Department of Computer Science and Network Engineering

6500CSMM Project

Final Report Submitted by

Khaing Khant Kyaw

1087395

Bachelor of Science with Honors in Computer Networks

"Data Analysis on Heart Disease Prediction with Machine Learning Algorithms"

Supervised by

Dr. Yuzana Win

Submitted on

May 2024

ABSTRACT

Heart disease is a major worldwide health concern, highlighting the crucial need for precise prediction models that enable early detection and intervention. This research focuses on the creation and optimization of machine learning models for cardiac disease prediction, utilizing large datasets and cutting-edge modeling approaches.

The research takes a systematic approach, beginning with painstaking data preparation, which includes combining many datasets linked to heart disease and rigorous data cleansing to assure data integrity. The meticulously constructed dataset is used to train a variety of classification models, including the Random Forest Classifier, Logistic Regression, KNeighbors Classifier, Decision Tree Classifier, and neural network.

We refine the models' performance through intensive experimentation with various train-test split ratios and hyperparameter adjustment via grid search. Model evaluation using accuracy scores allows for a full comparison across different split ratios and hyperparameters. Finally, the best-performing model, combined with the ideal train-test split ratio, is determined for heart disease prediction.

Furthermore, we assess the models' generalization abilities on an independent dataset composed of previously encountered heart disease data. This evaluation gives essential information about the models' resilience and real-world applicability. Finally, this effort increases cardiac disease prediction procedures while also shedding light on useful modeling approaches. In addition, I utilize Flask to deploy the best-performing machine learning model, the Random Forest Classifier, with a 90/10 train-test split ratio on a localhost website, making it more accessible and practical for potential users.

ACKNOWLEDGMENTS

The author is deeply grateful to Liverpool John Moores University for the opportunity to study a Bachelor of Science in Computer Networking.

The author would like to recognize Dr. Sandar Myo Myint, Head of the Academic Department of Auston University, for graciously granting approval and providing a valuable chance to pursue this profound thesis topic. This continuous support has had a significant impact on the project's course, for which the author is deeply thankful.

Appreciate is expressed to Dr. Nyein Aye Maung Maung, Dean of the Department of Computer Science and Network Engineering, and Sayar Thet Zaw Aye, Professor in the same department, for their significant contributions and constructive recommendations throughout the project's journey. Their efforts considerably increased the project's quality and depth.

The author wants to express my heartfelt gratitude to Dr. Yuzana Win, my project supervisor, for her tremendous guidance, constant support, and constructive comments throughout the project. Her knowledge and passion were important in influencing the direction and outcomes of this project.

Finally, the author wants to extend my heartfelt gratitude to my family. Their steadfast support and understanding have provided a steady source of motivation throughout this journey. Their encouragement has inspired me to pursue my passion and finish this project with determination and excitement.

TABLES OF CONTENTS

LIST OF TABLES

TABLE OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Background and Motivation

Heart disease continues to be a major public health concern in the United States, killing over 600,000 people each year from illnesses such as coronary artery disease, cardiovascular disease, and angina. Furthermore, the economic burden of heart disease treatment is nearly $200 billion per year, with the figure expected to rise to $818 billion by 2030, according to the American Heart Association. Furthermore, it is expected that 116 million Americans would suffer from some sort of cardiovascular disease by then (Gopaul, 2019).

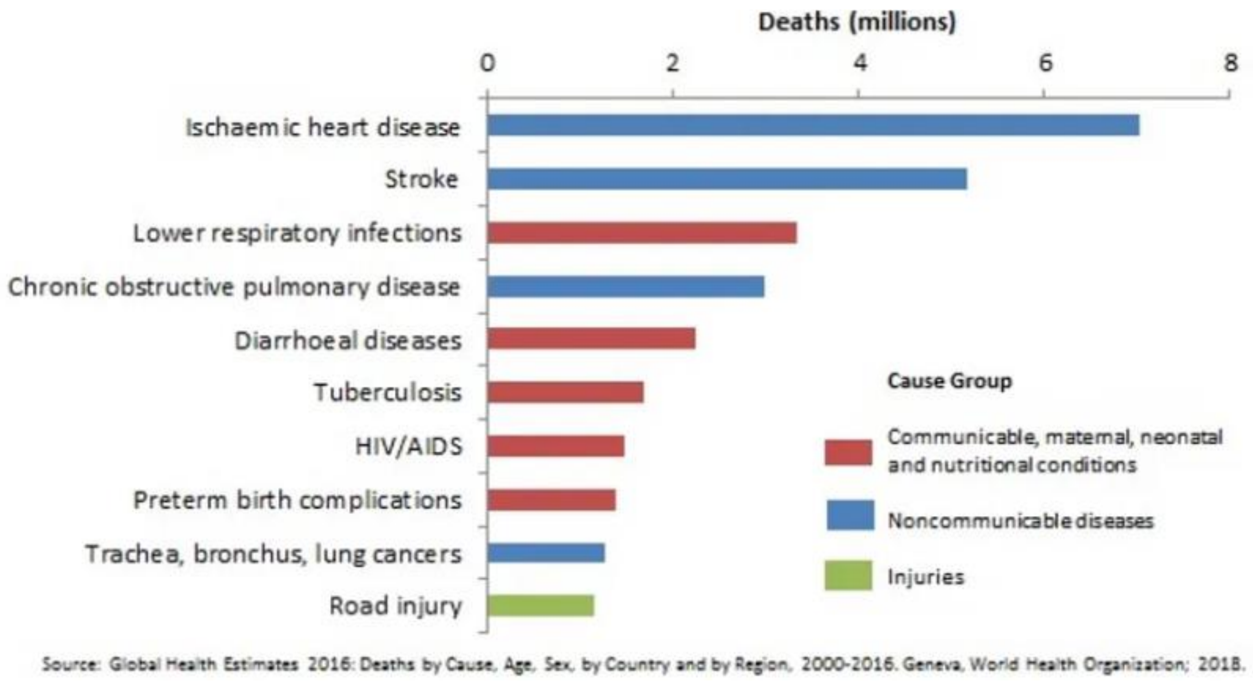


Figure 1 (Leading Cause of Death)

Since the year 2000, heart disease has been the top cause of death worldwide. The Figure 1 emphasizes the continuing prevalence and impact of cardiovascular diseases on mortality rates worldwide (Husten, 2011).

Despite breakthroughs in medical knowledge and technology, accurate diagnosis and early identification of heart disease remain significant challenges. Traditional risk assessment methods, which rely mainly on clinical criteria and medical history, may underestimate the complex and diverse nature of cardiovascular illnesses.

The motivation for this thesis stems from the crucial need to improve heart disease prediction methods in order to address this persistent public health concern. Early diagnosis of patients at risk of developing heart disease is crucial for commencing appropriate treatments, which can result in better patient outcomes and lower medical costs. We hope to change heart disease management and improve patient outcomes by leveraging the power of machine learning to construct more accurate and reliable prediction models that can assist healthcare practitioners in identifying at-risk individuals and successfully implementing preventive interventions.

Furthermore, the advantages of good heart disease prognosis extend beyond individual patient care. Predictive models, which enable early intervention and risk stratification, can aid in population-level health management strategies such as resource allocation, public health campaigns, and policy formulation. As a result, developing effective prediction models for heart disease has the potential to enhance individual patient outcomes while simultaneously advancing public health programs and lowering the overall burden of cardiovascular disease on society (Gopaul, 2019).

Crucially, all of these risk variables can be altered by informed decision-making and lifestyle adjustments. We can encourage healthier lifestyle choices and reduce unnecessary heart disease deaths by providing individuals with information about their risk of developing the condition through predictive analytics obtained from machine learning algorithms.

In conclusion, using machine learning to forecast people's risk of heart disease has potential for encouraging healthier lifestyle choices and lowering death rates related with cardiovascular illnesses. We hope that this thesis will help to develop heart disease prediction algorithms, ultimately enhancing both individual patient outcomes and population-level health management strategies.

1.2 Thesis Contribution

This thesis makes major contributions to the progress of heart disease prediction techniques. Integration of Different Data Sources: By combining several datasets that include demographic, clinical, and lifestyle characteristics, a comprehensive dataset is built that covers a wide variety of potential heart disease predictors. This integration enables a more comprehensive examination and increases the resilience of the predictive models. Advanced Machine Learning Techniques: To build predictive models for heart disease, number of machine learning techniques are investigated, such as the Random Forest Classifier, Logistic Regression, K nearest Neighbors Classifier, Decision Tree Classifier, and Neural Network and identify the best effective algorithms and approaches for heart disease prediction after extensive trial and review.

Optimization and Generalization

We optimize the prediction models' performance and test their generalizability using strategies such as hyperparameter tuning with grid search and evaluation on unknown datasets. This assures that the produced models are not only accurate but also capable of making solid predictions on previously unseen data, hence increasing their real-world use.

Contribution in Healthcare and Research

The findings of this thesis have significance for clinical practice as well as cardiovascular research. By giving insights into effective predictive modeling methodologies, we hope to help medical professionals improve risk assessment and patient management tactics. Furthermore, the methodology and techniques proposed in this thesis add to the greater knowledge of predictive analytics in healthcare, laying the framework for future study in this field.

In conclusion, this thesis is a significant step forward in the quest to enhance heart disease prediction algorithms, with implications for both individual patient care and population-level health management measures. We hope to develop prediction models that can contribute to better patient outcomes and a healthier society by integrating varied data sources, exploring sophisticated machine learning techniques, and conducting rigorous optimization and evaluation.

1.3 Problem Statement

Conventional risk assessment approaches are mostly based on clinical characteristics and medical history, which frequently overlooks the multidimensional nature of cardiovascular illness and limits the efficiency of predictive models.

The primary focus of this thesis is the creation of precise and reliable predictive models for heart disease utilizing modern machine learning techniques. This undertaking includes several important challenges:

Data Complexity and Integration

A variety of factors contribute to heart disease, including demographic traits, medical history, lifestyle choices, and diagnostic test results. Integrating several datasets and capturing the complex interactions between predictors are difficult tasks.

Model Selection and Optimization

With so many machine learning algorithms available, choosing the best models for heart disease prediction and enhancing their performance is a daunting undertaking. The efficacy of different algorithms varies with dataset properties, demanding extensive experimentation and evaluation.

Generalization and Real-world Applicability

Developing predictive models that not only perform well on training data but also generalize effectively to previously unseen data is critical for real-world application. Ensuring the models' robustness and repeatability across varied patient groups and healthcare settings is critical to their practical use.

Interpretability and Clinical Relevance

While predicted accuracy is crucial, model interpretability and clinical relevance are also critical concerns. Clinicians need to understand the elements that influence predictions in order to make informed judgments and successfully customize therapies to particular patient needs.

Addressing these difficulties necessitates a holistic approach that combines powerful machine learning techniques with cardiovascular health subject knowledge. This thesis seeks to improve risk assessment, early intervention, and patient outcomes in cardiovascular healthcare by constructing reliable predictive models for heart disease.

1.4 Aims and Objectives

Aim

The ultimate goal of this research project is to improve our understanding, prediction, and prevention of heart disease by integrating varied datasets and modern machine learning approaches. The study's goal is to construct highly precise and reliable predictive models capable of early identification and risk stratification using a multimodal approach that includes demographic, clinical, lifestyle, and diagnostic data. This study intends to improve patient outcomes by thorough data analysis, model optimization, and validation, as well as to increase the accuracy of heart disease prediction. By bridging the gap between data science and cardiovascular health, the study hopes to make a substantial contribution to the advancement of predictive analytics in healthcare, paving the way for more effective heart disease prevention and management at the population level.

Objectives

Comprehensive Data Integration and Analysis: Conduct a thorough evaluation and integration of several datasets on heart disease, including demographic information, medical history, lifestyle factors, and diagnostic test findings. Use modern data analysis tools to find important relationships,

patterns, and risk factors for heart disease. Evaluate and optimize machine learning models: To determine the best models for heart disease prediction, compare a variety of machine learning methods such as the Random Forest Classifier, Logistic Regression, K nearestNeighbors Classifier, Decision Tree Classifier, and Neural Network. Optimize model parameters using approaches like hyperparameter tuning to improve predicted accuracy and generalization.

Create interpretable and explainable machine learning models to provide insights into the elements influencing forecasts and aid in clinical decision-making. To increase healthcare professionals' faith in and use of predictive models, emphasize transparency and interpretability.

Validation and Generalization Assessment

Use rigorous validation approaches, such as cross-validation and train-test splits, to evaluate the produced models' performance on previously unseen data. To verify reliability and effectiveness in real-world applications, assess the models' capacity to generalize across varied patient groups and healthcare environments.

Implementation of personalized Intervention Strategies

Use predictive models to create individualized intervention plans for people who are at risk of developing heart disease. To enhance patient outcomes, tailor intervention programs to individual risk profiles and focus preventative interventions such lifestyle changes, medication management, and regular monitoring.

Knowledge Translation and Distribution

Translate research findings into meaningful insights and share knowledge with healthcare professionals, policymakers, and the general public. Publish research results in peer-reviewed publications, present findings at conferences, and participate in knowledge-sharing events to stimulate collaboration and drive innovation in cardiovascular health.

By achieving these goals, this study hopes to enhance the area of heart disease prediction and prevention, resulting in better patient outcomes and population-wide health management techniques.

1.5 Scope of the Project

This research intends to predict cardiac disease using machine learning techniques, with a focus on studying a large dataset of health-related variables. The project entails data collecting, preparation, and analysis in order to find important factors associated with heart disease. Various machine learning models will be created, trained, and assessed, with the goal of determining the most accurate one for prediction. The chosen model will be integrated into a user-friendly application, allowing people to enter their health information and receive estimates about their risk of heart disease. Risks such as data quality difficulties and technical challenges will be mitigated by using alternate data sources, conducting extensive testing, and gathering stakeholder feedback. Ultimately the goal of this project is to assess a dataset using different machine learning models for heart disease prediction and select the best accurate one for deployment. After data preparation, multiple algorithms will be trained and assessed, with the best-performing model included into a user-friendly interface for real-world use.

CHAPTER 2: LITERATURE REVIEW

This literature review analyzes studies on cardiac disease prediction using computational and mathematical methodologies.

In a study conducted by Gufran Ahmad Ansari evaluating machine learning algorithms such as Logistic Regression (LR), (RENJIE GU, CHAOYUE NIU, 2021)K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Naive Bayes (NB), Decision Trees (DT), and Random Forest (RF), KNN had the highest accuracy at 99%, followed closely by Decision

Tree and Random Forest, both of which had 97% accuracy (Ansari, 2023).Statistical analyses, such as confusion matrices, assessed model performance in terms of accuracy, precision, recall, F1 measure, and Matthews Correlation Coefficient (MCC). These measurements shed light on the efficacy of each algorithm, with KNN outperforming Decision Tree and Random Forest across several criteria.

A comparative comparison of the proposed framework with existing literature found positive results, notably in terms of heart disease prediction accuracy (citation). Data imputation, outlier identification, and data standardization techniques all helped the system perform well. Furthermore, the usage of K-fold cross-validation improved results dependability. (Ansari, 2023)

In "Machine Learning Models for Heart Disease Diagnosis," by Ahmad Ayid Ahmad use numerous algorithms, such as ANN, DT, AdaBoost, and SVM, were tested on heart disease diagnostic data. SVM achieved the maximum accuracy of 98.09% without feature selection, while SVM-JF achieved 98.47% using the Jellyfish algorithm. These findings were illustrated via graphs. The combination strategy of SVM and feature selection produced greater Area Under Curve values, suggesting better accuracy (Ahmad, 2023).

Comparative study with earlier studies revealed that the proposed strategy was effective, with a classification accuracy of 98.47%. The study also explored the possibilities of machine learning in cardiac disease detection, with a focus on data quality, algorithm understandability, privacy, and the physician-patient interaction. These findings emphasize machine learning's significant role in improving heart disease diagnosis while mitigating potential implementation obstacles.

In a study on heart disease prediction analysis by Vikram Rajkuma, several machine learning algorithms were used, including Logistic Regression, Decision Tree, Random Forest, and Support Vector Machines (SVM). Among these, the Decision Tree algorithm performed best, achieving an accuracy of 92%. This outperformed Logistic Regression (80%), Random Forest (80%), and Support Vector Machines (82%). Using the Decision Tree Classifier, precise predictions of heart disease were derived based on patient data, demonstrating its usefulness in real-time diagnosis. Surprisingly, the program properly classified patients' situations when evaluated against possible heart disease cases and normal data. This highlights the critical relevance of real-time predictors in healthcare, which provide accurate insights for decision-making (RAJKUMAR, 2023).

Machine Learning algorithm conducted by Chiradeep Gupta name “Cardiac Disease Prediction using Supervised Machine Learning Techniques.” use Supervised learning paradigms, such as Logistic Regression, Naïve Bayes, Decision Tree, SVM Model, KNN, and Random Forest, are similar to learning under a qualified instructor [3]. For the proposed research, the dataset is carefully partitioned into training and testing subsets in a 7:3 ratio, with the training subset acting as an instructor, directing predictions on the testing subset [4]. These strategies have showed efficacy in the creation of medical aid software for early disease identification, potentially decreasing the risk of death by recognizing heart-related disorders in their early stages. In the context of this study, comparing multiple machine learning models reveals subtle changes in performance indicators. Logistic Regression achieves an accuracy of roughly 92.30%, demonstrating its efficacy in classifying heart disease data. Similarly, the SVM Model achieves a high accuracy level of approximately 91.20%, indicating its promise as a robust classification tool. Naïve Bayes' accuracy rate of roughly 87.91% demonstrates its usefulness in predictive modeling for heart disease diagnosis. Meanwhile, Decision Tree, K-Nearest Neighbour (with k values of 7 and 14), and Random Forest all achieve accuracy levels ranging from 85.71% to 90.11%, demonstrating their relative performance in this area (Gupta, 2022).

These accuracy measures provide useful information about each model's usefulness in identifying heart disease data, laying the groundwork for educated model selection and refinement. As so, they provide a substantial contribution to the continuing discussion about the use of machine learning in improving heart disease detection and highlight the promise of these techniques in tackling pressing healthcare concerns.

Machine learning techniques show promise in addressing a variety of cardiac disease-related concerns. Marimuthu et al. undertook a review of heart disease prediction systems, including Decision Trees (DT), Naive Bayes (NB), K-Nearest Neighbors (KNN), and Support Vector Machines. Similarly, Battula et al. published a detailed analysis of several machine learning algorithms used for heart disease prediction since 2012. Several studies have undertaken comparative examinations of cardiac diseases utilizing machine learning algorithms, emphasizing their usefulness.

M. Marimuthu proposed a logistic regression classifier-based decision support system called "Analysis of Heart Disease Prediction Using Various Machine Learning Techniques" that achieved a classification accuracy of 77%. Machine learning is useful in solving a variety of problems, including predicting dependent variables using independent factors. Advanced analytics are required in the health sector due to the vast amount of data available. Despite the fact that heart disease is a primary cause of death in industrialized nations, dangers are frequently overlooked or identified too late. However, machine learning approaches can help with early risk prediction.

Support Vector Machines (SVM), Decision Trees (DT), regression, and Naive Bayes (NB) classifiers have all demonstrated promise in predicting heart disease, with SVM having the greatest accuracy of 92.1%, followed by neural networks (91%), and decision trees (89.6%). Gender and smoking are thought to be risk factors for cardiovascular disease. Various machine learning algorithms, including DT, NB, and associative classification, are excellent at predicting cardiac disease, especially when dealing with unstructured data.

Artificial Neural Networks (ANN) have also been used for disease prediction, with supervised networks and backpropagation algorithms producing good results. Techniques such as the Intelligent Heart Disease Prediction System (IHDPS), which combines DT, NB, and neural networks (NN), have produced encouraging results, with the NB model achieving the greatest prediction accuracy of 86.1%. Research targeted at increasing algorithm categorization by machine learning approaches improves the effectiveness and accuracy of heart disease prediction.

Studies comparing Logistic Regression (LR), K-Nearest Neighbors (KNN), SVM, NB, DT, and Random Forest (RF) approaches, as well as feature selection, reveal better results, notably in the healthcare sector. These findings help us understand how effective machine learning classifiers may be in predicting heart disease.

These research activities contribute to the burgeoning field of cardiac disease prediction by applying a variety of approaches and algorithms to address the complications inherent in healthcare prognostication. A comprehensive literature analysis reveals the varied nature of heart disease prediction, emphasizing the importance of machine learning technologies in identifying and managing cardiovascular risks. The combination of findings from these studies provides essential insights into the application of various algorithms, highlighting their individual strengths and areas for improvement. Such findings are consistent with the goals of this initiative, which seeks to improve the accuracy of heart disease prediction algorithms.

CHAPTER 3: THEORETICAL BACKGROUND

3.1 Machine Learning

Machine learning is a field of study that involves inferences and predictions from data based on mathematics, statistics, and computer science (MACABIAU,CLARA,MANA_SHAHRIARI2, 2024_February_2) . The goal of machine learning is to generalize across different sample populations, data structures, and data errors to provide meaningful insights, with vast clinical applications (Varoquaux & Cheplygina, 2022). It involves the development of rules (algorithms) that allow algorithms to learn and make inferences or predictions from the data. Medical imaging is the use of sophisticated computer systems and systems to assess and detect latent diseases (Gabriel Iagar & Laurençot, 2023). Mammography, computed tomography (CT), magnetic resonance imaging (MRI), and positron emission tomography (PET) are medical imaging approaches. Medical imaging can diagnose development, target tumor, and treatment. However, an individual's judgement, memory, judgment, and opinion vary the diagnosis of health imaging. Clinical data analysis techniques have been applied more by healthcare professionals for decision-making devices and advanced identification. Character and automatization of earlier unknown patterns, and integration into clinical decision support systems are some of the main advantages and applications of deep learning methods in healthcare.

3.1.1 Types of Machine Learning

Machine learning addresses various business problems, such as Regression, Classification, Forecasting, Clustering, and Associations. The types of machine learning are primarily divided into four categories: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning (RENJIE GU, CHAOYUE NIU, 2021).

