f4f')
    plt.xticks(index + bar_width / 2, model_names)
    plt.ylim([0, 1.1])
    plt.legend()

    for bars in [bars1, bars2]:
        for bar in bars:
            yval = bar.get_height()
            plt.text(bar.get_x() + bar.get_width() / 2, yval + 0.01, f'{yval:.2f}',
                     ha='center', va='bottom', fontsize=12, color='black')

    plt.gca().yaxis.grid(True, linestyle='--', alpha=0.6)
    plt.gca().spines['top'].set_visible(False)
    plt.gca().spines['right'].set_visible(False)
    plt.tight_layout()
    plt.show()
```

```
# Model names for labeling
model_names = ["Decision Tree", "Random Forest", "Naive Bayes", "Support Vector Machine"]

# Plot recall by class
plot_recall_by_class(model_names, recall_class_0, recall_class_1)
```

```python
# Extract F1 scores for each class from the class reports
f1_class_0 = [
    report['0']['f1-score'] for report in class_reports
]
f1_class_1 = [
    report['1']['f1-score'] for report in class_reports
]

# Plotting function for F1 score by class
def plot_f1_by_class(model_names, f1_class_0, f1_class_1):
    bar_width = 0.35
    index = np.arange(len(model_names))

    plt.figure(figsize=(12, 7))
    bars1 = plt.bar(index, f1_class_0, bar_width, label='Class 0')
    bars2 = plt.bar(index + bar_width, f1_class_1, bar_width, label='Class 1')

    plt.xlabel('Model', fontsize=14, fontweight='bold')
    plt.ylabel('F1 Score', fontsize=14, fontweight='bold')
    plt.title('F1 Score by Class for Each Model', fontsize=18, fontweight='bold',
     color='#4f4f4f')
    plt.xticks(index + bar_width / 2, model_names)
    plt.ylim([0, 1.1])
    plt.legend()

    for bars in [bars1, bars2]:
        for bar in bars:
            yval = bar.get_height()
            plt.text(bar.get_x() + bar.get_width() / 2, yval + 0.01, f'{yval:.2f}',
                     ha='center', va='bottom', fontsize=12, color='black')

    plt.gca().yaxis.grid(True, linestyle='--', alpha=0.6)
    plt.gca().spines['top'].set_visible(False)
    plt.gca().spines['right'].set_visible(False)
    plt.tight_layout()
    plt.show()
```

```python
# Model names for labeling
model_names = ["Decision Tree", "Random Forest", "Naive Bayes", "Support Vector Machine"]

# Plot F1 score by class
plot_f1_by_class(model_names, f1_class_0, f1_class_1)
```

```
rf_model = RandomForestClassifier(n_estimators=20, criterion='entropy', random_state=0)
rf_model.fit(X_train, y_train)

# Save the trained Random Forest model
with open('random_forest_model.pkl', 'wb') as model_file:
    joblib.dump(rf_model, model_file)

# Load the trained Random Forest model
with open('random_forest_model.pkl', 'rb') as model_file:
    random_model = joblib.load(model_file)

def predict_heart_disease():
    # Get input values from the user
    try:
        age = int(entry_age.get())
        sex = int(entry_sex.get())
        cp = int(entry_cp.get())
        trestbps = int(entry_trestbps.get())
        chol = int(entry_chol.get())
        fbs = int(entry_fbs.get())
        restecg = int(entry_restecg.get())
        thalach = int(entry_thalach.get())
        exang = int(entry_exang.get())
        oldpeak = float(entry_oldpeak.get())
        slope = int(entry_slope.get())
    except ValueError:
        messagebox.showerror("Input Error", "Please enter valid numeric values for all fields.")
        return
```

```
    # Create the feature array
    features = np.array([[age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
     slope]])

    # Predict using the loaded model
    prediction = random_model.predict(features)

    # Display the result
    if prediction[0] == 1:
        result_text.set("The patient is likely to have heart disease.")
    else:
        result_text.set("The patient is normal.")
    messagebox.showinfo("Success", result_text.get())

# Create the main window
window = tk.Tk()
window.title("Heart Disease Prediction")
window.geometry("500x600")
window.configure(bg='#d3d3d3')
```

```
# Title
title_label = tk.Label(window, text="Heart Disease Prediction System", font=("Helvetica", 20),
 bg='#d3d3d3', pady=20)
title_label.pack()

# Frame for the input fields
frame = tk.Frame(window, bg='#d3d3d3')
frame.pack(pady=10)

fields = [
    ("Age", "entry_age"),
    ("Sex (1=Male, 0=Female)", "entry_sex"),
    ("Chest Pain Type (1-4)", "entry_cp"),
    ("Resting Blood Pressure", "entry_trestbps"),
    ("Cholesterol", "entry_chol"),
    ("Fasting Blood Sugar (1 if >120mg/dl, 0 otherwise)", "entry_fbs"),
    ("Resting ECG (0-2)", "entry_restecg"),
    ("Max Heart Rate", "entry_thalach"),
    ("Exercise Induced Angina (1=Yes, 0=No)", "entry_exang"),
    ("Oldpeak", "entry_oldpeak"),
    ("ST Slope (1-3)", "entry_slope")
]

entries = {}

for i, (label_text, var_name) in enumerate(fields):
    tk.Label(frame, text=label_text, bg='#d3d3d3').grid(row=i, column=0, padx=10, pady=5,
     sticky="e")
    entries[var_name] = tk.Entry(frame)
    entries[var_name].grid(row=i, column=1, padx=10, pady=5, sticky="w")
globals().update(entries)
```

```
# Button to predict
predict_button = tk.Button(frame, text="Predict", command=predict_heart_disease, bg='#ff69b4',
 fg='white', font=("Helvetica", 12))
predict_button.grid(row=len(fields), column=0, columnspan=2, pady=20)

# Label to display the result
result_text = tk.StringVar()
result_label = tk.Label(window, textvariable=result_text, fg="blue", font=("Helvetica", 12),
 bg='#d3d3d3')
result_label.pack(pady=10)

# Start the GUI loop
window.mainloop()
```