HEART DISEASE PREDICTION WITH MACHINE LEARNING ALGORITHMS

By

SHRUTI GUPTA

Roll No: 25500120052

Registration No.: 202550100110048

SIMRAN SHARMA

Roll No: 25500120061

Registration No.: 202550100110039

SRISHTI BOSE

Roll No: 25500120065

Registration No.: 202550100110035

ANUBHAV MISHRA

Roll No: 25500120071

Registration No.: 202550100110029

Under the guidance of Mr. Anjan Kumar Payra

Bachelor of Technology (Computer Science & Engineering)

Department of Computer Science & Engineering

Dr. Sudhir Chandra Sur Institute of Technology and Sports Complex (JIS GROUP)

Maulana Abul Kalam Azad University of Technology, Kolkata, West Bengal, India.

Name of the Project: **HEART DISEASE PREDICTION WITH MACHINE LEARNING ALGORITHMS**

Project Submitted by:

Name	**Roll No.**	**Registration No.**
Shruti Gupta	25500120052	202550100110048 OF 2020-2021
Simran Sharma	25500120061	202550100110039 OF 2020-2021
Srishti Bose	25500120065	202550100110035 OF 2020-2021
Anubhav Mishra	25500120071	202550100110029 OF 2020-2021

Under the guidance of: **Mr. Anjan Kumar Payra**

Mr. Anjan Kumar Payra
(Asst. Prof., CSE, DSCSITSC)

SurTech

CERTIFICATE

This is to certify that this is a bona fide record of the FINAL YEAR project work "**HEART DISEASE PREDICTION WITH MACHINE LEARNING ALGORITHMS**" done satisfactorily at DR. SUDHIR CHANDRA SUR INSTITUTE OF TECHNOLOGY AND SPORTS COMPLEX, formerly known as DR. SUDHIR CHANDRA DEGREE ENGINEERING COLLEGE, **by Shruti Gupta, Simran Sharma, Srishti Bose, and Anubhav Mishra** of **7th Semester CSE**.

This report or a similar report on this topic has not been submitted for any other examination and is not part of any other course undergone by the candidate. I have no doubt that they have excellent research potential.

I would like to extend my heartfelt wishes for a future filled with brightness and success.

Date: ________________________________   ____

Mr. Anjan Kumar Payra
(Asst. Prof., CSE, DSCSITSC)

____

Dr. Himadri Biswas
(HOD, CSE, DSCSITSC)

ACKNOWLEDGEMENT

We would like to take this opportunity to express our gratitude towards all the people who have, in various ways, helped us in the successful completion of our final year project on "HEART DISEASE PREDICTION WITH MACHINE LEARNING ALGORITHMS," done satisfactorily at DR. SUDHIR CHANDRA SUR INSTITUTE OF TECHNOLOGY AND SPORTS COMPLEX. We must convey our gratitude to Mr. Anjan Kumar Payra, Assistant Professor of Computer Science and Engineering Department, for giving us a constant source of inspiration, helping us prepare the project, personally correcting our work, and providing encouragement throughout the project. In this regard, we are especially thankful to our Head of Department, Dr. Himadri Biswas, for steering us through the tough as well as easy phases of the project in a result-oriented manner with concerned attention. A special thanks to all the faculty members of our Computer Science Department.

We will also love to take this opportunity to tell the readers of the material that their comments, be it appreciation or criticism, would be the most valuable thing to us. That will be something we will always be thankful for.

Date:________________________________

SHRUTI GUPTA
Roll No: 25500120052
Reg. No.: 202550100110048
SIMRAN SHARMA
Roll No: 25500120061
Reg. No.: 202550100110039
SRISHTI BOSE
Roll No: 25500120065
Reg. No.: 202550100110035
ANUBHAV MISHRA
Roll No: 25500120071
Reg. No.: 202550100110029

CONTENTS

ABSTRACT

Day by day, the cases of heart diseases are increasing at a rapid rate, emphasising the significance of predicting these diseases beforehand. A precise and efficient diagnosis becomes a challenging task. This report draws inspiration from the impact of emerging technologies in various industries, especially healthcare, and focuses on predicting the likelihood of a patient having a heart disease using their medical attributes. The heart disease prediction system developed for this study utilises machine learning algorithms including logistic regression, K-Nearest Neighbours (KNN), Support Vector Machine (SVM), Decision Tree Classifier, Random Forest Classifier, and Naïve Bayes. The aim is to classify patients accurately and improve prediction accuracy. The proposed model exhibits strong predictive power in identifying individuals at risk of heart disease, alleviating pressure, and reducing costs in the healthcare system. By analysing the heart disease dataset, various insights are derived, shedding light on the weight and interrelationships of different features. However, the primary objective of this project is to detect the probability of an individual being affected by a severe heart problem.