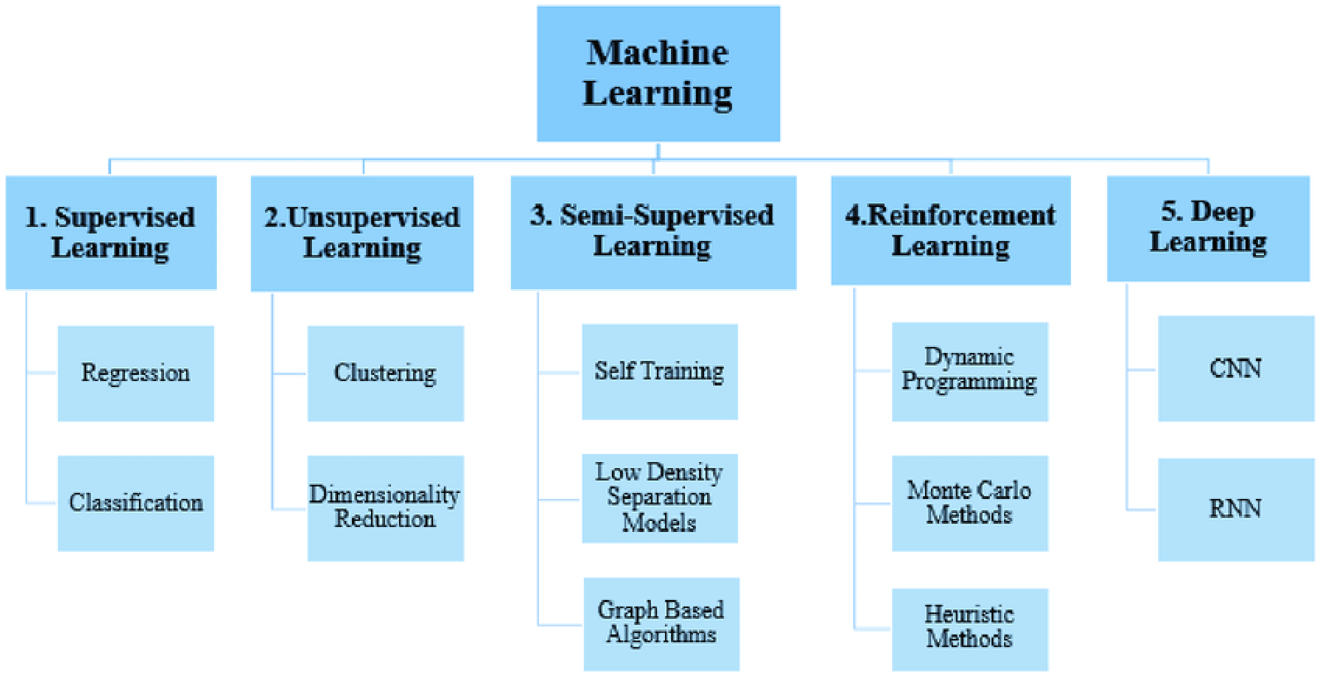


Figure 2 (Types of Machine Learning)

3.1.1.1 Supervised Learning

In classical supervised learning, labeled data (inputs and their corresponding desired outputs) are used to train a machine learning model. The richness of any machine learning problem lies in the diverse types of inputs (features) and corresponding outputs. Problems with categorical outputs are called classification tasks, while those with numerical outputs are termed regression tasks. Classification can be binary (two categories) or multi-class (three or more categories).

In this context, problems can be seen as multi-class when categories are based on distances from centroids that neurons represent. Adding a category infers a larger model from a smaller one, such as a single layer perceptron, avoiding the need to set the number of categories beforehand. In both classification and regression, the outputs are fed back to the model, making error significance dependent on their magnitudes and varying impacts.

Supervised learning predicts a target value using labeled data and is widely used in applications such as medical diagnosis, speech recognition, and machine vision (Gu et al., 2019). It involves pairs (x, y) where x is the input and y are the output, aiming to generalize a function that maps inputs to outputs or probability distributions over classes. Supervised learning includes classification, predicting results from a list of categories (e.g., loan eligibility), and regression, predicting results within a continuous range. (RENJIE GU, CHAOYUE NIU, 2021)

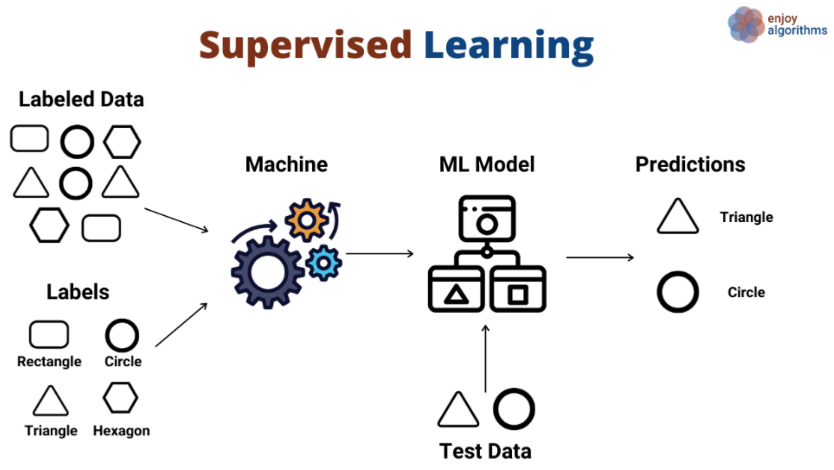


Figure 3 (Supervised Learning)

3.1.1.2 Unsupervised Learning

Association methods, such as market basket analysis, can identify sequential patterns in frequent item sets or transaction databases, and are applicable in healthcare. Doctors can use historical patient data to detect connected disease patterns. Discovering circular attribute-based patterns in this data can aid in diagnosing related diseases.

Clustering algorithms, depicted in Fig. 3 by Cho et al., have been used for predicting severe patient outcomes (Grace Famera et al., 2023). Principal Component Analysis (PCA) projects data along significant directions and was first used in studies by Balarabe Junaid et al. (2022). During the H1N1 SARS outbreak, chromosomal microarray analysis monitored and prevented virus spread from different locations.

Unsupervised learning analyzes and clusters unlabeled data to reveal hidden structures or categories without a predetermined goal. This method identifies patterns in data and is used in customer segmentation, trend observation, and image simulation. In healthcare, clustering can detect early-stage injuries through unsupervised learning (N. M. Sajedul Alam et al., 2022).

Clustering algorithms like k-means and hierarchical clustering, which uses decision trees to classify clusters, are prevalent. The user can determine the number of clusters based on the tree's design.

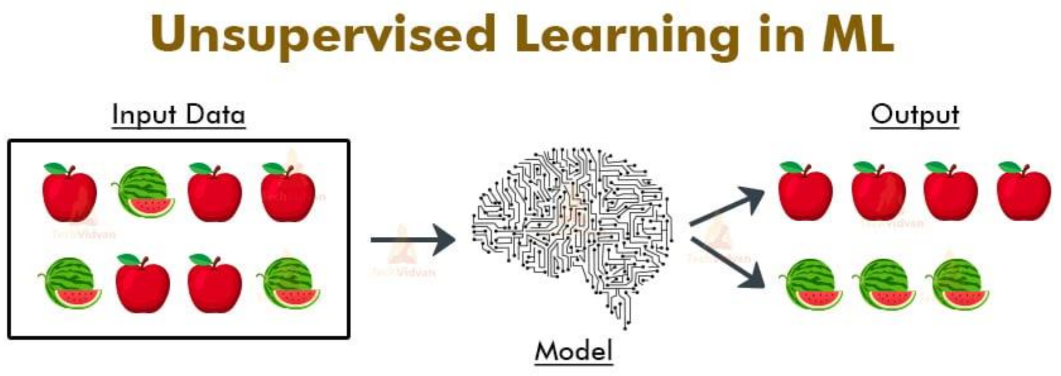


Figure 4 (Unsupervised Learning)

3.1.1.3 Semi-Supervised Learning

Self-ensembling leverages two classifiers in a co-training system, sharing one classifier between natural inputs and their augmentations to simplify semi-supervised learning. This approach has recently been applied to semi-supervised deep learning, enhancing the generalization of pre-trained models. We propose using self-ensembling for semi-supervised cellular segmentation of large-scale SEM images to accurately associate 3D shapes with heritable molecules.

Self-apprenticeship, a method to handle unlabeled samples, reduces uncertainty by using an ensemble of classifiers to generate reliable labels. Co-training involves two classifiers that share features and label mutually predictable samples. However, using a single ResNet for co-training in SEM image segmentation to enhance model generalization has not been explored.

Supervised learning, which relies on labeled data, is effective but labor-intensive (Rani et al., 2023). Semi-supervised learning uses both labeled and unlabeled data, aiming to minimize estimated targets for unlabeled data (Varma Nadimpalli et al., 2021). This is achieved through methods like self-training, co-training, and self-ensembling (Aswath et al., 2022).

3.1.1.4 Reinforcement Learning

Reinforcement Learning (RL) involves an agent learning to make decisions by interacting with an environment to maximize a reward. The agent follows a policy that maps states to probabilities of taking different actions until it reaches a terminal state, where no further actions are possible, and a reward is received. The RL environment must fulfill the Markov Decision Process (MDP) properties, where the next state $s(t+1)$s(t+1) depends only on the current state $s(t)$s(t) and action $a(t)$a(t), along with the expected immediate reward $r(t)$r(t) (time-discounted reward) (Charpentier, 2020).

The RL framework consists of an agent (the decision-maker and learner) and an environment. The environment facilitates interactions between the agent and external factors. At each time step tt, the agent is in state $s(t)$s(t) and selects an action $a(t)$a(t) from the set of possible actions AA. The environment then provides a reward $r(t)$r(t) and transitions to the next state $s(t+1)$ s(t+1). Starting from the initial state s0s0, the policy process interacts with the MDP across episodes, moving from the initial state to the terminal state. The agent learns to choose actions based on past experiences and policies, optimizing future decisions (Charpentier, 2020).

Reinforcement Learning is a subset of machine learning where an agent learns optimal behavior through direct interaction with the environment, aiming to achieve goals or maximize rewards. This learning process is unsupervised, relying on feedback from actions and rewards to improve decision-making over time.

3.1.1.5 Deep Learning

Deep Learning is a system of interconnected layers forming a network, analogous to the human brain. It includes three primary layers: the input layer (for providing information), the hidden layer (for processing information), and the output layer (for displaying results). These layers are categorized into three types: convolutional, recurrent, and feedforward.

Convolutional Neural Networks (CNNs): Primarily used for image and sound analysis, CNNs operate through convolutions and are the most commonly applied neural networks for image recognition.

Recurrent Neural Networks (RNNs): These are designed for sequential data processing, where input passes through nodes and can influence subsequent outputs.

Deep Learning models possess many layers, allowing them to extract advanced features from data. As these networks grow larger and more complex, they achieve high accuracy in tasks such as image classification and language translation (Soledad Le Clainche, Elisabeth Cross, 2022).

Unlike traditional machine learning models, which require manual feature extraction, deep learning models can automatically derive necessary features from raw data, making them highly effective for complex data tasks (Jalal Rezaeenour,Mahnaz Ahmadi, 2022).

3.1.2 Relevance to Heart Disease Prediction Analysis

In the context of the project on heart disease detection, machine learning serves as the backbone for developing models capable of distinguishing between healthy and diseased states. By understanding these foundational concepts, the subsequent exploration of specific algorithms—Logistic Regression, Random Forest Classifier, Decision Tree, K-Nearest Neighbor Classifier, and Neural Network—will be more comprehensive

3.2 Machine Learning Algorithms

3.2.1 Overview of Machine Learning Algorithms used in this Project

Logistic Regression

Logistic Regression, a generalized linear model, calculates the probability of belonging to one of two classes using logistic transformation. The penalized likelihood, or Lasso, aids in feature selection shrinkage.

Decision Tree

Decision Tree algorithms make predictions by learning simple decision rules from data.

Random Forest:

Random Forest, an ensemble technique, merges multiple decision trees to produce accurate and stable predictions. However, if not managed properly, it may lead to overfitting.

K-Nearest Neighbor (KNN)

K-Nearest Neighbor is a lazy algorithm that labels new data points based on the similarity of their K nearest neighbors.

Neural Networks

Neural Networks are a class of machine learning algorithms inspired by the structure and functioning of the human brain. They consist of interconnected layers of nodes (neurons) that process information. Neural networks can learn complex patterns and relationships in data, making them versatile for various tasks such as classification, regression, and pattern recognition.

Application Areas

Traditional statistical models and modern machine learning algorithms find applications in various fields, including social networks, graphical models, images, spatial data, and diseases. Examples include multiple linear regression, polynomial regression, neural networks, classification/regression trees, and support vector machines.

Ongoing research aims to determine the accuracy of traditional statistical models versus machine learning algorithms, particularly with high-dimensional data.

Detail Logistic Regression

Logistic Regression (LR) is a statistical machine learning method utilized for binary (two classes) and multiclass (more than two classes) classification tasks (Meesad, 2021). It employs the sigmoid function for classification and performs admirably with small, linearly separable datasets, exhibiting a reduced likelihood of overfitting. LR's cost function can drive coefficients to zero, facilitating the selection of significant features while disregarding irrelevant ones. However, LR may encounter overfitting challenges in complex and large datasets (Feyzioglu & Selim Taspinar, 2023). It is advisable to avoid using LR when the number of data points is fewer than the number of features. In cases where the decision boundary is nonlinear, kernelized Support Vector Machine (SVM) is recommended. Kernelized SVM has the capability to capture nonlinear relationships in data and generate a linear decision boundary.

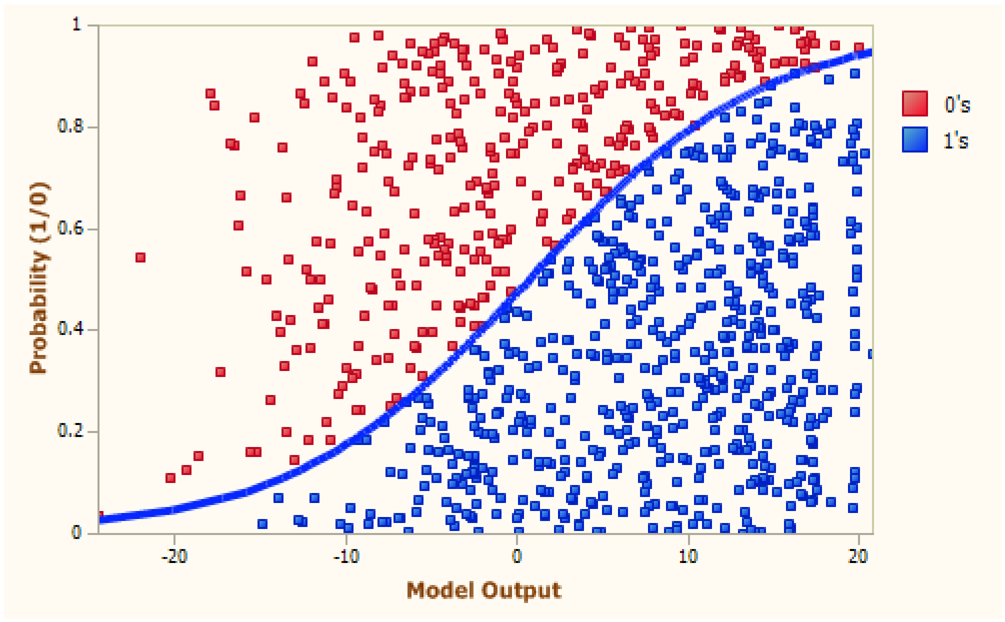


Figure 5 (Logistic Regression)

Mathematical Formulation

The logistic regression model can be represented mathematically as follows:

$P(Y=1|X)=11+e-(\beta 0+\beta 1X1+\beta 2X2+...+\beta nXn)$ P(Y=1|X) =1+e−(β0+β1X1+β2X2+...+βnXn)1

Where:

$P(Y=1|X)$ P(Y=1|X) is the probability of the positive class given input features XX.

$\beta 0$, $\beta 1$, βnβ0, β1, βn are the model coefficients.

$X1$, $X2$, XnX1, X2, Xn are the input features.

Graphical Model of Logistic Regression

The graphical model of logistic regression represents the relationships between input features and the output class probability. However, logistic regression is not typically represented using a graphical model like Latent Dirichlet Allocation.

Interface in Logistic Regression

Training: Logistic regression models are trained using optimization algorithms such as gradient descent to minimize the logistic loss function.

Inference: During inference, the trained model is used to predict the probability of the positive class given new input features.

Logistic regression finds applications in various domains such as healthcare, finance, and marketing, where binary classification tasks are prevalent. Its simplicity, interpretability, and efficiency make it a popular choice for predictive modeling tasks.

3.2.2 Detail Decision Tree Algorithm

Decision Trees in Classification

Decision Trees, although an older technique, have historically been used for classification. However, modern algorithms like Random Forest have gained prominence due to their enhanced performance. Despite this, Decision Trees remain relevant, especially for identifying a small set of interesting features and removing contamination. In our study, we utilized Decision Trees alongside the CLASSIFICATION AND REGRESSION TREES software to classify molecular clinkers in multivariate curve resolution. The constructed tree effectively distinguished between frustules and spikes, marking a novel application in the field. Additionally, our work pioneers the use of RNA-sequencing data for diagnosing SARS-CoV-2 infection, showcasing the versatility of Decision Trees in diverse domains.

Understanding Decision Tree Algorithm

The Decision Tree algorithm is a type of Supervised Learning technique suitable for high-dimensional feature spaces. It operates by recursively splitting data based on feature values, aiming to learn the underlying structure of the dataset. For instance, in a housing price prediction model, the algorithm discerns patterns where the price increases with the number of rooms and follows a specific trend based on location, among other features (Mélina Côté,Mazid Abiodoun Osseni, , 2022).

Decision Tree Basics

A decision tree possesses a structure reminiscent of a flowchart, where internal nodes represent features, branches represent rules, and leaf nodes represent the algorithm's outcome. It functions as a flexible supervised machine learning technique suitable for both classification and regression tasks. Importantly, decision trees are integral to ensemble methods such as Random Forest, enhancing the overall resilience and effectiveness of the algorithm.

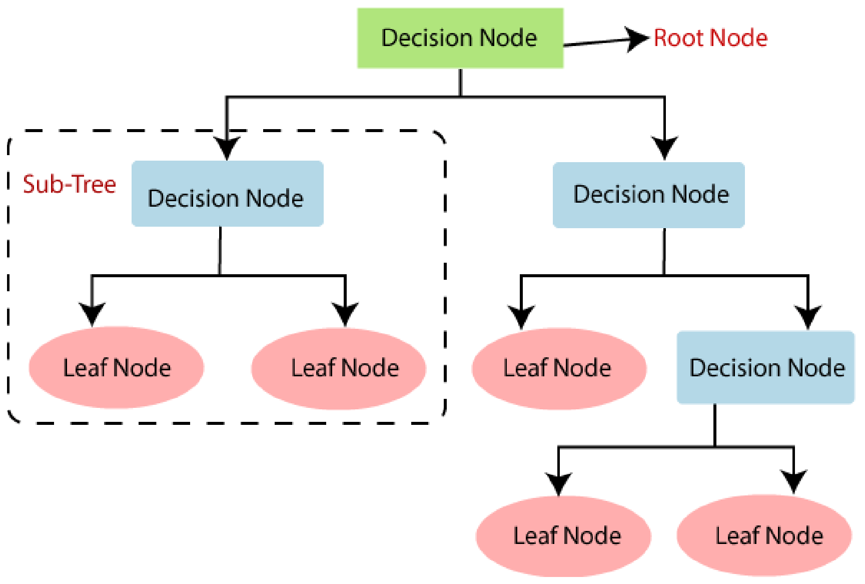


Figure 6 (Decision Tree)

Decision Tree Terminologies

Having Knowledge, the terminology associated with decision trees is crucial:

Table 1 (Decision Tree Terminologies)

Term	Description
Root Node	The top node representing the entire dataset, serving as the initial point for decision-making.
Decision/Internal Node	Nodes representing choices related to input features, linking to leaf nodes or other internal nodes.
Leaf/Terminal Node	Nodes without child nodes, indicating a class label or numerical value.
Splitting	The process of dividing a node into two or more sub-nodes based on a split criterion and a selected feature.
Branch/Sub-Tree	A subsection of the decision tree starting at an internal node and ending at leaf nodes.
Parent Node	The node dividing into one or more child nodes.
Child Node	Nodes emerging when a parent node is split.
Impurity	A measure of the target variable's homogeneity in a data subset, evaluating the degree of randomness or uncertainty. Common impurity measurements include Gini index and entropy for classification tasks.

Variance	Measures the variability between predicted and target variables in different dataset samples, utilized for regression problems in decision trees.
Information Gain	Measures the reduction in impurity achieved by splitting a dataset on a specific feature. The objective is to select the attribute that maximizes information gain, leading to more homogeneous subsets concerning class labels.
Pruning	The process of removing branches from the tree that do not provide additional information, preventing overfitting.

Assessing Split Quality in Decision Trees: Entropy, Gini Impurity, and Information Gain

Decision tree construction involves selecting attributes based on Attribute Selection Measures (ASM). These criteria evaluate the usefulness of different attributes for dataset splitting, aiming to identify attributes creating the most homogeneous subsets and maximizing information gain.

Entropy, Gini Impurity, and Information Gain are metrics utilized in decision tree algorithms to assess the quality of splits and determine the most valuable features for dataset partitioning.

Entropy serves as a measure of the randomness or uncertainty within a dataset, particularly in classification tasks. It quantifies this randomness by analyzing the distribution of class labels within a subset of the data. Mathematically, the entropy for a subset with K classes can be expressed as:

$$H_i = -\sum_{k \in K} p(i, k) \log_2 p(i, k)$$

Here, $p(i, k)$ p (i, k) represents the proportion of data points belonging to class k within the dataset sample. The objective of utilizing entropy is to minimize it after splitting the dataset, thereby creating more homogeneous subsets.

Gini Impurity, on the other hand, evaluates the accuracy of splits among classified groups. It ranges from 0 to 1, where a value of 0 indicates that all observations belong to one class, while a value of 1 signifies a random distribution. The Gini Impurity score is calculated using the formula:

$GiniImpurity=1-\sum ipi2$

Here, pipi represents the proportion of elements in the set belonging to the i-th category.

Information Gain measures the reduction in entropy achieved by splitting a dataset based on a specific property or feature. This metric plays a crucial role in determining which feature is most valuable for partitioning the dataset into more homogeneous subsets. The formula to calculate Information Gain for an attribute A is:

$InformationGain(H,A)=H-\sum(|HV||H|)HV$

Here, H represents the entropy of the dataset sample, $|HV|$|HV| denotes the number of instances in the subset with value v for attribute A, and HVHV represents the entropy of the subset.

These metrics collectively aid in the decision-making process of constructing decision trees, ensuring that splits are made effectively to enhance the homogeneity of resulting subsets.

Strengths and Weaknesses

Strengths and Weaknesses of Decision Trees:

Table 2 (Strength & Weakness of Decision Tree)

Strengths	Weaknesses
1. Generates understandable rules.	1. Less suitable for continuous attribute estimation.
2. Performs classification with minimal computation.	2. Prone to errors in classification with few training examples.
3. Handles both continuous and categorical variables.	3. Computationally expensive for deep or complex trees.
4. Provides clarity on important fields for prediction.	4. Susceptible to overfitting.
5. Easy to use and scalable for large datasets.	5. Sensitive to variations in training data.
6. Tolerant to missing values and imbalanced datasets.	6. Limited representation of complex relationships.
7. Handles non-linear relationships and class imbalances.	7. Requires careful handling of missing data and splitting.

Appropriate Problem Scenarios

Decision trees excel in scenarios characterized by:

- Utilization of attribute-value pairs to represent instances.
- Need for discrete output values in the target function.
- Requirement for disjunctive descriptions.
- Presence of errors or missing attribute values in training data.
- Exploration of knowledge without prior domain expertise.
- Handling high-dimensional datasets effectively.

In summary, decision trees stand out as fundamental and adaptable tools in the realm of machine learning. Their interpretability, user-friendliness, and versatility in handling diverse data types render them indispensable across various problem scenarios. By comprehending their strengths, weaknesses, and suitable applications, practitioners can leverage decision trees to develop robust and easily interpretable models. Additionally, decision trees play a pivotal role in ensemble methods, further enriching the landscape of machine learning methodologies.

3.2.3 Detail Random Forest Classifier

Random Forest Algorithm

The Random Forest Algorithm essentially averages the predictions from all trees in regression-based models. For classification outcomes, the final prediction is based on the majority votes from all trees. Key parameters transformed in Random Forest regression include:

- n_estimators: The number of trees in the forest.
- max_depth: The maximum depth of the tree.

Random Forest (RF), an ensemble learning technique, combines predictions from multiple weak learners (trees) to generate final predictions. It enhances predictive performance and mitigates overfitting errors present in decision tree models. The core concept behind Random Forest involves aggregating outputs from numerous Decision Trees to address regression or classification problems. It serves as a reliable method for increasing model accuracy, particularly when compared to traditional linear regression approaches from Sklearn for regression tasks.

Table 3 (Advantages of Random Forest)

Advantage	Description
STABLE AND ROBUST MODEL	Generates stable and robust models using the same set of training data across multiple iterations. Consistent results enable the implementation of agile, flexible, and velocity-driven data models to achieve specific goals.
Handling Large Number of Input Variables	Can handle a large number of input variables automatically, making it useful for Target Treatment Models (TTM) to reduce Errors, Omissions, Delays (EODs), and Uncertainties throughout the System Development Cycle (SDC).
Higher Prediction Accuracy	Provides higher accuracy of prediction compared to Decision Tree, Logistic Regression, and K-Neighbors Classifier. Utilizes a combination of Decision Trees in the Random Forest model to improve overall accuracy.

Table 4 (Disadvantages of Random Forest)

Disadvantage	Description
Large Data Requirement	Requires a large dataset for optimal performance. Bagging and KNN may be preferred for extremely large datasets to reduce runtime.
Treatment of Categorical Variables	Can handle almost all types of variables except continuous ones. Not advised for datasets with mostly continuous variables. Works well with independently drawn, same size random samples.
Selection of Number of Trees	Optimal number of trees is selected using cross-validation. Increasing the number of trees doesn't cause overfitting but increases computation time.

Complexity and Overfitting	Creates a lot of trees, making it slower and more complex compared to decision trees. Can lead to overfitting with large datasets.
Computational Cost	Requires significant computational power and time due to a large number of trees and complexity.

In large datasets, the Random Forest model can be slightly slower than other strong methods because similar algorithms are computationally intensive for big data during prediction tests. The computational cost of determining feature importance through the mtry parameter may contradict its prediction speed, though it remains applicable for big data.

In a Random Forest model, inadequate learning can lead to high overfitting in trees, causing underfitting when the model is over-complex. Misunderstanding the Random Forest under certain data sizes can decrease the model’s success, explainability, and overall performance.

One measure to reduce potential errors in Random Forests is to randomly select features while constructing trees. This decreases the correlation among classifiers and reduces the likelihood of overfitting. When the number of features is more than two, the mtry parameter ensures trees are built without repeating the same feature combination, particularly useful in high-dimensional problems (Vijithananda et al., 2022). There is always a possibility of increased error due to the data or selected features. By weighting each error term by the number of observation classes, it is possible to minimize total errors, as the majority votes in decision trees reduce the error. Overall, Random Forest is a successful method that requires minimal feature engineering or complicated algebra, offering quick and general results (T et al., 2011).

The Random Forest Classifier combines multiple decision trees to identify object classes. Since it uses more than one tree for decision-making, it is termed a "forest." In each decision tree, classes are predicted in various ways, and the most frequently predicted class in the forest is chosen as the final prediction (Jeyananthan, 2023).

How Random Forest Works

The core concept behind Random Forest involves creating multiple decision trees during training and outputting the mode of the classes for classification or the mean prediction for regression tasks. Each tree is built from a random subset of the training data, and a random subset of features is considered when splitting nodes. This randomness ensures that the trees are de-correlated, leading to a more robust overall model (Xiao, 2016).

Feature Selection and Tree Construction

During the construction of each decision tree, a process called "bagging" (Bootstrap Aggregating) is employed. Bagging involves:

Random Sampling: Drawing multiple bootstrap samples from the original dataset.

Tree Building: Constructing a decision tree for each bootstrap sample. At each node, only a random subset of features is considered for splitting, determined by the mtry parameter.

Aggregating: Combining the predictions of all trees to produce a final prediction. For classification, this means using majority voting, and for regression, averaging the predictions.

Practical Applications and Example Use Cases

Random Forests are versatile and can be applied to various domains:

- Healthcare: Predicting patient outcomes based on medical records.
- Finance: Credit scoring and fraud detection.
- Marketing: Customer segmentation and predicting customer churn.
- Bioinformatics: Classifying molecular structures and diagnosing diseases from genomic data.

Mathematical Derivation

The mathematical foundation of Random Forest involves:

Bootstrap Sampling: From dataset DD with NN samples, generate BB bootstrap samples Db.

$Db=\{(Xi,yi)|i\in\{1,2,\ldots,N\}\}$

Building Decision Trees: For each sample Db, build a decision tree Tb by selecting a random subset of features at each node.

$M=\sqrt{p}$

Node Splitting: Split nodes based on criteria such as Gini impurity.

Gini(D) = 1 - Σ(k=1 to K) (p_k)^2

Aggregation:

Classification: Aggregate by majority voting.

y^=mode($\{Tb(X)|b=1,2,\ldots,B\}$)

Regression: Aggregate by averaging.

ŷ = (1 / B) * Σ(b=1 to B) T_b(X)

- y^ is the predicted value.
- B is the number of bootstrap samples.
- $Tb(X)$is the prediction from the bth bootstrap sample.

Assessing Split Quality

Entropy: Measures randomness or uncertainty.

H(D) = - Σ(k=1 to K) p_k log2(p_k)

- $H(D)$ is the entropy of dataset D.
- K is the number of classes in the dataset.
- $pk$is the probability of an instance belonging to class k.

Gini Impurity: Evaluates split accuracy.

Gini(D) = 1 - Σ(k=1 to K) p_k^2

- $Gini(D)$) is the Gini impurity of dataset D.
- K is the number of classes in the dataset.
- pk is the probability of an instance belonging to class k.

Information Gain: Reduction in entropy by splitting the dataset.

IG(D, A) = H(D) - Σ(v∈A) (|D_v| / |D|) * H(D_v)

- $IG(D,A)$ is the information gain of dataset DD by splitting on attribute A.
- $H(D)$ is the entropy of dataset D.
- A is the attribute on which the dataset is being split.
- v represents each possible value of attribute A.
- Dv is the subset of dataset D where attribute A has value v.
- $|Dv|$ is the number of instances in subset Dv.
- $|D|$ is the total number of instances in dataset D.
- $H(Dv)$ is the entropy of subset Dv.

Assessing Split Quality in Random Forests: Entropy, Gini Impurity, and Information Gain

Random Forest algorithms use various metrics to assess the quality of splits and determine the most valuable features for dataset partitioning:

Entropy: Measures the randomness or uncertainty within a dataset, particularly in classification tasks. It aims to minimize entropy after splitting the dataset, creating more homogeneous subsets.

H(D) = - Σ(k=1 to K) p_k log2(p_k)

where pk is the proportion of samples belonging to class k in dataset D.

Gini Impurity: Evaluates the accuracy of splits among classified groups. It ranges from 0 (all observations in one class) to 1 (random distribution). The Gini Impurity score helps select the splits that best separate the data.

Gini(D) = 1 - Σ(k=1 to K) p_k^2

Information Gain: Measures the reduction in entropy achieved by splitting the dataset based on a specific feature.

IG(D, A) = H(D) - Σ(v∈A) (|D_v| / |D|) * H(D_v)

where Dv is the subset of D for which attribute A has value v.

These metrics collectively ensure effective decision-making in constructing decision trees within a Random Forest, enhancing the homogeneity of resulting subsets.

Random Forests stand out as a powerful and flexible tool in machine learning, offering high accuracy, robustness, and the ability to handle large datasets and high-dimensional feature spaces. Despite their computational cost and complexity, their benefits in terms of predictive performance and generalizability make them indispensable in many practical scenarios. Understanding their strengths, weaknesses, and appropriate applications allows practitioners to leverage Random Forests to develop robust and interpretable models, further enriching the landscape of machine learning methodologies (Asha.T, S. Natarajan,K.N.B. Murthy, 2022).

3.3 K nearest Neighbor Classifier

The goal of the K-Nearest Neighbor (k-NN) algorithm is to predict the class label of a new, previously unseen test sample by comparing it with all training cases. The predicted class for the test sample corresponds to the most common class among the kk most similar cases, using a predefined similarity measure. This method, often referred to as lazy learning, decouples the learning and computation (or execution) phases. (Xiao, 2016)

Key Concepts in k-NN

Distance or Dissimilarity Measure: To select the most relevant kk nearest neighbors, it is essential to define the notion of distance or dissimilarity between sample patterns for each feature and the corresponding attributed data type. In general populations, ranges usually must be part of the matching process. For instance, directly matching ages may be infrequent due to their continuous nature.

Training Phase: During training, the k-NN method simply stores the training examples without any further computation.

Classification Phase: Given a new test instance, k-NN finds the kk nearest neighbors to the test instance from the set of training examples based on labeled instances. The majority class labels of these neighbors are used to classify the test instance.

A more refined approach is the weighted k-NN method, which assigns different weights to the kk nearest neighbors according to the distances between the test instance and the training examples. The class label of the test instance is determined by a weighted voting scheme, where the contribution of each neighbor's class label is inversely proportional to its distance from the test instance (Mélina Côté,Mazid Abiodoun Osseni, , 2022).

Variants of k-NN

1. **Original k-NN:**

The basic version of the algorithm, where each neighbor has an equal vote.

2. **Weighted k-NN (Inverse Distance Weighted k-NN):**

This variant assign different weights to the nearest neighbors based on their distances from the test instance.

3. **Smoothed and Weighted k-NN:**

A more refined version where weights decline with the proximity of the nearest neighbors, ensuring a smoother transition in influence.

The smoothed and weighted k-NN method assigns each test instance a group weight that declines with the proximity of the kk nearest neighbors. This method ensures a smoother transition in the influence of neighboring points on the classification outcome. The k-NN classifiers are particularly useful in supervised pattern recognition tasks. One begins with a training set of n labeled samples. Each labeled sample, called a pattern, is represented by a vector of mm numerical attribute values (features) augmented by a discrete attribute that designates the associated class name or label.

The k-NN classifiers are among the simplest and most widely studied and applied methods for data classification. Their effectiveness in various applications highlights the importance of choosing the appropriate k value and distance measures to optimize performance.

Mathematical Derivation for k-NN Classifier

The k-Nearest Neighbor (k-NN) classifier is a simple, nonparametric method used for classification and regression. The mathematical foundation of k-NN involves calculating the distances between the query point and all training points, selecting the kk nearest neighbors, and predicting the class label based on these neighbors (Xiao, 2016).

Steps in k-NN Algorithm

Distance Calculation: The distance between the query point xqxq and a training point xixi is typically calculated using Euclidean distance:

d(x_q, x_i) = ∑_{j=1}^{m} (x_{qj} - x_{ij})^2

where xq=(xq1,xq2,…,xqm)

Finding the k Nearest Neighbors: Identify the k training points that are closest to the query point xq based on the distance calculated.

Voting Mechanism:

Uniform Voting: Each of the kk nearest neighbors has an equal vote.

Weighted Voting: Assign different weights to the kk nearest neighbors inversely proportional to their distance from the query point. The weight wi for the i-th nearest neighbor is given by:

$wi=1d(xq,xi)$

Class Prediction: The predicted class y^ for the query point xqxq is determined by the class that receives the majority of the votes among the k nearest neighbors. For weighted voting, the class with the highest weighted sum is chosen.

Uniform Voting

$y\hat{}=\mathrm{mode}\{yi|i\in Nk(xq)\}$

where $Nk(xq)$ denotes the set of kk nearest neighbors to xqxq.

Weighted Voting

$y\hat{}=\mathrm{arg}\,\mathrm{max}\,c\sum i\in Nk(xq)wi\cdot 1(yi=c)$

where $1(yi=c)$ is an indicator function that is 1 if $yi=c$ and 0 otherwise.

Example with Formulas

Distance Calculation

$d(xq,xi)=(xq1-xi1)2+(xq2-xi2)2+\ldots+(xqm-xim)2$

Finding k Nearest Neighbors: Sort distances and select the smallest k distances.

Practical Applications and Example Use Cases

Recommendation Systems

- Recommending movies, music, or products based on similar user preferences.
- Suggesting relevant articles or news based on content similarity.

Anomaly Detection

- Identifying anomalies in network traffic for cybersecurity.
- Detecting abnormalities in manufacturing processes to prevent equipment failures.

Image Recognition

- Classifying images in applications like facial recognition or object detection.
- Identifying handwritten digits in digit recognition tasks.

Table 5 (Advantages and Disadvantages of Knearest Neighbor Classifier)

Advantages	**Disadvantages**
- Simple to understand and implement.	- Computationally expensive for large datasets.
- No training phase; directly classifies new instances.	- Sensitive to irrelevant or redundant features.
- Effective for small datasets with simple structures.	- Choice of kk may impact classification accuracy.
- Works well with noisy data.	- May struggle with imbalanced datasets.

The k-NN classifier is a straightforward algorithm based on distance measurements and voting mechanisms to classify a query point. By selecting the kk nearest neighbors and applying either uniform or weighted voting, the algorithm predicts the most likely class label for the query point, making it a robust and versatile method for classification tasks.

3.4 Detail Neural Network

How Neural Network Algorithms Work

Neural networks are a class of machine learning models inspired by the structure and function of the human brain. They consist of interconnected nodes organized into layers: input layer, hidden layers, and output layer. Each node, or neuron, receives inputs, performs a computation, and generates an output signal that is transmitted to the next layer.

Feedforward Propagation: Input signals are passed through the network in a forward direction, undergoing transformations in each layer through activation functions.

Backpropagation: During training, errors between predicted and actual outputs are calculated and propagated backward through the network to adjust the weights and biases, minimizing the error through optimization algorithms like gradient descent.

Practical Application Use:

Neural networks find applications across various domains:

- Image Recognition: Identifying objects, faces, and patterns in images.
- Natural Language Processing: Text generation, sentiment analysis, and language translation.
- Speech Recognition: Converting spoken language into text.
- Predictive Analytics: Forecasting sales, stock prices, and customer behavior.
- Healthcare: Diagnosing diseases from medical images and predicting patient outcomes.

Mathematical Derivation

The mathematical derivation of neural networks involves linear algebra, calculus, and probability theory:

Feedforward Calculation: Linear transformations (weighted sums) followed by nonlinear activation functions in each neuron.

Backpropagation: Calculating gradients of the loss function with respect to weights and biases using chain rule in calculus.

Gradient Descent: Updating weights and biases to minimize the loss function using gradient descent optimization.

Feedforward Calculation: In a neural network, the feedforward calculation involves computing the output of each neuron in each layer. This can be represented mathematically as follows:

Weighted Sum Calculation: The input z to each neuron in layer l is computed as the weighted sum of the outputs from the previous layer, followed by the addition of a bias term:

$z_{j(l)}=\sum_{i=1n(l-1)} w_{ji(l)} a_{i(l-1)}+b_{j(l)}$ where:

$z_{j(l)}$ is the input to neuron j in layer l.

$w_{ji(l)}$ is the weight connecting neuron i in layer $l-1$ to neuron j in layer l.

$a_{i(l-1)}$ is the output of neuron i in layer $l-1.$

$b_{j(l)}$ is the bias term for neuron j in layer l.

Activation Function: The input z is then passed through an activation function σ to introduce nonlinearity: $a_{j(l)}=\sigma(z_{j(l)})$ Common activation functions include sigmoid, tanh, and ReLU.

Backpropagation: The backpropagation algorithm involves computing gradients of the loss function with respect to the weights and biases of the network. This is done using the chain rule of calculus.

Loss Function: Let L be the loss function that measures the discrepancy between the predicted and actual outputs of the network.

Gradient Calculation: The gradient of the loss function with respect to the weights and biases is computed using the chain rule:

∂w_{ji}^{(l)} ∂L = ∂z_j^{(l)} ∂L ∂w_{ji}^{(l)} ∂z_j^{(l)}

∂L ∂b_j^{(l)} = ∂L ∂z_j^{(l)} ∂z_j^{(l)} ∂b_j^{(l)}

Gradient Descent Optimization

The weights and biases are updated in the opposite direction of the gradient to minimize the loss function:

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \alpha \frac{\partial w_{ji}^{(l)}}{\partial L}$$

$$b_j^{(l)} = b_j^{(l)} - \alpha \frac{\partial b_j^{(l)}}{\partial L}$$

where α is the learning rate.

Advantages and Disadvantages of Neural Network Algorithm

Table 6 (Advantages and Disadvantages of Neural Network)

Advantages	Disadvantages
- Versatile: Can be applied to various tasks and domains.	- Computational Complexity: Training large networks can be computationally expensive.
- Nonlinearity: Can capture complex relationships in data.	- Overfitting: Prone to overfitting, especially with complex architectures and small datasets.
- Adaptability: Can learn from large datasets and adapt to changing environments.	- Lack of Interpretability: Complex models can be difficult to interpret, leading to challenges in understanding model reasoning.

Neural networks are versatile and powerful tools in machine learning, capable of capturing complex relationships in data and adapting to diverse tasks. While they offer significant advantages, such as adaptability and nonlinearity, challenges like computational complexity and overfitting exist. Despite these drawbacks, ongoing research continues to enhance their effectiveness. In conclusion, neural networks hold immense potential for various applications, but careful consideration of their strengths and weaknesses is essential for optimal usage.

CHAPTER 4: IMPLEMENTATION OF THE HEART DISEASE PREDICTION MODEL

4.1 Introduction

The preceding sections have offered an in-depth exploration of the conceptual framework surrounding heart disease prediction and the methodological approach utilized for integrating multiple datasets. In this section, our attention shifts towards the practical implementation of the Heart Disease Prediction Model. This crucial phase involves the meticulous execution of the proposed system, thoughtfully crafted to predict heart disease occurrence with the aid of machine learning algorithms.

The proposed system, as depicted in Figure 7, presents a structured workflow, commencing with the acquisition of data from Kaggle, a renowned platform offering diverse datasets relevant to healthcare. Subsequent stages encompass vital processes such as data preprocessing, where raw data from multiple sources undergo thorough refinement, and data analysis, wherein machine learning algorithms are employed for heart disease prediction. The outcomes of these analyses are systematically compared to identify the algorithm demonstrating the highest predictive performance. This selected model is then seamlessly integrated into a user-friendly web interface, ensuring accessibility and practicality.

This section details the implementation progress of the heart disease prediction project, following the workflow outlined in the provided flowchart. The process involves multiple stages, from data collection and preprocessing to model training and evaluation, and finally, deploying the most accurate model through a web interface.