CHAPTER 1: INTRODUCTION

1.1 Introduction

More than half a billion people around the world continue to be affected by cardiovascular diseases, which accounted for 20.5 million deaths in 2021 – close to a third of all deaths globally and an overall increase on the estimated 121 million CVD deaths [6]. This staggering statistic reflects a significant health challenge, prompting the development of advanced tools for early detection and prevention. In response to this, our team has innovatively designed a comprehensive heart disease prediction model. Leveraging traditional and ensemble machine learning algorithms, including Logistic Regression, K-Nearest Neighbours, Support Vector Machine, Decision Tree Classifier, Random Forest Classifier, and Naïve Bayes, this model stands at the forefront of predictive analytics in healthcare.

Focused on identifying individuals at risk through a nuanced analysis of their medical history, this model goes beyond conventional approaches. Its robust capabilities extend to accurate classifier identification, leading to enhanced medical care, reduced costs, and the provision of valuable insights for predicting heart diseases. The integration of diverse machine learning techniques ensures a holistic approach, providing a nuanced understanding of the intricate factors contributing to heart disease risk.

1.2 Motivation

The motivation for developing the Heart Disease Prediction Model arises from the global impact of cardiovascular diseases, claiming a third of all deaths in 2021. According to the WHO estimate, the overall number of deaths from CVDs will rise to 23.6 million by 2030, with heart disease and stroke being the leading causes [4]. This alarming statistic underscores the urgent need for proactive measures, particularly in the realm of predictive analytics and healthcare technology. The motivation is fuelled by a commitment to mitigating the impact of cardiovascular diseases and fostering a paradigm shift towards preventative healthcare.

Key risk factors for heart disease, including an unhealthy lifestyle, hypertension, tobacco use, excessive sugar consumption, and obesity, highlight the multifaceted nature of this global health challenge. The societal and economic burdens associated with these risk factors necessitate innovative solutions. In this context, the heart disease prediction model emerges as a strategic response, not only addressing immediate health concerns but also aligning with a broader vision of promoting health and well-being.

The high diagnostic device costs, particularly burdensome for low- and middle-income countries, underscore the importance of cost-effective solutions. Although incomplete medical data can affect prediction accuracy, the paper reports high accuracy rates of up to 92% using the mentioned machine learning techniques, which is promising for predicting heart disease in the human body. This high precision, coupled with a commitment to accessibility and affordability, positions the model as a game-changer in global efforts towards early detection and prevention of heart diseases.

CHAPTER 2:
RELATED WORKS

2.1 Existing approaches in heart disease prediction

In recent years, there have been numerous studies exploring the application of data mining and machine learning techniques for the prediction and diagnosis of heart disease and other chronic conditions. These studies have utilised various algorithms and datasets to develop predictive models and improve accuracy rates. (Polaraju, Durga Prasad, & Tech Scholar, 2017) [1] presented a study on heart disease prediction utilising a multiple regression model, asserting its appropriateness for forecasting the likelihood of heart disease. The study involved a training dataset comprising 3000 instances with 13 attributes, further divided into 70% for training and 30% for testing.

In a parallel effort, Deepika and Seema (2017) concentrated on predicting chronic diseases by mining historical health records, employing Naïve Bayes, Decision Tree, Support Vector Machine (SVM), and Artificial Neural Network (ANN). Their comparative study revealed SVM's superior accuracy, especially for diabetes. (Mr. Chala Beyene & Prof. Pooja Kamat, 2018) [2] recommended various algorithms, including Naïve Bayes, Classification Tree, KNN, Logistic Regression, SVM, and ANN, with Logistic Regression demonstrating heightened accuracy. Their suggestion encompassed the use of WEKA software for automatic disease diagnosis. (Soni, Ansari, & Sharma, 2011) [3] proposed a non-linear classification algorithm for heart disease prediction, integrating big data tools such as Hadoop Distributed File System (HDFS), MapReduce, and SVM. Their investigation emphasised parallel SVM, yielding enhanced computation time compared to sequential SVM.