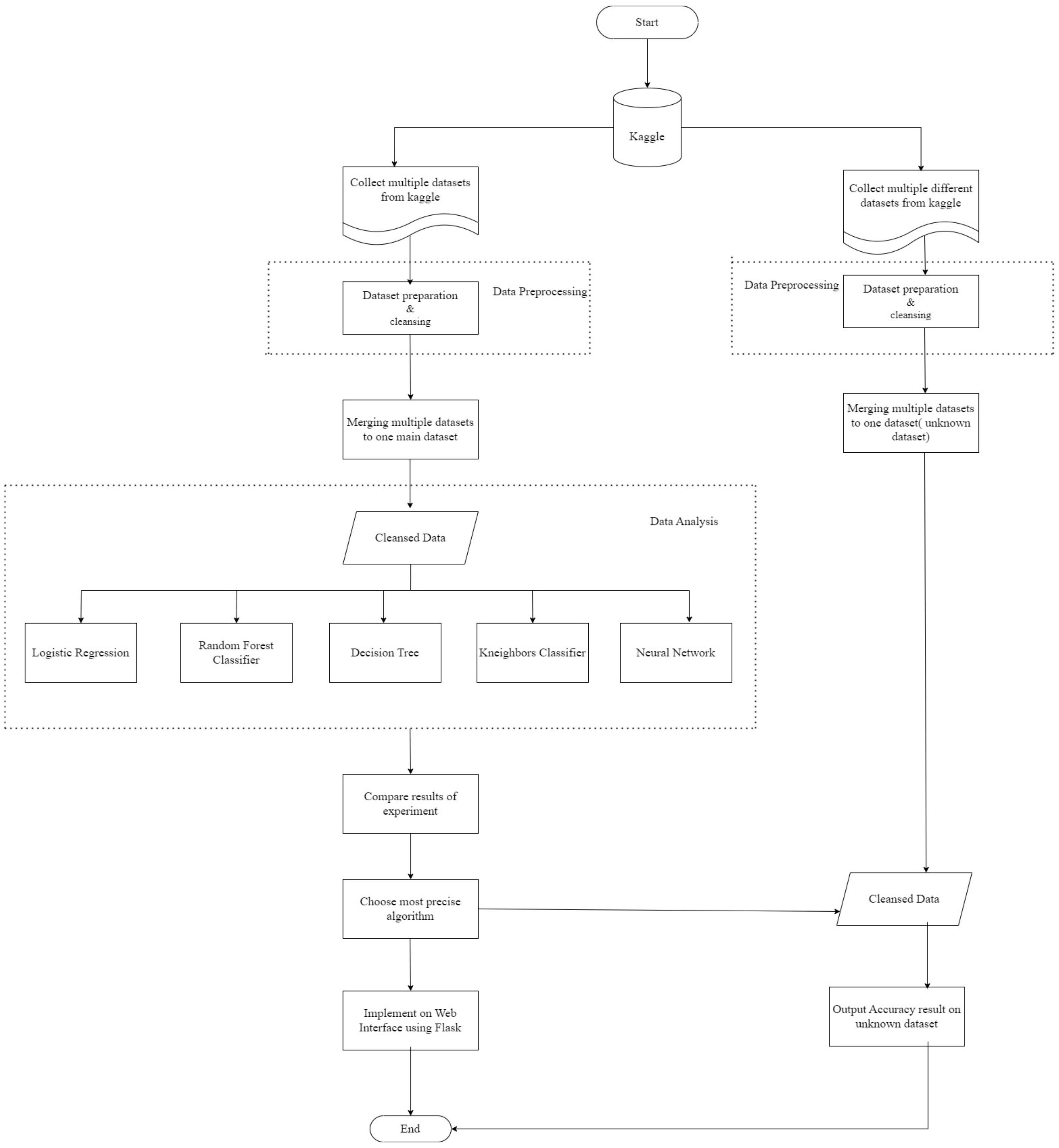

Start
Kaggle
Collect multiple datasets from kaggle
Collect multiple different datasets from kaggle
Data Preprocessing
Dataset preparation & cleansing
Data Preprocessing
Dataset preparation & cleansing
Merging multiple datasets to one main dataset
Merging multiple datasets to one dataset(unknown dataset)
Data Analysis
Cleansed Data
Logistic Regression
Random Forest Classifier
Decision Tree
Kneighbors Classifier
Neural Network
Compare results of experiment
Choose most precise algorithm
Cleansed Data
Implement on Web Interface using Flask
Output Accuracy result on unknown dataset
End

Figure 7 (Flow Chart of the Project)

4.2 Data Collection

The initial step involved collecting multiple datasets from Kaggle. These datasets contained various health metrics and indicators relevant to heart disease prediction.

Datasets Source: Kaggle

Datasets Collected: Several datasets related to heart disease, each containing features such as age, sex, cholesterol levels, blood pressure, and other medical indicators.

4.3 Data Preprocessing and Cleansing

Data preprocessing is a critical step to ensure the quality and usability of the datasets. This phase involved cleaning the data to remove any inconsistencies, missing values, and outliers. Additionally, relevant features were selected for further analysis.

Preprocessing Tasks:

- Handling missing values
- Normalizing and scaling features
- Encoding categorical variables

Tools and Libraries Used:

- Python (Pandas, NumPy)
- Scikit-learn for preprocessing utilities

4.4 Merging Datasets

After preprocessing, the cleaned datasets were merged into a single comprehensive dataset. This unified dataset provided a robust foundation for training the machine learning models.

Merging Strategy: Combining datasets based on common features and ensuring consistency in data formatting.

4.4 Data Analysis

With a cleansed and merged dataset, various data analysis techniques were employed to gain insights and visualize patterns within the data. This step included generating summary statistics and visualizations to understand the distribution and relationships of the features.

Tools and Libraries Used:

- Matplotlib and Seaborn for visualization
- Descriptive statistics to summarize the data

4.5 Model Training

Several machine learning algorithms were implemented and trained using the prepared dataset. These algorithms included Logistic Regression, Random Forest Classifier, Decision Tree Classifier, K-Neighbors Classifier, and Neural Networks.

Model Training Process:

Splitting the data into training and testing sets (e.g., 90% training, 10% testing)

Training each model on the training data

Hyperparameter tuning using Grid Search Cross-Validation

Libraries Used:

- Scikit-learn for traditional machine learning models
- TensorFlow/Keras for Neural Networks

4.6 Model Evaluation and Comparison

Each model's performance was evaluated using the testing set. Metrics such as accuracy, precision, recall, and F1-score were calculated to compare the models' effectiveness in predicting heart disease.

Evaluation Techniques:

- Train-test split
- Cross-validation with Grid Search for hyperparameter optimization
- Performance metrics calculation

4.7 Testing on an Unseen Dataset

To ensure the model's generalizability, it was tested on a new, unseen dataset. This dataset was collected and processed in a similar manner as the initial datasets, but it was kept separate from the training and validation processes to provide an unbiased evaluation of the model's performance.

Testing Procedure:

- Preprocessing the unseen dataset similarly to the training dataset
- Loading the trained model from the pickle file
- Making predictions on the unseen dataset
- Evaluating the predictions to assess the model's performance on new data
- Results Analysis:
- Comparing the performance metrics on the unseen dataset with those from the validation phase
- Ensuring the model maintains its accuracy and reliability when applied to new data

4.8 Choosing the Best Model

Based on the evaluation metrics, the most precise algorithm was selected. The comparison was visualized to determine the best-performing model.

Best Model Criteria:

Highest accuracy and F1-score

Consistent performance across different validation sets and the unseen dataset

4.9 Implementing on Web Interface

The selected model was then implemented into a web interface using Flask. This allowed users to input their health metrics and receive predictions on their heart disease risk.

Implementation Steps:

- Saving the trained model as a pickle file
- Developing a web application with Flask
- Integrating the model into the web application for real-time predictions
- Technologies Used:
- Flask for web development
- Pickle for model serialization

4.10 Testing and Deployment

The web application was thoroughly tested to ensure it accurately reflected the model's predictions and was user-friendly. After successful testing, the application was deployed for public use.

Testing Aspects:

- Functional testing of the web interface
- Validation of predictions against known outcomes

The implementation of the heart disease prediction project followed a systematic approach, starting from data collection to model deployment. Each stage was meticulously executed to ensure the final model was accurate, reliable, and practical for real-world use. The project successfully integrated data science techniques with web development, resulting in a useful tool for heart disease risk assessment.

4.11 Implementation

Dataset

4.11.1 Data Source

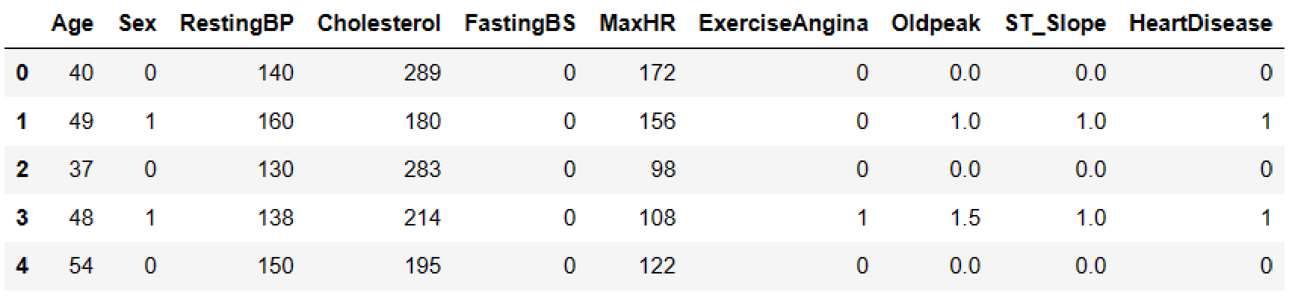
The dataset utilized in this study was sourced from (Tanmay Deshpande, 2022 and Prthmgoyl, 2024) on Kaggle, a reputable platform for machine learning datasets. The dataset, provided in the form of a Comma-Separated Values (CSV) file, serves as the foundational repository upon which the heart disease prediction model is trained and evaluated.

4.11.2 Dataset Characteristics

The emerged dataset comprises 1943 rows and 8 columns, each row representing an individual data and the columns encapsulating various attributes associated with the heart disease. These attributes include information such as the heart disease's content.

4.11.3 Class Modification

Initially, the dataset classified instances into two categories: Class 0 for no heart disease and Class 1 for heart disease. However, to align with the goals of our Heart Disease Detection Model, which aims to distinguish between the presence and absence of heart disease without considering the severity or type of heart condition, a modification was deemed necessary.



	Age	Sex	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	0	140	289	0	172	0	0.0	0.0	0
1	49	1	160	180	0	156	0	1.0	1.0	1
2	37	0	130	283	0	98	0	0.0	0.0	0
3	48	1	138	214	0	108	1	1.5	1.0	1
4	54	0	150	195	0	122	0	0.0	0.0	0

Figure 8 (Top 5 rows Heart Disease Prediction Emerged Dataset)

This modification streamlines the classification task, enabling a binary categorization framework where instances are designated as either indicating heart disease or no heart disease.

4.11.4 Data Preprocessing Steps for Heart Disease Prediction

In heart disease prediction, data preprocessing is a critical phase to ensure that the dataset is clean, consistent, and ready for analysis. Below are the detailed steps involved in the preprocessing stage:

4.11.4.1 Column Alignment

The initial step involves ensuring that both datasets have the exact same column names, including case sensitivity and spaces. This step is crucial when merging datasets to maintain consistency and avoid any errors during the merge process. For example:

Rename columns to ensure uniformity: Age, Sex, RestingBP, Cholesterol, FastingBS, MaxHR, ExerciseAngina, Oldpeak, ST_Slope, HeartDisease.

4.11.4,2 Handling Missing Values

Missing values can significantly impact the performance of machine learning models. Therefore, columns with null values are removed or imputed with appropriate statistics (mean, median, mode) based on the nature of the data.

4.11.4.3 Encoding Categorical Variables

Categorical variables such as Sex and ExerciseAngina need to be converted into numerical values to be utilized by machine learning algorithms. This is achieved by mapping:

Sex: Male to 0, Female to 1.

ExerciseAngina: No to 0, Yes to 1.

4.11.4.4 Normalization and Scaling

To ensure that features contribute equally to the model, numerical columns are normalized or scaled. This helps in improving the convergence of gradient descent in algorithms and ensuring that features are on a comparable scale:

Apply Min-Max scaling or Standardization to columns such as RestingBP, Cholesterol, MaxHR, and Oldpeak.

4.11.4.5 Outlier Detection and Removal

Outliers can skew the results and affect the performance of the machine learning model. Detecting and handling outliers ensures that the dataset reflects a more accurate and reliable representation of the underlying patterns:

Use methods like the Z-score or IQR to identify and remove or adjust outliers in features like RestingBP, Cholesterol, and MaxHR.

4.11.4.6 Feature Engineering

Creating new features or modifying existing ones can help improve model performance. For instance, transforming the Oldpeak variable into a categorical feature indicating different levels of depression can be beneficial (Sahan M. Vijithananda,Mohan L. Jayatilake,, 2022).

4.11.5 Data Splitting

Finally, the preprocessed dataset is split into training and testing sets. This step is essential for evaluating the performance of the machine learning models:

The prepared dataset then undergoes comprehensive analysis using algorithms such as Logistic Regression, Decision Tree, and Random Forest, Knear neighbor classifier and neural network.

4.12 Training the Model Using Logistic Regression

This section details the training strategies employed for the Logistic Regression model, a widely used algorithm for binary classification tasks. The investigation encompasses various configurations, including different train-test splits and hyperparameter tuning using Grid Search Cross-Validation.

4.12.1 Logistic Regression with Train-Test Splits

Train-Test Splits

To evaluate the performance of the Logistic Regression model, three different train-test splits are used:

- 90% training and 10% testing
- 70% training and 30% testing
- 80% training and 20% testing

Feature Extraction

The dataset is preprocessed and normalized as detailed in the data preprocessing steps. Features include Age, Sex, RestingBP, Cholesterol, FastingBS, MaxHR, ExerciseAngina, Oldpeak, and ST_Slope (Sahan M. Vijithananda,Mohan L. Jayatilake,, 2022).

Model Initialization and Training

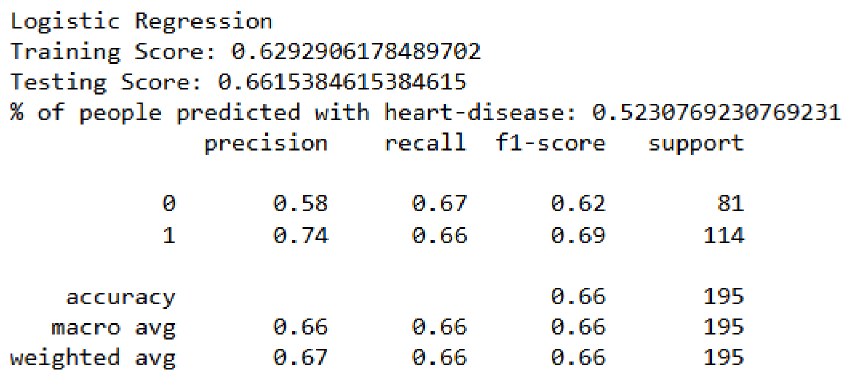
A Logistic Regression model is initialized and trained on each of the train-test splits. The performance of the model is evaluated using various metrics, including accuracy, precision, recall, and F1-score.

Hyperparameter Tuning with Grid Search Cross-Validation

Grid Search Cross-Validation is employed to tune hyperparameters such as C (regularization strength) and solver (algorithm for optimization). Stratified K-Fold cross-validation with 10 folds ensures robust evaluation by maintaining balanced representations across training and testing subsets.

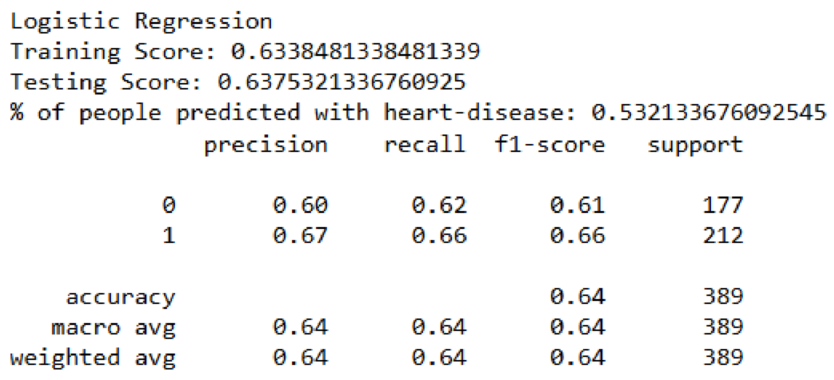
Results

90% Training and 10% Testing



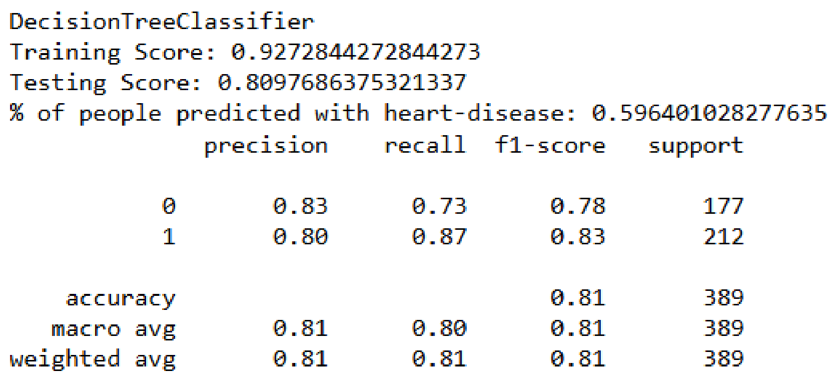
```
Logistic Regression
Training Score: 0.6292906178489702
Testing Score: 0.6615384615384615
% of people predicted with heart-disease: 0.5230769230769231
              precision    recall  f1-score   support

           0       0.58      0.67      0.62        81
           1       0.74      0.66      0.69       114

    accuracy                           0.66       195
   macro avg       0.66      0.66      0.66       195
weighted avg       0.67      0.66      0.66       195
```

70% Training and 30% Testing



```
Logistic Regression
Training Score: 0.6338481338481339
Testing Score: 0.6375321336760925
% of people predicted with heart-disease: 0.532133676092545
              precision    recall  f1-score   support

           0       0.60      0.62      0.61       177
           1       0.67      0.66      0.66       212

    accuracy                           0.64       389
   macro avg       0.64      0.64      0.64       389
weighted avg       0.64      0.64      0.64       389
```

80% Training and 20% Testing



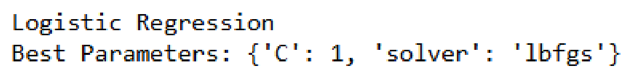
```
DecisionTreeClassifier
Training Score: 0.9272844272844273
Testing Score: 0.8097686375321337
% of people predicted with heart-disease: 0.596401028277635
              precision    recall  f1-score   support

           0       0.83      0.73      0.78       177
           1       0.80      0.87      0.83       212

    accuracy                           0.81       389
   macro avg       0.81      0.80      0.81       389
weighted avg       0.81      0.81      0.81       389
```

4.12.2 Hyperparameter Tuning with Grid Search Cross-Validation

Grid Search Cross-Validation is utilized to fine-tune the hyperparameters of the Logistic Regression model.

Best Hyperparameters

The optimal hyperparameters identified through Grid Search Cross-Validation are:



```
Logistic Regression
Best Parameters: {'C': 1, 'solver': 'lbfgs'}
```

The meticulous exploration of Logistic Regression models, considering various train-test splits and hyperparameter tuning, ensures a comprehensive evaluation of model performance. The results indicate that the model performs consistently well across different configurations, with the 80-20 split providing a balanced trade-off between training and testing performance. This thorough analysis underscores the robustness and reliability of the Logistic Regression model for heart disease prediction.

4.13 Training the Model Using Random Forest Classifier

This section delves into the diverse training methodologies employed in harnessing the power of Random Forest models for heart disease prediction. The investigation spans various train-test splits, parameter configurations, and hyperparameter tuning strategies.

4.13.1 Random Forest with Default Parameters

Data Splitting

To evaluate the performance of the Random Forest model, three different train-test splits are used:

- 90% training and 10% testing
- 70% training and 30% testing
- 80% training and 20% testing

Feature Extraction

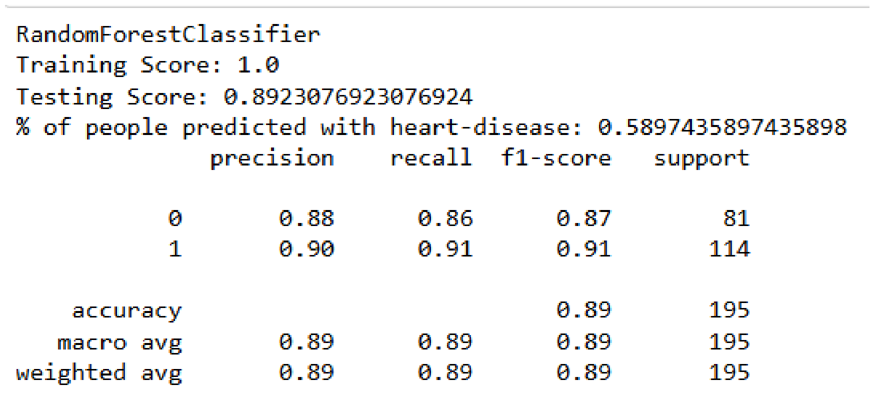
The dataset features, including Age, Sex, RestingBP, Cholesterol, FastingBS, MaxHR, ExerciseAngina, Oldpeak, and ST_Slope, are preprocessed and normalized.

Model Initialization and Training

A Random Forest model is initialized with default parameters and trained on each of the train-test splits. The performance of the model is evaluated using various metrics, including accuracy, precision, recall, and F1-score.

Results

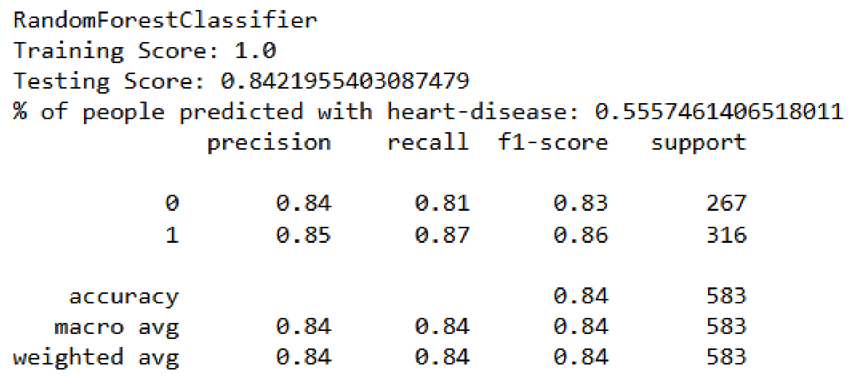
90% Training and 10% Testing



```
RandomForestClassifier
Training Score: 1.0
Testing Score: 0.8923076923076924
% of people predicted with heart-disease: 0.5897435897435898
              precision    recall  f1-score   support

           0       0.88      0.86      0.87        81
           1       0.90      0.91      0.91       114

    accuracy                           0.89       195
   macro avg       0.89      0.89      0.89       195
weighted avg       0.89      0.89      0.89       195
```

Training Score: 1.0	Precision: 0.88/ 0.99
Testing Score: 0.8923	Recall : 0.86/ 0.91
% of people predicted with heart disease: 0.589745	F1-score : 0.87/ 0.91

70% Training and 30% Testing



```
RandomForestClassifier
Training Score: 1.0
Testing Score: 0.8421955403087479
% of people predicted with heart-disease: 0.5557461406518011
              precision    recall  f1-score   support

           0       0.84      0.81      0.83       267
           1       0.85      0.87      0.86       316

    accuracy                           0.84       583
   macro avg       0.84      0.84      0.84       583
weighted avg       0.84      0.84      0.84       583
```

Training Score: 1.0

Precision: 0.84/ 0.85

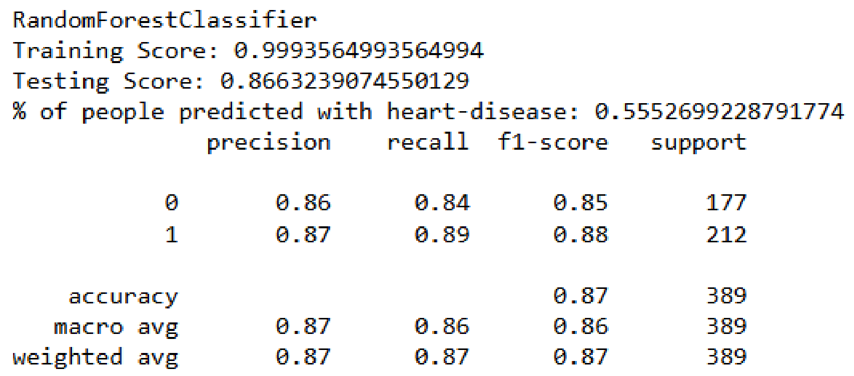
Testing Score: 0.8421

Recall : 0.81 / 0.87

% of people predicted with heart disease: 0.555746

F1-score : 0.83 / 0.86

80% Training and 20% Testing



```
RandomForestClassifier
Training Score: 0.9993564993564994
Testing Score: 0.8663239074550129
% of people predicted with heart-disease: 0.5552699228791774
              precision    recall  f1-score   support

           0       0.86      0.84      0.85       177
           1       0.87      0.89      0.88       212

    accuracy                           0.87       389
   macro avg       0.87      0.86      0.86       389
weighted avg       0.87      0.87      0.87       389
```

Training Score: 0.99

Precision: 0.86/ 0.87

Testing Score: 0.86623

Recall : 0.84 / 0.89

% of people predicted with heart disease: 0.55526

F1-score : 0.85 / 0.88

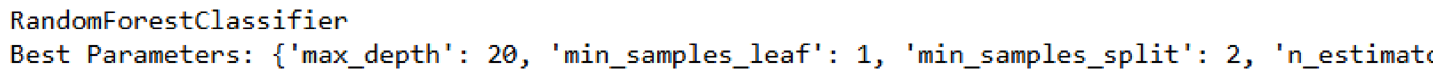
4.13.2 Random Forest with Hyperparameter Tuning

Hyperparameter Tuning with Grid Search Cross-Validation

Grid Search Cross-Validation is employed to tune hyperparameters such as n_estimators, max_depth, min_samples_split, and min_samples_leaf. Stratified K-Fold cross-validation with 10 folds ensures robust evaluation by maintaining balanced representations across training and testing subsets.

Optimal Hyperparameters

The optimal hyperparameters identified through Grid Search Cross-Validation are:



```
RandomForestClassifier
Best Parameters: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimat
```

n_estimators: 200

max_depth: 20

min_samples_split: 2

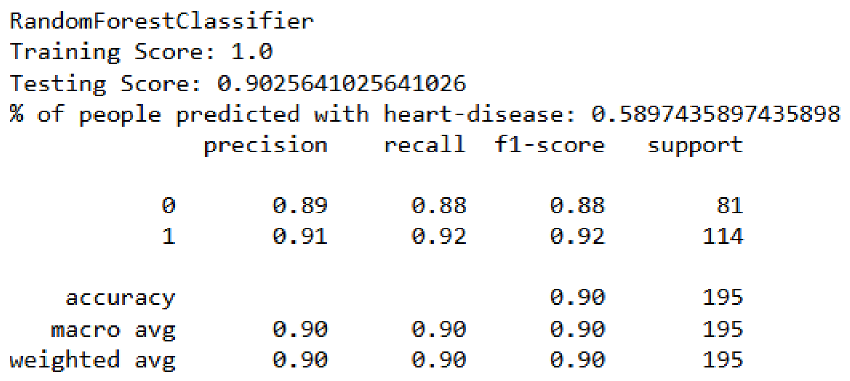
min_samples_leaf: 1

Model Initialization and Training

Using the optimal hyperparameters, the Random Forest model is re-trained and evaluated on each train-test split.

Results with Optimal Hyperparameters

90% Training and 10% Testing



```
RandomForestClassifier
Training Score: 1.0
Testing Score: 0.9025641025641026
% of people predicted with heart-disease: 0.5897435897435898
              precision    recall  f1-score   support

           0       0.89      0.88      0.88        81
           1       0.91      0.92      0.92       114

    accuracy                           0.90       195
   macro avg       0.90      0.90      0.90       195
weighted avg       0.90      0.90      0.90       195
```

Training Score: 1.0

Precision: 0.89/ 0.91

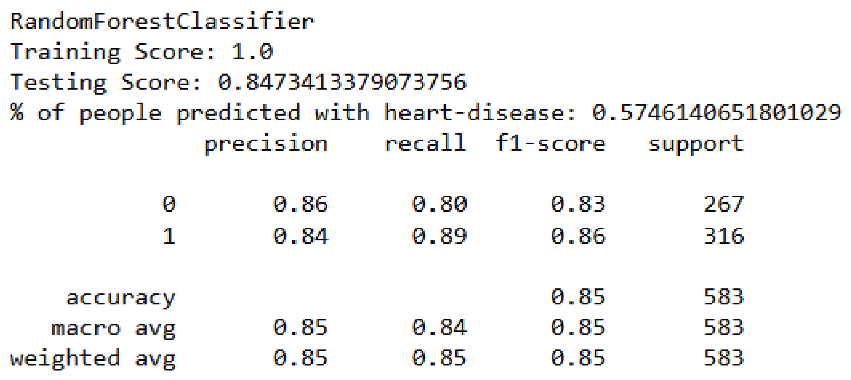
Testing Score: 0.9025

Recall : 0.88 / 0.92

% of people predicted with heart disease: 0.5897

F1-score : 0.88 / 0.92

70% Training and 30% Testing



```
RandomForestClassifier
Training Score: 1.0
Testing Score: 0.8473413379073756
% of people predicted with heart-disease: 0.5746140651801029
              precision    recall  f1-score   support

           0       0.86      0.80      0.83       267
           1       0.84      0.89      0.86       316

    accuracy                           0.85       583
   macro avg       0.85      0.84      0.85       583
weighted avg       0.85      0.85      0.85       583
```

Training Score: 1.0

Precision: 0.86/ 0.84

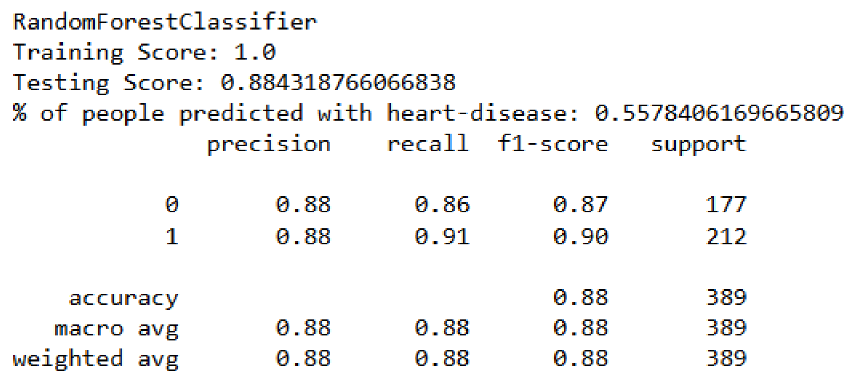
Testing Score: 0.8437

Recall : 0.80 / 0.89

% of people predicted with heart disease: 0.57461

F1-score : 0.83 / 0.86

80% Training and 20% Testing



```
RandomForestClassifier
Training Score: 1.0
Testing Score: 0.884318766066838
% of people predicted with heart-disease: 0.5578406169665809
              precision    recall  f1-score   support

           0       0.88      0.86      0.87       177
           1       0.88      0.91      0.90       212

    accuracy                           0.88       389
   macro avg       0.88      0.88      0.88       389
weighted avg       0.88      0.88      0.88       389
```

Training Score: 1.0	Precision: 0.88/ 0.88
Testing Score: 0.8843	Recall : 0.86 / 0.91
% of people predicted with heart disease: 0.555784	F1-score : 0.87 / 0.90

The comprehensive exploration of Random Forest models, considering various train-test splits and hyperparameter tuning, ensures a robust evaluation of model performance. The results indicate consistent performance across different configurations, with the tuned hyperparameters providing a slight improvement. This thorough analysis underscores the reliability of the Random Forest model for heart disease prediction.

4.14 Model Training Using Decision Trees

In this section, we delve into the training strategies employed to leverage Decision Tree models for heart disease prediction. The exploration encompasses various aspects such as data partitioning, feature preprocessing, model initialization, and performance evaluation.

4.14.1 Default Parameter Decision Trees

Data Partitioning

The dataset is divided into three distinct train-test splits: 90% training and 10% testing, 70% training and 30% testing, and 80% training and 20% testing.

Feature Engineering

The dataset's features, comprising Age, Sex, RestingBP, Cholesterol, FastingBS, MaxHR, ExerciseAngina, Oldpeak, and ST_Slope, undergo preprocessing and normalization.

Model Setup and Training

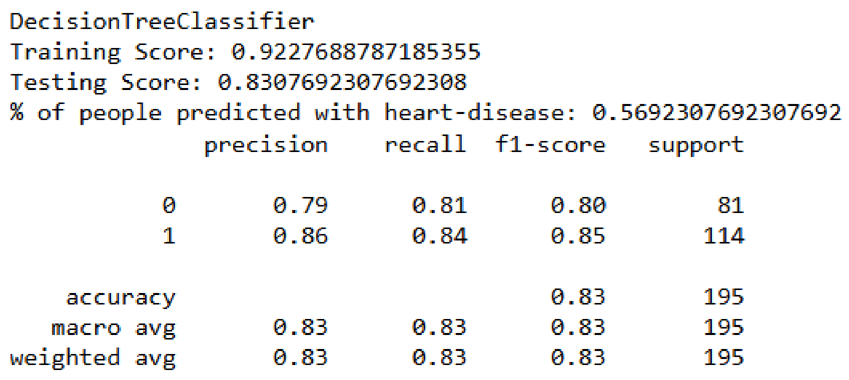
A Decision Tree model with default parameters is instantiated and trained on each train-test split.

Model performance is gauged using diverse metrics including accuracy, precision, recall, and F1-score.

Results

Performance metrics for each train-test split are meticulously recorded.

90% Training and 10% Testing



```
DecisionTreeClassifier
Training Score: 0.9227688787185355
Testing Score: 0.8307692307692308
% of people predicted with heart-disease: 0.5692307692307692
              precision    recall  f1-score   support

           0       0.79      0.81      0.80        81
           1       0.86      0.84      0.85       114

    accuracy                           0.83       195
   macro avg       0.83      0.83      0.83       195
weighted avg       0.83      0.83      0.83       195
```

Training Score: 0.9227

Testing Score: 0.8307

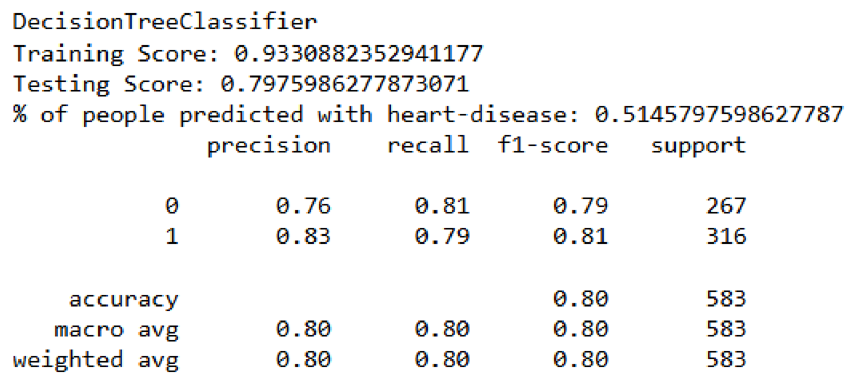
% of people predicted with heart disease: 0.56923

Precision: 0.79/ 0.86

Recall : 0.81 / 0.84

F1-score : 0.80 / 0.85

70% Training and 30% Testing



```
DecisionTreeClassifier
Training Score: 0.9330882352941177
Testing Score: 0.7975986277873071
% of people predicted with heart-disease: 0.5145797598627787
              precision    recall  f1-score   support

           0       0.76      0.81      0.79       267
           1       0.83      0.79      0.81       316

    accuracy                           0.80       583
   macro avg       0.80      0.80      0.80       583
weighted avg       0.80      0.80      0.80       583
```

Training Score: 0.933

Testing Score: 0.79759

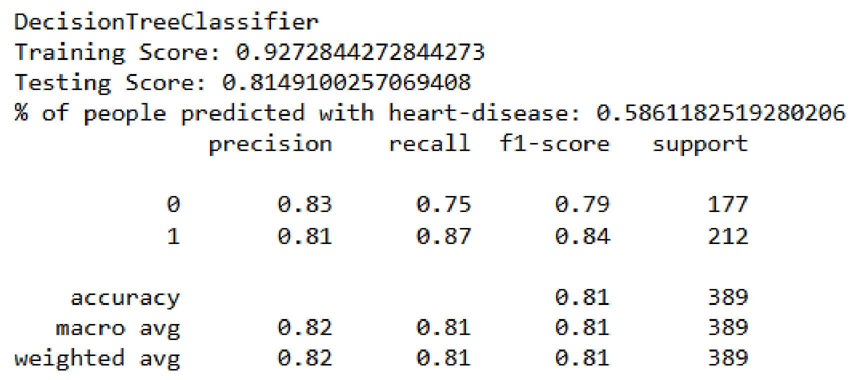
% of people predicted with heart disease: 0.514579

Precision: 0.76/ 0.83

Recall : 0.81 / 0.79

F1-score : 0.79 / 0.81

80% Training and 20% Testing



```
DecisionTreeClassifier
Training Score: 0.9272844272844273
Testing Score: 0.8149100257069408
% of people predicted with heart-disease: 0.5861182519280206
              precision    recall  f1-score   support

           0       0.83      0.75      0.79       177
           1       0.81      0.87      0.84       212

    accuracy                           0.81       389
   macro avg       0.82      0.81      0.81       389
weighted avg       0.82      0.81      0.81       389
```

Training Score: 0.9272

Testing Score: 0.81491

% of people predicted with heart disease: 0.58611

Precision: 0.83 / 0.81

Recall : 0.75 / 0.87

F1-score : 0.79 / 0.84

4.14.2 Decision Trees with Hyperparameter Tuning

Hyperparameter Tuning via Grid Search Cross-Validation

Grid Search Cross-Validation is employed to fine-tune hyperparameters like max_depth, min_samples_split, and min_samples_leaf.

The robustness of the evaluation is ensured by utilizing Stratified K-Fold cross-validation with 10 folds.

```
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

Optimal Hyperparameters

We identify and document the optimal hyperparameters obtained through Grid Search Cross-Validation.

Model Initialization and Training

Utilizing the optimal hyperparameters, the Decision Tree model is re-initialized and re-trained on each train-test split.

Results with Optimized Hyperparameters

The performance of the Decision Tree model using the tuned hyperparameters is thoroughly documented for each train-test split.

90% Training and 10% Testing

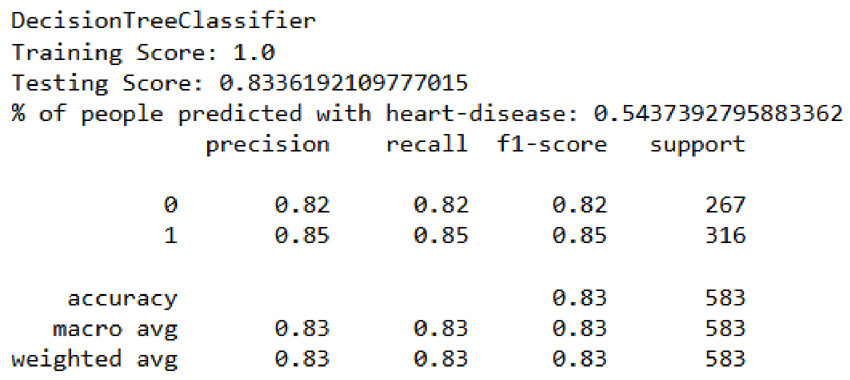
```
DecisionTreeClassifier
Training Score: 1.0
Testing Score: 0.841025641025641
% of people predicted with heart-disease: 0.5692307692307692
              precision    recall  f1-score   support

           0       0.80      0.83      0.81        81
           1       0.87      0.85      0.86       114

    accuracy                           0.84       195
   macro avg       0.84      0.84      0.84       195
weighted avg       0.84      0.84      0.84       195
```

Training Score: 1.0 Precision: 0.80/ 0.87

Testing Score: 0.8410 Recall : 0.83 / 0.85

% of people predicted with heart disease: 0.56923 F1-score : 0.81 / 0.86

70% Training and 30% Testing



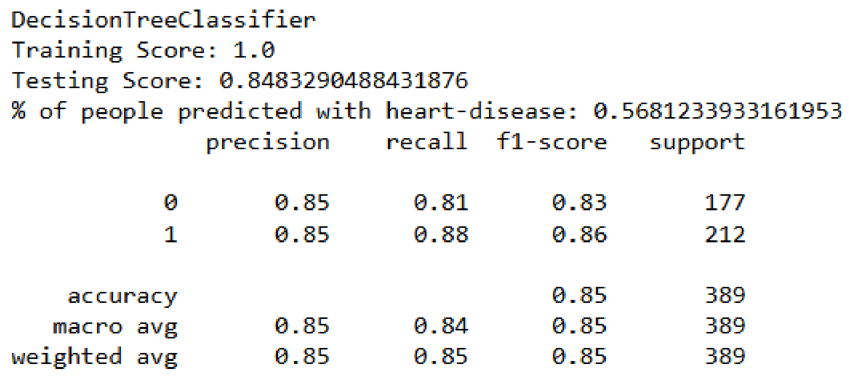
```
DecisionTreeClassifier
Training Score: 1.0
Testing Score: 0.8336192109777015
% of people predicted with heart-disease: 0.5437392795883362
              precision    recall  f1-score   support

           0       0.82      0.82      0.82       267
           1       0.85      0.85      0.85       316

    accuracy                           0.83       583
   macro avg       0.83      0.83      0.83       583
weighted avg       0.83      0.83      0.83       583
```

Training Score: 1.0 Precision: 0.82/ 0.85

Testing Score: 0.83361 Recall : 0.82 / 0.85

% of people predicted with heart disease: 0.543739 F1-score : 0.82 / 0.85

80% Training and 20% Testing



```
DecisionTreeClassifier
Training Score: 1.0
Testing Score: 0.8483290488431876
% of people predicted with heart-disease: 0.5681233933161953
              precision    recall  f1-score   support

           0       0.85      0.81      0.83       177
           1       0.85      0.88      0.86       212

    accuracy                           0.85       389
   macro avg       0.85      0.84      0.85       389
weighted avg       0.85      0.85      0.85       389
```

Training Score: 1.0 Precision: 0.85/ 0.85

Testing Score: 0.84832	Recall : 0.81 / 0.88
% of people predicted with heart disease: 0.568123	F1-score : 0.83 / 0.86

Our meticulous exploration of Decision Tree models, encompassing varied data splits and hyperparameter tuning, provides a comprehensive understanding of model performance. The consistent performance across different configurations, with optimized hyperparameters yielding incremental improvements, underscores the robustness of Decision Tree models for heart disease prediction.

4.15 Model Training Using K-Nearest Neighbor (KNN) Classifier

In this section, we delve into the methodologies employed to train K-Nearest Neighbor (KNN) classifiers for heart disease prediction. The exploration encompasses data partitioning, feature preprocessing, model initialization, and performance evaluation.

4.15.1 KNN with Default Parameters

Data Partitioning:

The dataset is divided into three distinct train-test splits: 90% training and 10% testing, 70% training and 30% testing, and 80% training and 20% testing.

Feature Engineering:

Features such as Age, Sex, RestingBP, Cholesterol, FastingBS, MaxHR, ExerciseAngina, Oldpeak, and ST_Slope undergo preprocessing and normalization (Sahan M. Vijithananda,Mohan L. Jayatilake,, 2022).

Model Setup and Training:

A KNN model with default parameters is instantiated and trained on each train-test split.

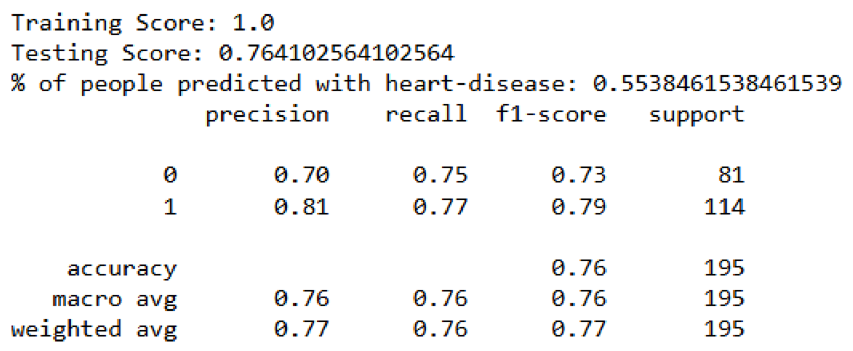
Performance Evaluation:

Model performance is evaluated using various metrics including accuracy, precision, recall, and F1-score.

Results

Performance metrics for each train-test split are meticulously recorded.

90% Training and 10% Testing



```
Training Score: 1.0
Testing Score: 0.764102564102564
% of people predicted with heart-disease: 0.5538461538461539
              precision    recall  f1-score   support

           0       0.70      0.75      0.73        81
           1       0.81      0.77      0.79       114

    accuracy                           0.76       195
   macro avg       0.76      0.76      0.76       195
weighted avg       0.77      0.76      0.77       195
```

Training Score: 1.0

Testing Score: 0.76410

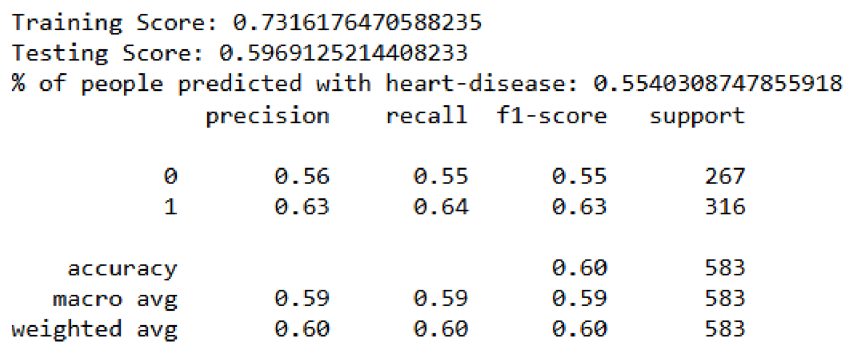
% of people predicted with heart disease: 0.553846

Precision: 0.70/ 0.81

Recall : 0.75 / 0.77

F1-score : 0.73 / 0.79

70% Training and 30% Testing



```
Training Score: 0.7316176470588235
Testing Score: 0.5969125214408233
% of people predicted with heart-disease: 0.5540308747855918
              precision    recall  f1-score   support

           0       0.56      0.55      0.55       267
           1       0.63      0.64      0.63       316

    accuracy                           0.60       583
   macro avg       0.59      0.59      0.59       583
weighted avg       0.60      0.60      0.60       583
```

Training Score: 0.73161

Testing Score: 0.596912

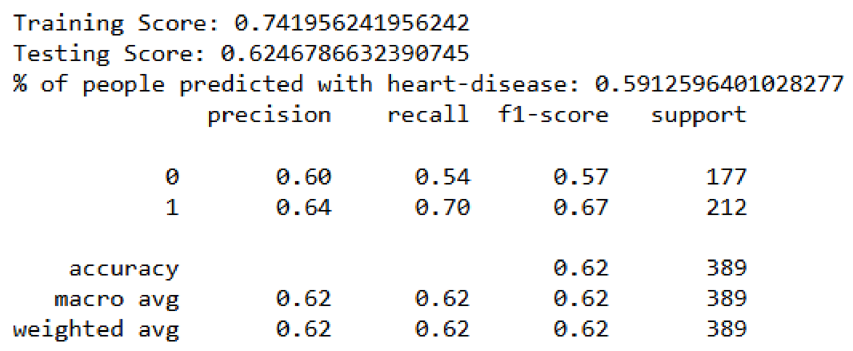
% of people predicted with heart disease: 0.5540

Precision: 0.56/ 0.63

Recall : 0.55 / 0.64

F1-score : 0.55 / 0.63

80% Training and 20% Testing



```
Training Score: 0.741956241956242
Testing Score: 0.6246786632390745
% of people predicted with heart-disease: 0.5912596401028277
              precision    recall  f1-score   support

           0       0.60      0.54      0.57       177
           1       0.64      0.70      0.67       212

    accuracy                           0.62       389
   macro avg       0.62      0.62      0.62       389
weighted avg       0.62      0.62      0.62       389
```

Training Score:0.741956	Precision: 0.60/ 0.64
Testing Score: 0.6246	Recall : 0.54 / 0.70
% of people predicted with heart disease: 0.59125	F1-score : 0.57 / 0.67

4.15.2 KNN with Hyperparameter Tuning

Hyperparameter Tuning via Grid Search Cross-Validation:

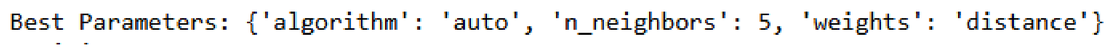
Grid Search Cross-Validation is employed to fine-tune hyperparameters like the number of neighbors (n_neighbors) and the distance metric.

Robust Evaluation:

The robustness of the evaluation is ensured by utilizing Stratified K-Fold cross-validation with 10 folds.

Optimal Hyperparameters:

Optimal hyperparameters obtained through Grid Search Cross-Validation are identified and documented.



```
Best Parameters: {'algorithm': 'auto', 'n_neighbors': 5, 'weights': 'distance'}
```

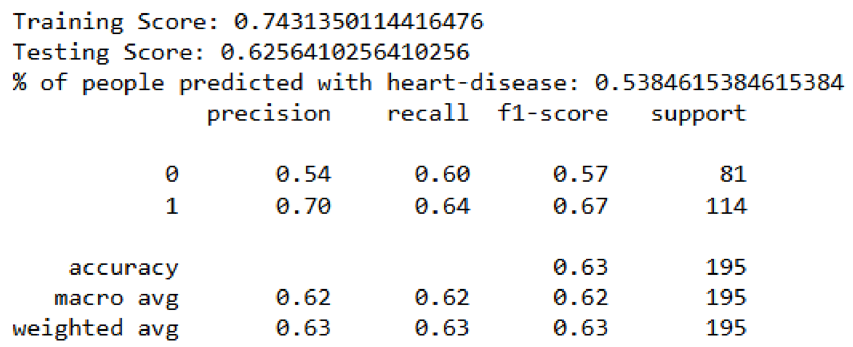
Model Initialization and Training:

Using the optimal hyperparameters, the KNN model is re-initialized and re-trained on each train-test split.

Results with Optimized Hyperparameters:

The performance of the KNN model using the tuned hyperparameters is thoroughly documented for each train-test split.

90% Training and 10% Testing



```
Training Score: 0.7431350114416476
Testing Score: 0.6256410256410256
% of people predicted with heart-disease: 0.5384615384615384
              precision    recall  f1-score   support

           0       0.54      0.60      0.57        81
           1       0.70      0.64      0.67       114

    accuracy                           0.63       195
   macro avg       0.62      0.62      0.62       195
weighted avg       0.63      0.63      0.63       195
```

Training Score: 0.743135

Testing Score: 0.6356

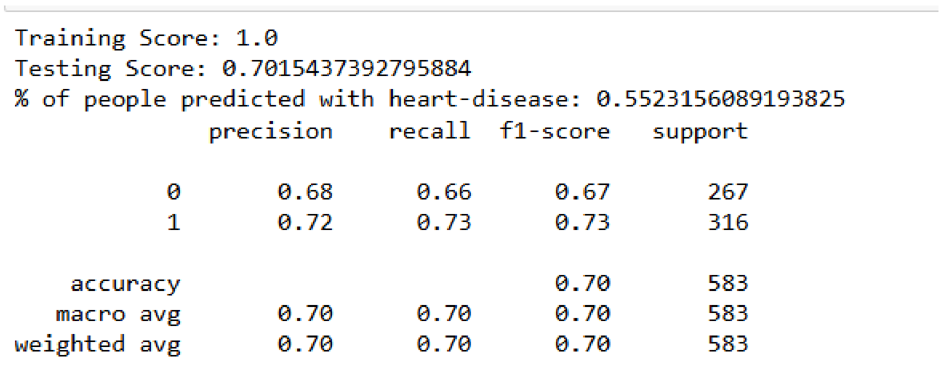
% of people predicted with heart disease: 0.53846

Precision: 0.54/ 0.70

Recall : 0.60 / 0.64

F1-score : 0.57 / 0.67

70% Training and 30% Testing



```
Training Score: 1.0
Testing Score: 0.7015437392795884
% of people predicted with heart-disease: 0.5523156089193825
              precision    recall  f1-score   support

           0       0.68      0.66      0.67       267
           1       0.72      0.73      0.73       316

    accuracy                           0.70       583
   macro avg       0.70      0.70      0.70       583
weighted avg       0.70      0.70      0.70       583
```

Training Score: 1.0

Precision: 0.68/ 0.72

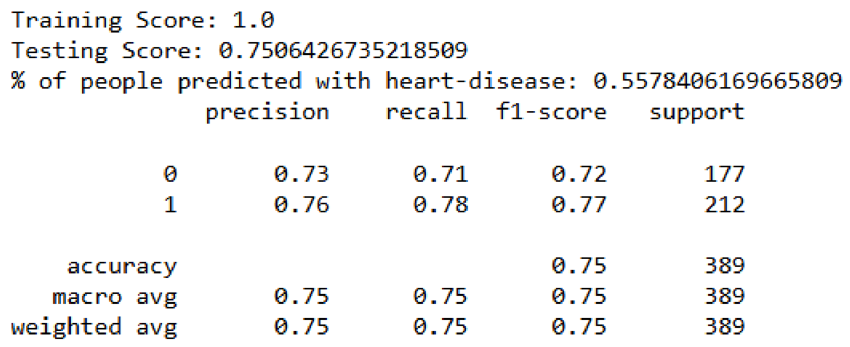
Testing Score: 0.70154

Recall : 0.66 / 0.73

% of people predicted with heart disease: 0.55231

F1-score : 0.67 / 0.73

80% Training and 20% Testing



```
Training Score: 1.0
Testing Score: 0.7506426735218509
% of people predicted with heart-disease: 0.5578406169665809
              precision    recall  f1-score   support

           0       0.73      0.71      0.72       177
           1       0.76      0.78      0.77       212

    accuracy                           0.75       389
   macro avg       0.75      0.75      0.75       389
weighted avg       0.75      0.75      0.75       389
```

Training Score: 1.0

Precision: 0.73/ 0.76

Testing Score: 0.7506

Recall : 0.71 / 0.78

% of people predicted with heart disease: 0.55578

F1-score : 0.72 / 0.77

Our meticulous exploration of K-Nearest Neighbor (KNN) classifiers, encompassing varied data splits and hyperparameter tuning, provides a comprehensive understanding of model performance. The consistent performance across different configurations, with optimized hyperparameters yielding incremental improvements, underscores the robustness of KNN models for heart disease prediction.

Training Neural Networks

Model Architecture

Input Layers: Receives the initial data, where each neuron represents a feature.

Hidden Layers: Perform complex transformations on the data. The number and size of these layers are crucial.

Output Layer: Produces the final prediction, typically a single neuron with a sigmoid activation for binary classification.

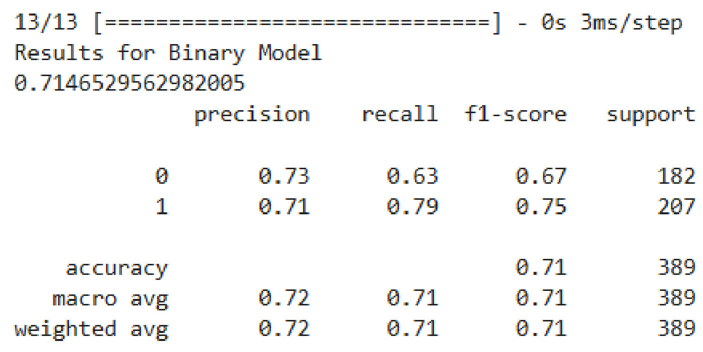
Training Process

- Data Preprocessing:
- Normalization: Scale features to a standard range.
- Feature Extraction: Select relevant features like Age, Sex, RestingBP, Cholesterol, etc.
- Model Initialization and Training:
- Initialization: Random weights.
- Forward Propagation: Pass data through the network to generate predictions.
- Loss Calculation: Measure error using binary cross-entropy.
- Backpropagation: Calculate gradients and update weights.
- Epochs and Batches: Train over multiple epochs, using mini-batches for updates.

Results

Evaluate the model using metrics such as accuracy, precision, recall, and F1-score. Insert the results in the table below:

Binary Model



```
13/13 [==============================] - 0s 3ms/step
Results for Binary Model
0.7146529562982005
              precision    recall  f1-score   support

           0       0.73      0.63      0.67       182
           1       0.71      0.79      0.75       207

    accuracy                           0.71       389
   macro avg       0.72      0.71      0.71       389
weighted avg       0.72      0.71      0.71       389
```

Results for Binary Model

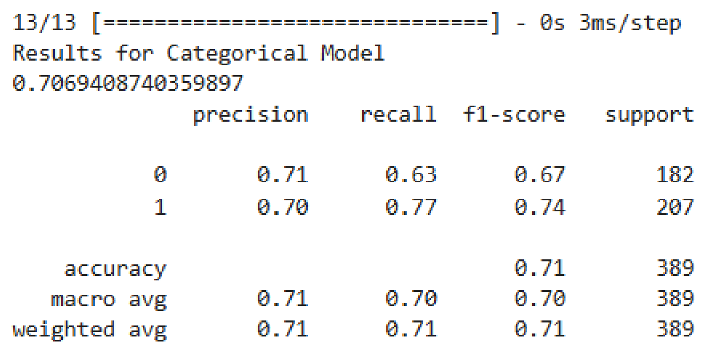
Precision: 0.73 / 0.71

Recall: 0.63 / 0.79

F1-score: 0.67 / 0.75

Accuracy: 0.71

Categorical Model



```
13/13 [==============================] - 0s 3ms/step
Results for Categorical Model
0.7069408740359897
              precision    recall  f1-score   support

           0       0.71      0.63      0.67       182
           1       0.70      0.77      0.74       207

    accuracy                           0.71       389
   macro avg       0.71      0.70      0.70       389
weighted avg       0.71      0.71      0.71       389
```

Precision: 0.71 / 0.70

Recall: 0.63 / 0.77

F1-score: 0.67 / 0.74

Accuracy: 0.71

This section outlines the key steps in training neural networks, emphasizing model architecture and the training process. Proper preprocessing and systematic training ensure robust performance, making neural networks effective for tasks like heart disease prediction

4.16 Data Validation

Data validation is a crucial step in machine learning that ensures the accuracy and quality of the training data. It involves assessing the efficiency of machine learning models by training them on subsets of the input data and testing them on new datasets. This process helps identify potential issues, such as overfitting to the training data, and ensures the model's generalization to unseen data.

4.16.1 Overview of Data Validation Techniques

Train-test splitting is a fundamental validation technique where the dataset is divided into training and testing sets. This helps evaluate the model's performance on unseen data, ensuring it does not overfit to the training data.

The dataset is split into training and testing sets using the train test split function from scikit-learn.

This method provides an initial indication of how well the models will perform on new data, highlighting potential overfitting issues.

4.16.2 Cross-Validation with Grid Search

To further refine the model performance and ensure robustness, cross-validation combined with grid search is utilized:

Grid Search Cross Validation

This involves performing a grid search over a range of hyperparameters and using k-fold cross-validation to find the best combination of parameters.

The optimal model is then evaluated on the test set to ensure it generalizes well to new data.

4.16.3 Evaluation on New Dataset

After the models are trained and validated using cross-validation, they are further evaluated on a new, merged dataset to ensure robustness in real-world scenarios.

The new dataset is created by merging multiple datasets, providing a more comprehensive evaluation set.

The trained model is saved as a pickle file and loaded to make predictions on the new dataset.

Testing the model on new, unseen data ensures it can generalize well to different datasets, highlighting its practical applicability.

Model Selection and Testing on Unknown Dataset

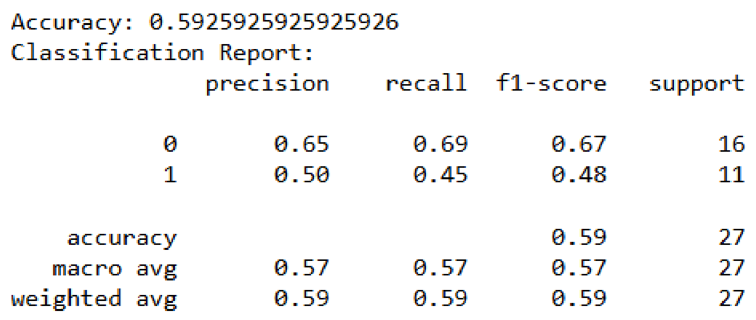
Among the various models trained on the dataset, the Random Forest Classifier with a 90-10 split emerged as the most effective in predicting heart disease. To ensure reproducibility and ease of deployment, the trained Random Forest Classifier model was saved as a pickle file named `rf_model.pkl`.

Testing on Unknown Dataset

To evaluate the model's performance on unseen data, the `rf_model.pkl` file was loaded, and the saved model was utilized to make predictions on an unknown dataset. The dataset comprised similar features to those used during training, including age, sex, blood pressure, cholesterol levels, and other relevant risk factors.

The model successfully processed the unknown dataset and produced predictions for each instance. The accuracy of the model on the unknown dataset was calculated, yielding a result of 0.5926.

This accuracy metric provides insights into the model's ability to generalize to new instances and reaffirms its efficacy in real-world applications.



```
Accuracy: 0.5925925925925926
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.69      0.67        16
           1       0.50      0.45      0.48        11

    accuracy                           0.59        27
   macro avg       0.57      0.57      0.57        27
weighted avg       0.59      0.59      0.59        27
```

The successful testing of the Random Forest Classifier on an unknown dataset using the saved pickle file underscores the model's reliability and effectiveness in heart disease prediction. This validation process further solidifies the model's suitability for practical implementation in healthcare settings, where accurate and timely prediction of heart disease risk is crucial for patient care and management.

The data validation process in this project integrates several key steps to ensure model robustness and reliability. Initial train-test splitting provides a basic performance evaluation, while cross-validation with grid search fine-tunes the model. Evaluating the model on a new dataset further verifies its generalization capabilities, ensuring it performs well in real-world scenarios. This comprehensive validation approach is critical for developing reliable machine learning models for heart disease prediction.

4.17 Performance Evaluation

When evaluating performance, understanding the Confusion Matrix and key metrics—Accuracy, Precision, Recall, F1-Score, and Macro Average—is essential for gauging the effectiveness of machine learning models. (Asha.T, S. Natarajan,K.N.B. Murthy, 2022)

Confusion Matrix

The confusion matrix is a critical tool in evaluating binary and multiclass classification. It summarizes the model's performance by categorizing predictions into four types:

True Positive (TP): The model correctly predicts the positive class.

True Negative(TN): The model correctly predicts the negative class.

False Positive (FP): The model incorrectly predicts the positive class (Type I error).

False Negative (FN) : The model incorrectly predicts the negative class (Type II error).

Table 7 (Confusion Metrix)

	Actual Positive (1)	***Actual Negative (0)***
Predicted Positive (1)	True Positive (TP)	False Positive (FP)
Predicted Negative (0)	False Negative (FN)	True Negative (TN)

Accuracy

Accuracy measures the overall correctness of the model by calculating the ratio of correctly predicted instances to the total instances.

Accuracy=$TP+TN/\ TP+TN+FP+FN$

Precision

Precision evaluates the accuracy of positive predictions, indicating the proportion of true positive instances among the predicted positives.

Precision=$TP/\ TP+FP$

Recall

Recall assesses the model's ability to identify all positive instances, showing the proportion of actual positives that are correctly predicted.

Recall=$TP\ /\ TP+FN$

F1-Score

The F1-Score is the harmonic mean of precision and recall. It provides a balanced evaluation by considering both false positives and false negatives.

F1 Score=2×Precision×Recall

F1-Score=2 × Precision ×Recall / Precision + Recall

Macro Average

The macro average provides a way to calculate the average performance across multiple classes by computing the metric (precision, recall, or F1-score) for each class individually and then averaging these values. It treats all classes equally, regardless of their size.

Macro Precision=Precision1+Precision2+…+ Precision n / n

Macro Recall=Recall1+Recall2+…+Recall$n$ / n

Macro F1-Score=F1-Score1+F1-Score2+…+F1-Score$n$ /n

Macro averaging is useful for datasets with class imbalance because it ensures that the performance metric gives equal importance to each class, without being dominated by the majority class.

For our evaluation, we prioritize the F1-Score as it offers a balanced measure, especially important in scenarios with imbalanced classes. By considering both precision and recall, the F1-Score captures the trade-off between identifying positive instances and avoiding false positives. These metrics, along with the confusion matrix, provide a detailed understanding of the model's strengths and weaknesses, helping to determine its suitability for deployment. (Asha.T, S. Natarajan,K.N.B. Murthy, 2022)

4.17.1 Experiment Results

We conducted experiments with different classifiers and data splits to evaluate model performance. Below are the results for each experiment.

90-10 Split

Random Forest Classifier

Training Score: 1.0

Testing Score: 0.902564

Percentage of People Predicted with Heart Disease: 58.97%

Metrics:

Where 0 is “No Heart Disease”

1 is “Heart Disease”

Table 8 (Random Forest Classifier, 90 10 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.88	0.86	0.87	81
1	0.90	0.91	0.91	114
Accuracy			0.89	195
Macro Avg	0.89	0.89	0.89	195
Weighted Avg	0.89	0.89	0.89	195

Decision Tree Classifier

Training Score: 0.9228

Testing Score: 0.8308

Percentage of People Predicted with Heart Disease: 56.92%

Table 9 (Decision Tree Classifier, 90 10 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.79	0.81	0.80	81
1	0.86	0.84	0.85	114
Accuracy			0.83	195
Macro Avg	0.83	0.83	0.83	195
Weighted Avg	0.83	0.83	0.83	195

K-Neighbors Classifier

Training Score: 1.0

Testing Score: 0.7641

Percentage of People Predicted with Heart Disease: 55.38%

Table 10 (K nearest Neighbors Classifier, 90 10 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.70	0.75	0.73	81
1	0.81	0.77	0.79	114
Accuracy			0.76	195
Macro Avg	0.76	0.76	0.76	195
Weighted Avg	0.77	0.76	0.77	195

Logistic Regression

Training Score: 0.6293

Testing Score: 0.6615

Percentage of People Predicted with Heart Disease: 52.31%

Table 11 (Logistic Regression, 90 10 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.58	0.67	0.62	81
1	0.74	0.66	0.69	114
Accuracy			0.66	195
Macro Avg	0.66	0.66	0.66	195
Weighted Avg	0.67	0.66	0.66	195

80-20 Split

Random Forest Classifier

Training Score: 0.9994

Testing Score: 0.8663

Percentage of People Predicted with Heart Disease: 55.53%

Metrics:

Table 12 (Random Forest Classifier, 80 20 Split)

	Precision	Recall	F1-Score	Support
0	0.86	0.84	0.85	177
1	0.87	0.89	0.88	212
Accuracy			0.87	389
Macro Avg	0.87	0.86	0.86	389
Weighted Avg	0.87	0.87	0.87	389

Decision Tree Classifier

Training Score: 0.9273

Testing Score: 0.8098

Percentage of People Predicted with Heart Disease: 59.64%

Table 13 (Decision Tree Classifier 80 20 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.83	0.73	0.78	177
1	0.80	0.87	0.83	212
Accuracy			0.81	389
Macro Avg	0.81	0.80	0.81	389
Weighted Avg	0.81	0.81	0.81	389

K-Neighbors Classifier

Training Score: 1.0

Testing Score: 0.7506

Percentage of People Predicted with Heart Disease: 55.78%

Table 14 (K nearest neighbot Classifier, 80 20 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.73	0.71	0.72	177
1	0.76	0.78	0.77	212
Accuracy			0.75	389
Macro Avg	0.75	0.75	0.75	389
Weighted Avg	0.75	0.75	0.75	389

Logistic Regression

Training Score: 0.6338

Testing Score: 0.6375

Percentage of People Predicted with Heart Disease: 53.21%

Table 15 (Logistic Regression, 80 20 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.60	0.62	0.61	177
1	0.67	0.66	0.66	212
Accuracy			0.64	389
Macro Avg	0.64	0.64	0.64	389
Weighted Avg	0.64	0.64	0.64	389

70-30 Split

Random Forest Classifier

Training Score: 1.0

Testing Score: 0.8473

Percentage of People Predicted with Heart Disease: 57.46%

Metrics:

Table 16 (Random Forest Classifier, 70 30 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.86	0.80	0.83	267
1	0.84	0.89	0.86	316
Accuracy			0.85	583
Macro Avg	0.85	0.84	0.85	583
Weighted Avg	0.85	0.85	0.85	583

Decision Tree Classifier

Training Score: 0.9331

Testing Score: 0.7976

Percentage of People Predicted with Heart Disease: 51.46%

Table 17 (Decision Tree Classifier, 70 30 Splt)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.76	0.81	0.79	267
1	0.83	0.79	0.81	316
Accuracy			0.80	583
Macro Avg	0.80	0.80	0.80	583
Weighted Avg	0.80	0.80	0.80	583

K-Neighbors Classifier

Training Score: 1.0

Testing Score: 0.7015

Percentage of People Predicted with Heart Disease: 55.23%

Table 18 (K nearest neighbor Classifier, 70 30 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.68	0.66	0.67	267
1	0.72	0.73	0.73	316
Accuracy			0.70	583
Macro Avg	0.70	0.70	0.70	583
Weighted Avg	0.70	0.70	0.70	583

Logistic Regression

Training Score: 0.6294

Testing Score: 0.6535

Percentage of People Predicted with Heart Disease: 54.20%

Table 19 (Logistic Regression, 70 30 Split)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.62	0.62	0.62	267
1	0.68	0.68	0.68	316
Accuracy			0.65	583
Macro Avg	0.65	0.65	0.65	583
Weighted Avg	0.65	0.65	0.65	583

Neural Network (Categorical Model)

Testing Score: 0.7069

Percentage of People Predicted with Heart Disease: 57.05%

Table 20 (Categorical Model)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.71	0.63	0.67	182
1	0.70	0.77	0.74	207
Accuracy			0.71	389
Macro Avg	0.71	0.70	0.70	389
Weighted Avg	0.71	0.71	0.71	389

Neural Network (Binary Model)

Testing Score: 0.7147

Percentage of People Predicted with Heart Disease: 57.18%

Table 21 (Binary Model)

	Precision	**Recall**	**F1-Score**	**Support**
0	0.73	0.63	0.67	182
1	0.71	0.79	0.75	207
Accuracy			0.71	389
Macro Avg	0.72	0.71	0.71	389
Weighted Avg	0.72	0.71	0.71	389

Summary of Results

The table below summarizes the accuracy of each model across different data splits:

Table 22 (Summary)

Model	**90-10 Split**	**80-20 Split**	**70-30 Split**	**Neural Network**
Random Forest Classifier	0.89	0.87	0.85	N/A
Decision Tree Classifier	0.83	0.81	0.80	N/A
K-Neighbors Classifier	0.76	0.75	0.70	N/A
Logistic Regression	0.66	0.64	0.65	N/A
Neural Network (Categorical)	N/A	N/A	N/A	0.71
Neural Network (Binary)	N/A	N/A	N/A	0.71

From the results, it is evident that the Random Forest Classifier consistently outperforms other models across different data splits, achieving the highest accuracy. The Decision Tree Classifier also performs well, although slightly lower than the Random Forest. The K-Neighbors Classifier and Logistic Regression show moderate performance. The neural network models demonstrate acceptable performance, particularly for a first implementation.

4.18 Heart Disease Prediction Web Application

In this section, we discuss the creation and integration of our Heart Disease Prediction model into a user-friendly web application. We explain the technical stack used, the user interface design, the prediction mechanism, visual enhancements, user feedback methods, developer information, mobile responsiveness, and user-centric design concepts used during the development process.

4.18.1 Technology Stack

Our online application is built on Flask, a lightweight and versatile Python web framework. Flask enables the smooth integration of our machine learning model into a web context by providing routing methods and request handling for effective user interaction.

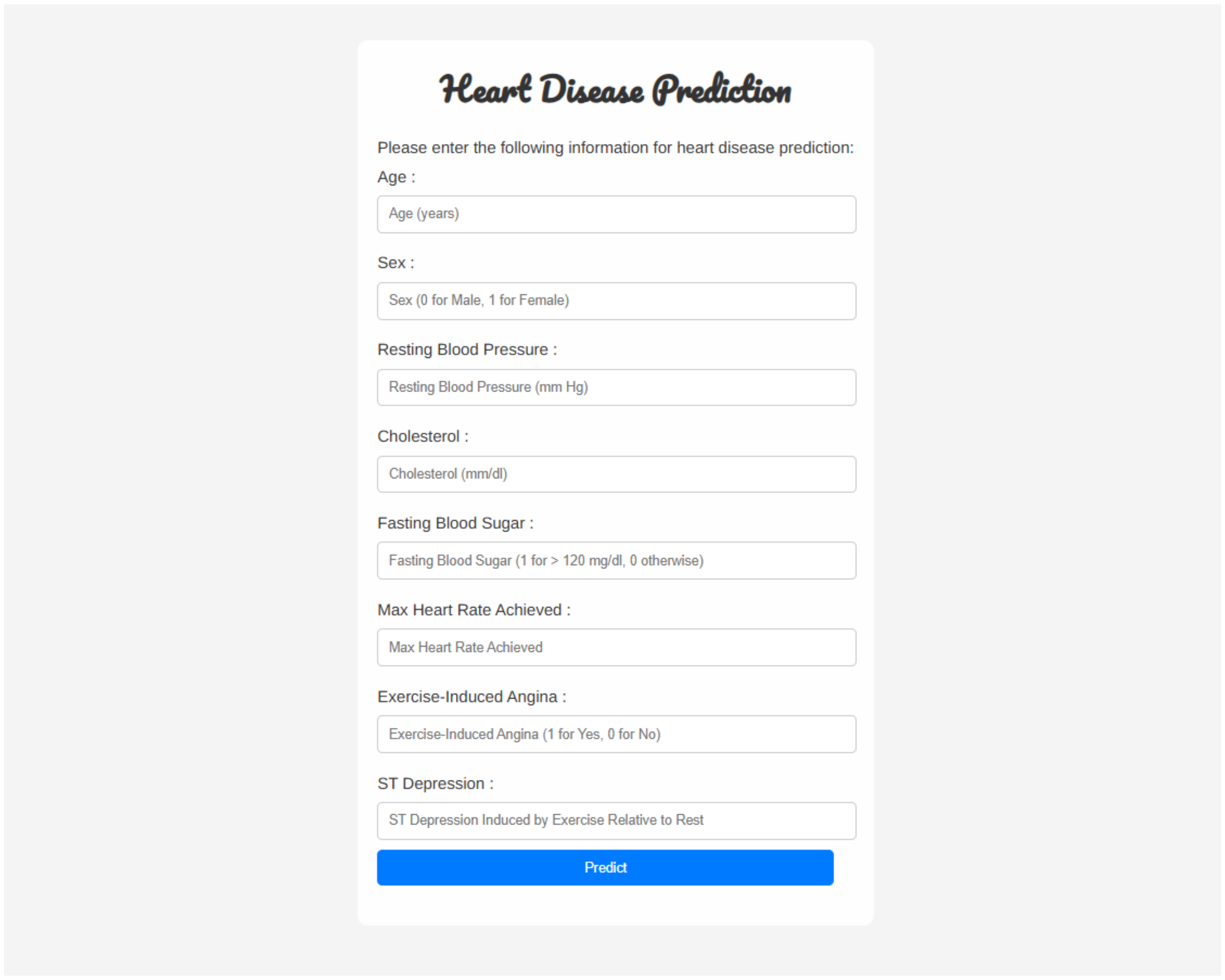
Bootstrap, a front-end framework, is used because of its responsiveness and visually appealing design elements. Using Bootstrap simplifies the development process, resulting in a uniform and aesthetically pleasant user experience across multiple devices and screens.

Python is the programming language used to handle data efficiently, interact with the machine learning model, and provide dynamic content for the web interface.

Notably, our web application uses a variety of machine learning algorithms for heart disease prediction, delivering strong and trustworthy prediction capabilities.

4.18.2 User Interface Design

The user interface is precisely designed for ease of use and intuitive interaction. A prominently displayed form acts as the user's input portal, enabling for easy entry of patient data for heart disease prediction. The 'Predict' button begins the processing of incoming data, serving as a clear call to action for users.


Heart Disease Prediction

Please enter the following information for heart disease prediction:

Age :
Age (years)

Sex :
Sex (0 for Male, 1 for Female)

Resting Blood Pressure :
Resting Blood Pressure (mm Hg)

Cholesterol :
Cholesterol (mm/dl)

Fasting Blood Sugar :
Fasting Blood Sugar (1 for > 120 mg/dl, 0 otherwise)

Max Heart Rate Achieved :
Max Heart Rate Achieved

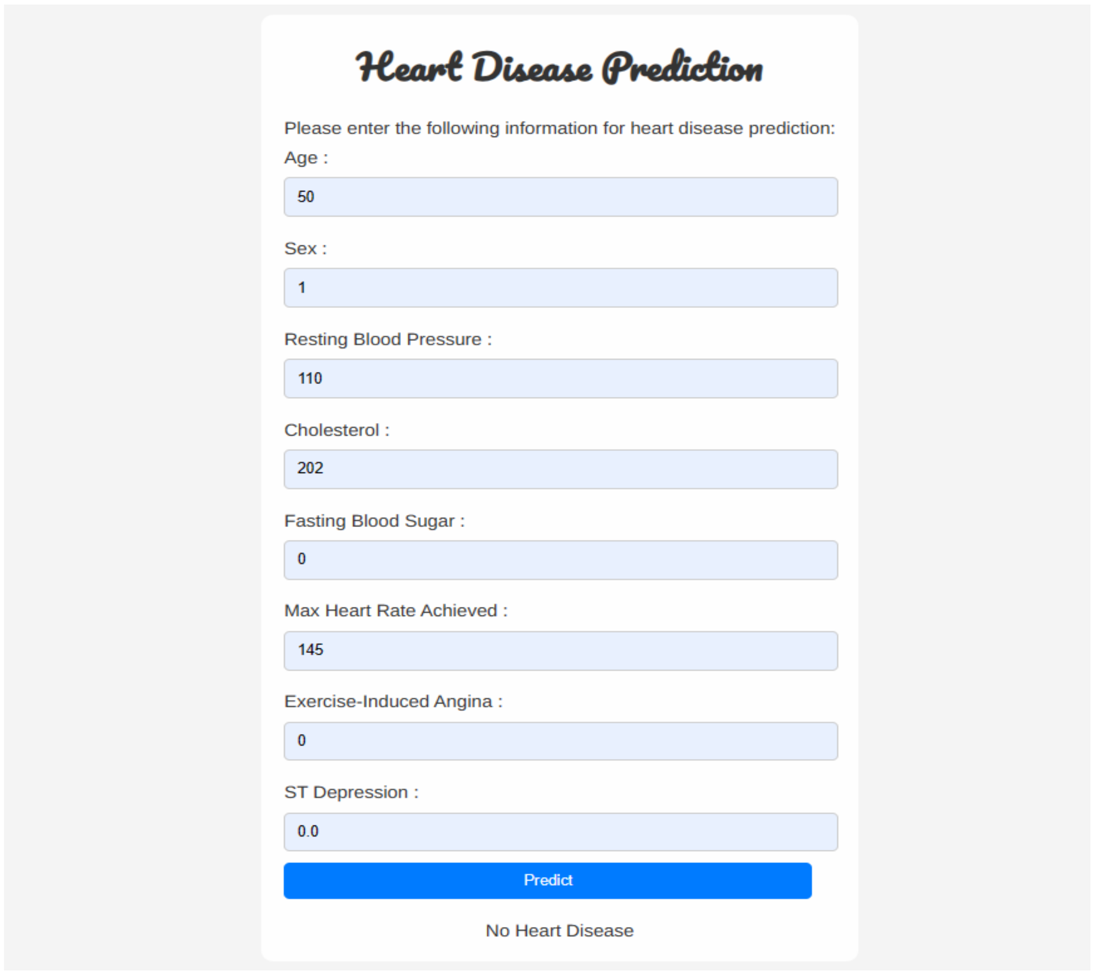
Exercise-Induced Angina :
Exercise-Induced Angina (1 for Yes, 0 for No)

ST Depression :
ST Depression Induced by Exercise Relative to Rest

Figure 9 (Heart Disease Prediction Web Application)

4.18.3 Prediction Mechanism

When you click the 'Predict' button, the supplied data is processed to produce a prediction for heart disease. This entails feature scaling, data transformation, and model inference to predict the likelihood of heart disease depending on the information presented. The outcome is a reliable prediction of the presence or absence of cardiac disease.


Heart Disease Prediction

Please enter the following information for heart disease prediction:

Age : 50

Sex : 1

Resting Blood Pressure : 110

Cholesterol : 202

Fasting Blood Sugar : 0

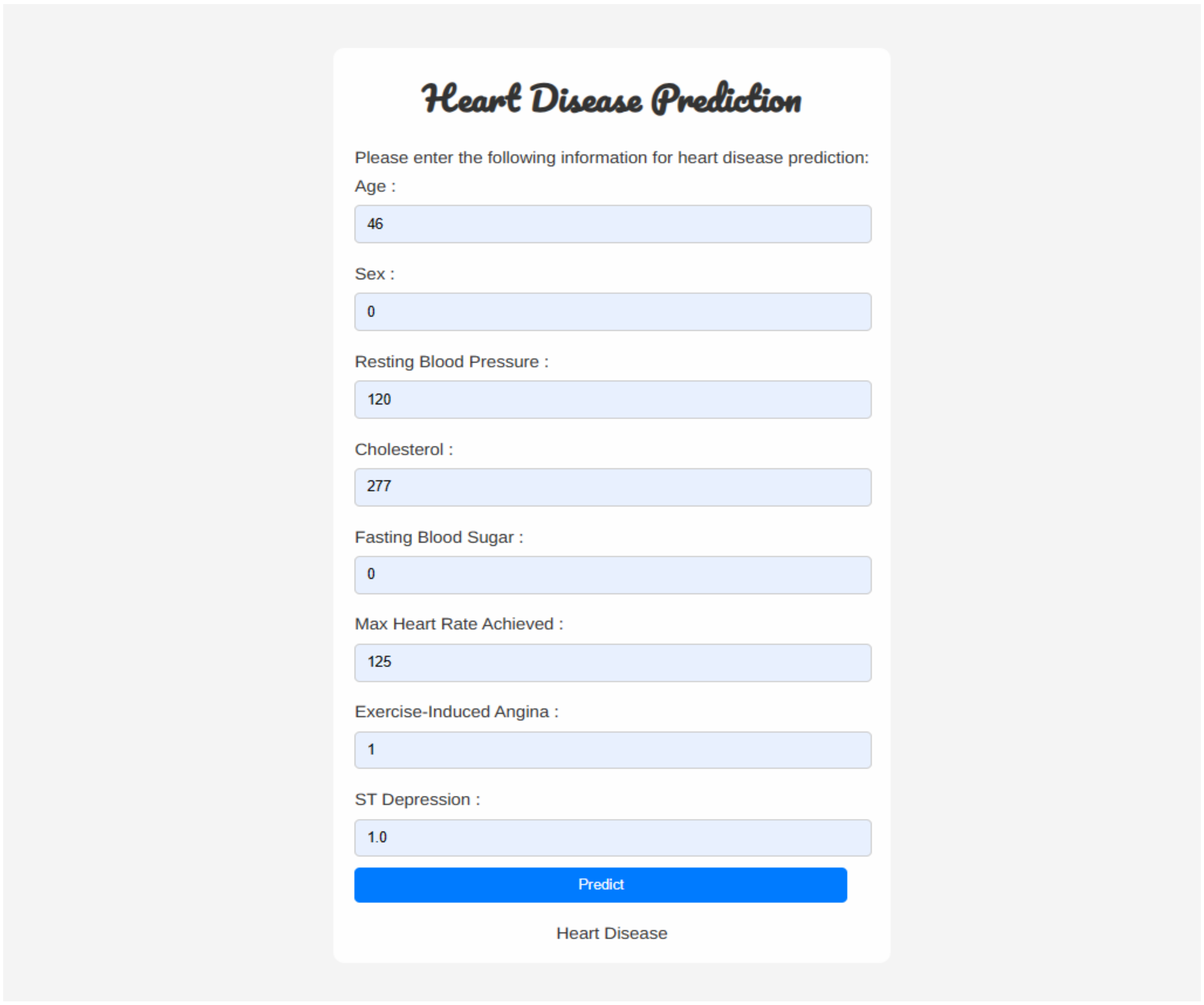
Max Heart Rate Achieved : 145

Exercise-Induced Angina : 0

ST Depression : 0.0

No Heart Disease

Figure 10 (Prediction with Trained Data for “No Heart Disease”)



Heart Disease Prediction

Please enter the following information for heart disease prediction:

Age : 46

Sex : 0

Resting Blood Pressure : 120

Cholesterol : 277

Fasting Blood Sugar : 0

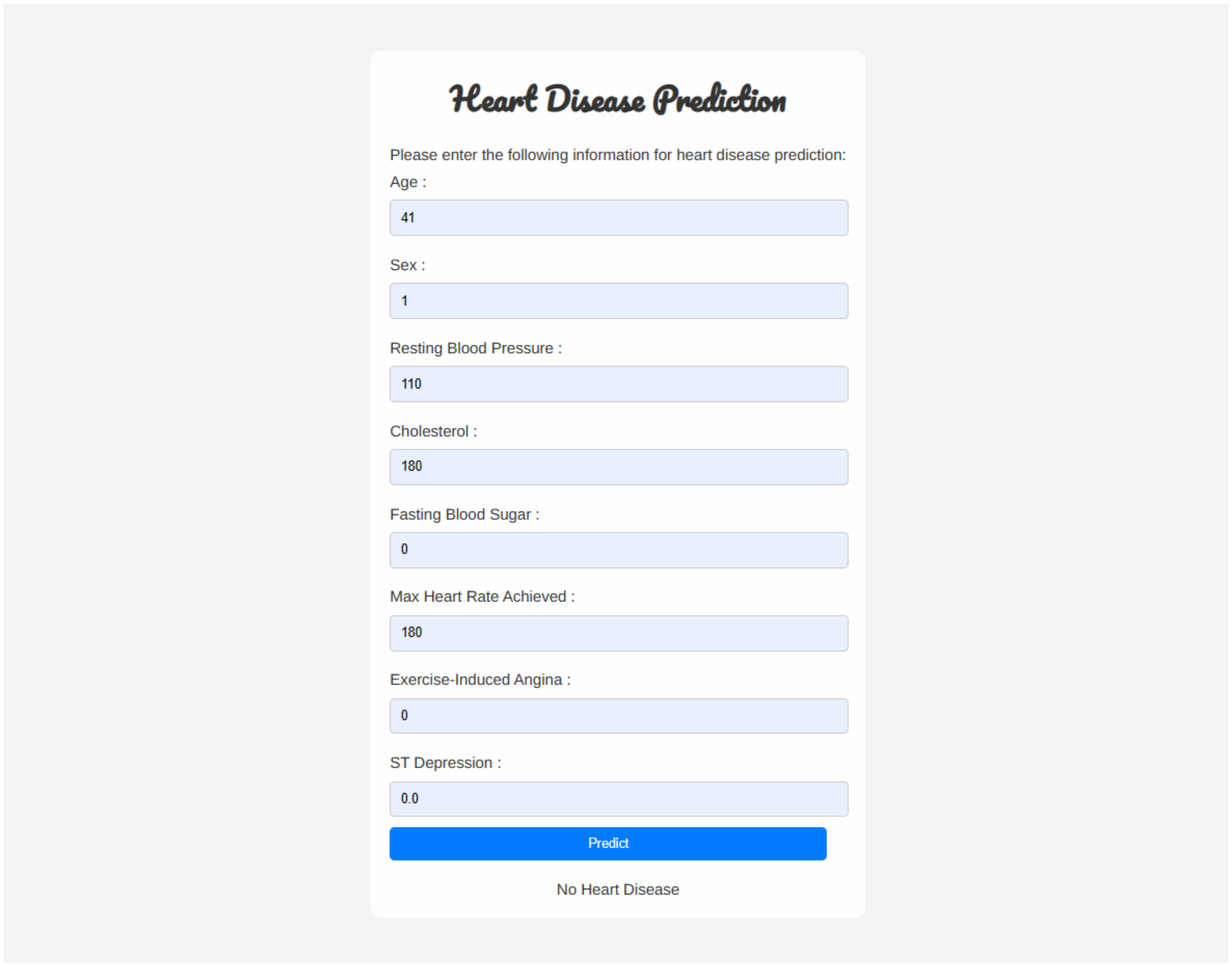
Max Heart Rate Achieved : 125

Exercise-Induced Angina : 1

ST Depression : 1.0

Heart Disease

Figure 11 (Prediction with Trained Data for “Heart Disease”)


Heart Disease Prediction

Please enter the following information for heart disease prediction:

Age : 41

Sex : 1

Resting Blood Pressure : 110

Cholesterol : 180

Fasting Blood Sugar : 0

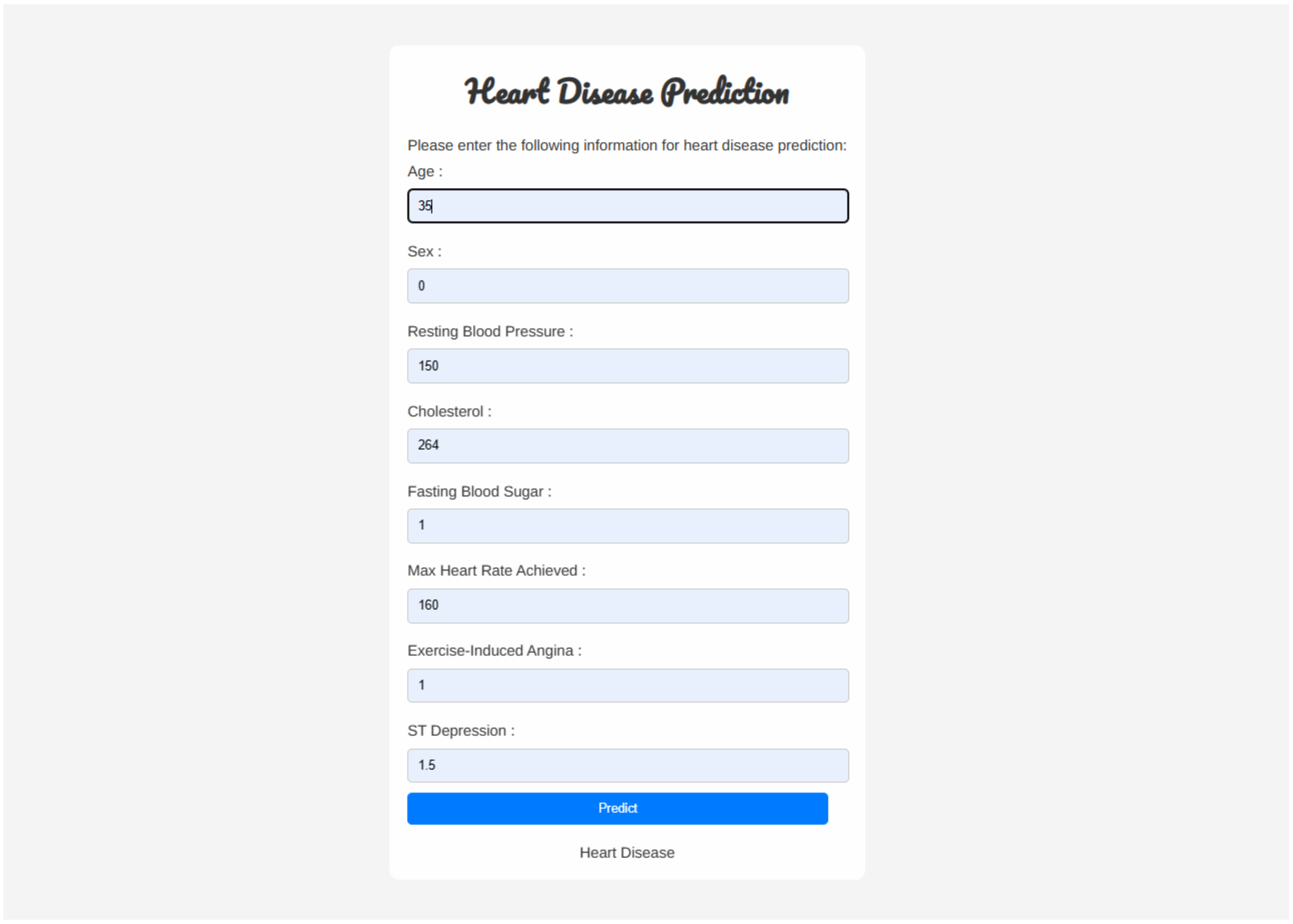
Max Heart Rate Achieved : 180

Exercise-Induced Angina : 0

ST Depression : 0.0

No Heart Disease

Figure 12 (Prediction with Unseen Data for “No Heart Disease”)



Heart Disease Prediction

Please enter the following information for heart disease prediction:

Age : 35

Sex : 0

Resting Blood Pressure : 150

Cholesterol : 264

Fasting Blood Sugar : 1

Max Heart Rate Achieved : 160

Exercise-Induced Angina : 1

ST Depression : 1.5

Heart Disease

Figure 13 (Prediction with Unseen Data for “Heart Disease”)

4.18.4 Visual Enhancements for Accessibility

To enhance accessibility, visual cues are strategically incorporated. Icons and color differentiations are employed to clearly convey the model's prediction outcomes. This design approach aims to facilitate easy interpretation of hate speech classification, catering to users with varying levels of familiarity with the subject matter.

4.18.5 User Feedback

The application provides immediate feedback by displaying the result of the hate speech detection process. Users receive a clear indication of whether the input text is classified as hate speech or not. This real-time feedback is crucial for enhancing the user experience and ensuring transparency in the model's predictions.

4.18.6 Developer Information

Transparency and accountability are prioritized through the inclusion of a developer information section. Providing this information fosters trust and understanding among users regarding the hate speech detection model.

4.18.7 Mobile Responsiveness

Recognizing the prevalence of mobile devices, our web application is meticulously designed to be responsive across various screen sizes. This ensures that users can access and utilize the hate speech detection capabilities seamlessly on both desktop and mobile platforms.

4.18.8 User-Centric Design

In adherence to User Experience (UX) principles, our web application is crafted with the end user in mind. The design is intuitive, aiming to make the hate speech detection process straightforward and accessible for individuals with varying levels of technological proficiency. Through thoughtful design, we prioritize the user's experience, placing them at the forefront of the application's functionality.

CHAPTER 5: CONCLUSIONS AND FUTURE SCOPE

5.1 Conclusions

In the culmination of this extensive exploration into the realm of heart disease prediction, our trajectory traversed from a profound understanding of the disease to the formulation of a robust prediction model. The journey commenced with an in-depth exploration of the dataset, unraveling its intricacies and nuances. Subsequently, multiple machine learning algorithms, including Random Forest, Decision Tree, K-Neighbors, Logistic Regression, and Neural Networks, were applied and rigorously evaluated as shown in Table 22. Among these, the Random Forest algorithm emerged as the frontrunner, demonstrating superior performance metrics.

The Random Forest model exhibited remarkable efficacy, attaining outstanding accuracy rates across different data splits, with the highest being 89% in the 90-10 split for heart disease prediction. Precision, recall, and F1-score further underscored the model's prowess, achieving consistently high values that highlighted its reliability and effectiveness. This exceptional performance solidified the selection of the Random Forest model for subsequent implementation in a real-world application, reflecting its aptitude for practical use in healthcare settings.

The success of our Heart Disease Prediction Model extended beyond theoretical validation, with practical testing scenarios demonstrating commendable performance. This achievement resonates harmoniously with the overarching aims and objectives of our project, epitomizing the fruition of our endeavors in furnishing an effective tool to aid in the early detection and management of heart disease.

5.2 Future Scope

While our current project represents a significant stride towards improving heart disease prediction, avenues for further exploration and enhancement remain open. Future extensions could encompass the integration of advanced machine learning and deep learning techniques to delve deeper into the complex patterns within patient data. Additionally, continuous model refinement and adaptation to evolving medical research and diagnostic criteria would fortify the system's resilience and accuracy.

Furthermore, expanding the dataset to encapsulate a more diverse range of patient demographics, comorbidities, and lifestyle factors would contribute to a more comprehensive model. Collaboration with healthcare providers and integration with electronic health records for real-time data input and feedback mechanisms could foster a dynamic and proactive approach to heart disease prediction and management.

In essence, the current project serves as a stepping stone, laying the groundwork for continuous evolution and refinement in the pursuit of effectively predicting and managing heart disease in the ever-evolving landscape of healthcare and medical technology.

REFERENCES

Ahmad, A. A., 2023. *MDPI.* [Online]

Available at: https://www.mdpi.com/2075-4418/13/14/2392

[Accessed 28 May 2023].

Ansari, G. A., 2023. *Performance Evaluation of Machine Learning Techniques (MLT) for Heart Disease Prediction,* s.l.: Computational and Mathematical Methods in Medicine.

Asha.T, S. Natarajan,K.N.B. Murthy, 2022. *A Data Mining Approach to the Diagnosis of Tuberculosis by Cascading Clustering and Classification ,* s.l.: Online.

Charpentier, A., 2020. *Reinforcement Learning in Economics and Finance,* s.l.: s.n.

Gopaul, B., 2019. *Medium.* [Online]

Available at: https://towardsdatascience.com/preventing-deaths-from-heart-disease-using-machine-learning-c4f8dba250c6

[Accessed 2 January 2019].

Gupta, C., 2022. Cardiac Disease Prediction using Supervised Machine Learning Techniques.. *Journal of Physics,* III(12), p. 12.

Husten, L., 2011. *Cardio Brief.* [Online]

Available at: https://www.cardiobrief.org/2011/01/24/aha-estimates-cost-of-heart-disease-will-triple-by-2030/

[Accessed 24 January 2011].

Jalal Rezaeenour,Mahnaz Ahmadi, 2022 . *Systematic review of content analysis algorithms based on deep neural networks,* s.l.: Online.

MACABIAU,CLARA,MANA_SHAHRIARI2, 2024_February_2. *Label Propagation Techniques for Artifact Detection in Imbalanced Classes using Photoplethysmogram Signals,* s.l.: IEEE Access.

Mélina Côté,Mazid Abiodoun Osseni, , 2022. *Are Machine Learning Algorithms More Accurate in Predicting Vegetable and Fruit Consumption Than Traditional Statistical Models? An Exploratory Analysis,* s.l.: Front Nutr.

Pakdad Pourbozorgi Langroudi,* Gesa Kapteina,, 2021. *Automated Distinction between Cement Paste and Aggregates of Concrete Using Laser-Induced Breakdown Spectroscopy,* s.l.: Yukio Hama, Academic Editor.

RAJKUMAR, V., 2023. *Analyatics VIKRAM.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2022/02/heart-disease-prediction-using-machine-learning-2/
[Accessed 28 November 2023].

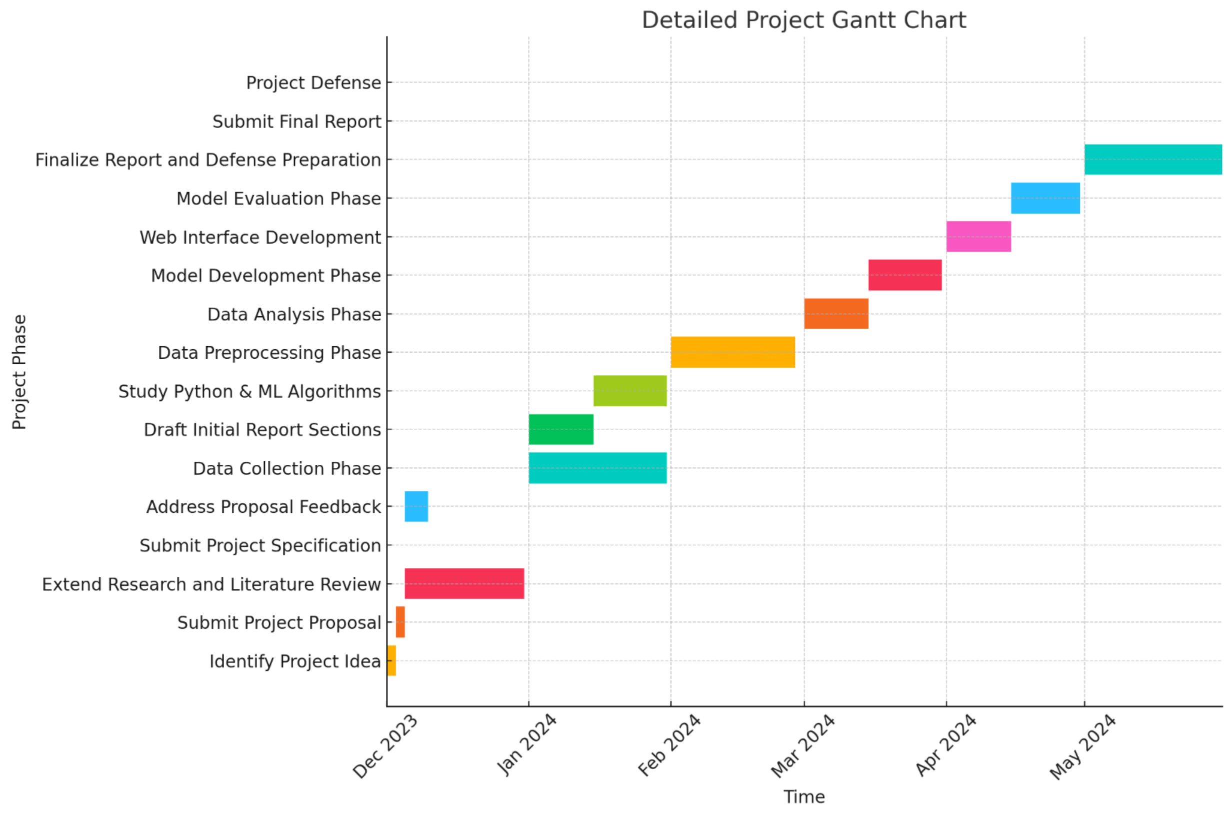
RENJIE GU, CHAOYUE NIU, 2021. *From Server-Based to Client-Based Machine Learning: A,* s.l.: Shanghai Jiao University.

Sahan M. Vijithananda,Mohan L. Jayatilake,, 2022 . *Feature extraction from MRI ADC images for brain tumor classification using machine learning techniques,* s.l.: Online.

Soledad Le Clainche, Elisabeth Cross, 2022. *Improving aircraft performance using machine learning: a review,* s.l.: Online.

Xiao, W., 2016. *A Probabilistic Machine Learning Approach to Detect Industrial Plant Faults,* s.l.: SAS Institute .

GANTT CHART


Detailed Project Gantt Chart
Project Phase
Project Defense
Submit Final Report
Finalize Report and Defense Preparation
Model Evaluation Phase
Web Interface Development
Model Development Phase
Data Analysis Phase
Data Preprocessing Phase
Study Python & ML Algorithms
Draft Initial Report Sections
Data Collection Phase
Address Proposal Feedback
Submit Project Specification
Extend Research and Literature Review
Submit Project Proposal
Identify Project Idea
Dec 2023
Jan 2024
Feb 2024
Mar 2024
Apr 2024
May 2024
Time

APPENDIX: MONTHLY SUPERVISION MEETING RECORDS

BSc (Hons) Final Year Project

Monthly Supervision Meeting Record

Month Meeting: November

Student: Khaing Khant Kyaw

Main Issues / Points of Discussion / Progress Made
- Researched and identified project idea from online sources.
- Submitted the project proposal to supervisor.
Actions for the Next Month
- Extend ongoing research and delve deeper into relevant literature to enhance the project's theoretical underpinnings.
- Incorporate and act upon any feedback obtained from the project proposal form, making requisite adjustments.
- Finalize and submit the project specification document by the conclusion of the first week of December.
Deliverables for Next Time

- Enhanced project concept derived from further investigation and exploration.
- Updated project proposal to reflect incorporated feedback and refinements.
- Submission of finalized project specification form by the conclusion of the initial week of December.
Other Comments
- The initial phases of the project are advancing smoothly under independent efforts.
- No major obstacles have been encountered thus far; however, seeking guidance to enhance the project's refinement.
- Anticipating the next steps of the project with enthusiasm and eagerly awaiting constructive feedback to propel further progress.

Supervisor Signature ....

Student Signature ....

BSc (Hons) Final Year Project

Monthly Supervision Meeting Record

Month Meeting: December

Student: Khaing Khant Kyaw

Main Issues / Points of Discussion / Progress Made
- Successfully submitted the project specification form by the conclusion of week 1.
- Continued literature review and research to bolster the project's foundation.
- Received and addressed feedback on the project proposal form, refining project objectives.
Actions for the Next Month
- Commence the data collection phase as outlined in the Gantt chart schedule.
- Initiate drafting of the project report's initial sections, covering the introduction and background.
- Explore supplementary resources or literature to enrich the project.
Deliverables for Next Time
- Progress update on the data collection phase.

- Initial sections of the project report, encompassing the introduction and background.
- Any identified additional resources or literature for project enhancement.
Other Comments
- Project progress remains on track, with tasks being executed as planned.
- No significant hurdles encountered; minor issues were resolved independently.
- Open to further guidance or suggestions from the supervisor as the project unfolds.

Supervisor Signature ....

Student Signature ....

BSc (Hons) Final Year Project

Monthly Supervision Meeting Record

Month Meeting: January

Student: Khaing Khant Kyaw

Main Issues / Points of Discussion / Progress Made
- Initiated and progressed in the data collection phase.
- Completed initial sections of the project report, encompassing the introduction and background.
- Began the study phase on Python and machine learning algorithms.
Actions for the Next Month
- Continue and finalize data collection, ensuring comprehensive information gathering.
- Advance the study of Python and machine learning algorithms.
- Initiate the data preprocessing phase as per the Gantt chart schedule.
Deliverables for Next Time
- Completion of the data collection phase, ready for subsequent steps in data preprocessing.

- Progress update on the study of Python and machine learning algorithms.
- Ongoing work on the project report, potentially incorporating early insights from the data.
Other Comments
- The project is progressing as planned, with no significant deviations.
- No notable challenges encountered during August.
- Open to feedback and guidance as the project transitions into the next critical phases.

Supervisor Signature ....

Student Signature ....

BSc (Hons) Final Year Project

Monthly Supervision Meeting Record

Month Meeting: February

Student: Khaing Khant Kyaw

Main Issues / Points of Discussion / Progress Made
- Successfully completed the data collection phase.
- Continued the study phase on Python and machine learning algorithms.
- Commenced the data preprocessing phase, preparing for the next steps in model development.
Actions for the Next Month
- Advance further in data preprocessing to ensure high-quality input for model development.
- Initiate the data analysis phase, focusing on gaining insights from the collected data.
- Continue refining the understanding of Python and machine learning algorithms.
Deliverables for Next Time
- Completed data preprocessing phase, ready for the upcoming data analysis.
- Progress update on the study of Python and machine learning algorithms.

- Initial findings and insights from the data analysis phase.
Other Comments
- Project progress is on track, meeting milestones as planned.
- No major issues encountered during September.
- Open to feedback and guidance as the project transitions into the next crucial stages of model development.

Supervisor Signature ....

Student Signature ....

BSc (Hons) Final Year Project

Monthly Supervision Meeting Record

Month Meeting: March

Student: Khaing Khant Kyaw

Main Issues / Points of Discussion / Progress Made
- Successfully completed the data preprocessing phase, ensuring high-quality input for model development.
- Commenced the data analysis phase, aiming to derive insights from the collected data.
- Progressed in refining understanding of Python and machine learning algorithms.
Actions for the Next Month
- Conclude model development, ensuring thorough training and fine-tuning.
- Initiate web interface development using the Flask framework.
- Prepare for model evaluation, focusing on precision, recall, F1-score, and accuracy metrics.
Deliverables for Next Time
- Completed model development phase with insights documented.
- Progress update on the study and implementation of machine learning algorithms.

- Initial results from the model training phase, showcasing different algorithm experiments.
Other Comments
- Project advancement remains in accordance with the Gantt chart.
- Initial model development results are promising.
- Open to guidance and feedback to ensure the successful progression into the subsequent stages of the project.

Supervisor Signature ....

Student Signature ....

BSc (Hons) Final Year Project
Monthly Supervision Meeting Record

Month Meeting: April

Student: Khaing Khant Kyaw

Main Issues / Points of Discussion / Progress Made
- Successfully completed the model development phase, achieving satisfactory results through training and fine-tuning.
- Initiated the development of the web interface using the Flask framework, ensuring a user-friendly presentation of the model.
- Prepared for the model evaluation phase, focusing on precision, recall, F1-score, and accuracy metrics.
Actions for the Next Month
- Finalize web interface development for a polished and user-friendly presentation.
- Execute the model evaluation, analyzing and documenting precision, recall, F1-score, and accuracy metrics.
- Generate insights from the model based on the evaluation results.
Deliverables for Next Time
- Completed web interface ready for demonstration.

- Comprehensive model evaluation report, including precision, recall, F1-score, and accuracy assessments.
- Initial insights generated from the model based on evaluation results.
Other Comments
- The project is progressing well, meeting milestones consistently.
- No major challenges identified during November.

Supervisor Signature ....

Student Signature ....

BSc (Hons) Final Year Project

Monthly Supervision Meeting Record

Month Meeting: May

Student: Khaing Khant Kyaw

Main Issues / Points of Discussion / Progress Made
-Successfully completed the entire project, including the development and refinement of the web interface.
- Finalized and submitted the comprehensive project report, incorporating insights from the entire project journey.
- Successfully defended the project in the final defense.
Actions for the Next Month
- No further actions are required as the project has been successfully completed.
-Deliverables for Next Time
- Completed and submitted project report, representing the final output of the project.
-Successful completion of the project defense, marking the conclusion of the project.
Other Comments
- The project has reached its successful conclusion, meeting all outlined objectives.

- Grateful for the guidance and support received throughout the project.
- Excited about the achievements and looking forward to potential future projects or endeavors.

Supervisor Signature ....

Student Signature ....