(Purushottam, Saxena, & Sharma, 2016) [4] put forth an efficient heart disease prediction system utilising data mining, achieving 86.3% accuracy in the testing phase and 87.3% in the training phase. (Mr. P. Sai Chandrasekhar Reddy, Mr. Puneet Palagi, & Ms. Jaya, 2017) [5] proposed heart disease prediction using an ANN algorithm in data mining, showcasing the system's accuracy in Java. Each study contributes to the collective understanding of heart disease prediction, showcasing diverse methodologies and algorithms in the realm of data mining and machine learning for healthcare practitioners and medical decision-making.

CHAPTER 3:
METHODOLOGY

3.1 Overview of the Prediction Model

The purpose of developing a heart disease prediction model is to identify individuals who are at risk of developing heart disease based on specific features or risk factors. By analysing these factors, the model can help healthcare professionals in their efforts to detect heart disease at an early stage, potentially leading to better treatment outcomes and improved patient care. Additionally, the model aims to contribute to the prevention of heart disease by identifying individuals who may be susceptible to developing the condition and allowing for targeted interventions, such as lifestyle modifications and medical interventions. Moreover, the model can aid in the development of personalised healthcare interventions by providing insights into an individual's specific risk factors and enabling tailored treatment plans that address their unique needs. Overall, the heart disease prediction model serves the overarching goal of improving patient outcomes and promoting heart health in the population.

The heart disease prediction model has a specific focus on predicting heart disease in a diverse population, as it combines five heart datasets [14] with a total of 918 observations. The datasets used in the model's curation include Cleveland (303 observations), Hungary (294 observations), Switzerland (123 observations), Long Beach, VA (200 observations), and the Stalog (Heart) Data Set (270 observations). This combination of datasets creates the largest heart disease dataset available for research purposes. There is no targeting of a specific type of heart disease in this model.

However, it is important to note that the limitations and constraints of the model include the availability of data and geographical considerations. The dataset for the model was taken from Kaggle, which may introduce bias and limitations associated with the source and collection methods.

Moreover, the aim is to make the heart disease prediction model accessible to middle-income and low-income sections of society. This suggests a focus on providing an affordable and widely available tool for predicting heart disease, potentially addressing disparities in healthcare access, and improving preventive measures in these underserved populations.

The key components of a heart disease prediction model include problem definition, data collection, data preprocessing, feature selection, data splitting, model selection, model training, model evaluation, result validation, model deployment, and ongoing monitoring and updating.

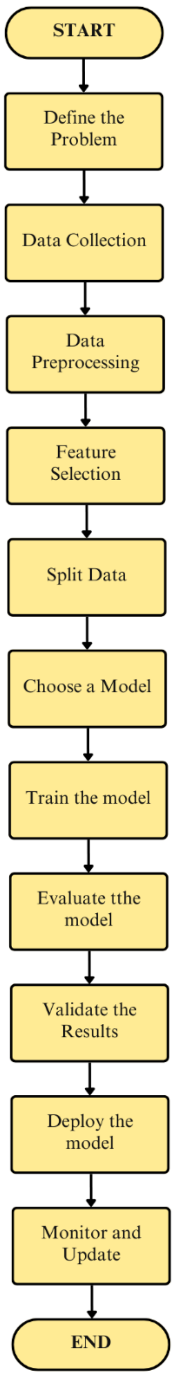


Fig. 1. Flowchart of the proposed approach

In the problem definition phase, the specific goal of the model is defined, which is to predict the likelihood of heart disease in individuals based on certain features or risk factors.

Data collection involves gathering relevant data, such as medical records, test results, and demographic information, from diverse sources to create a comprehensive dataset for training the model.

Data preprocessing is crucial for cleaning and preparing the dataset, which may involve handling missing values, dealing with outliers, and normalising or scaling variables.

Feature selection aims to identify the most important and relevant features that contribute to predicting heart disease. This step eliminates redundant or irrelevant variables.

Splitting the data into training and testing subsets allows for model training on a portion of the data and evaluating its performance on unseen data. This helps assess the model's generalisation capability.

Choosing a suitable model involves selecting from various machine learning algorithms that best fit the heart disease prediction task, considering factors like performance, interpretability, and computational requirements.

Model training involves using the training data to teach the selected algorithm to learn patterns and relationships between the features and the presence of heart disease.

Model evaluation is carried out using the testing data to measure the performance of the trained model, such as accuracy, precision, recall, and F1-score.

Validating the results involves assessing the model's performance on new, unseen data to ensure it can generalise well and produce reliable predictions.

The deployed model is made accessible for use in real-world scenarios, enabling healthcare professionals to input new patient data and receive predictions about their likelihood of developing heart disease.

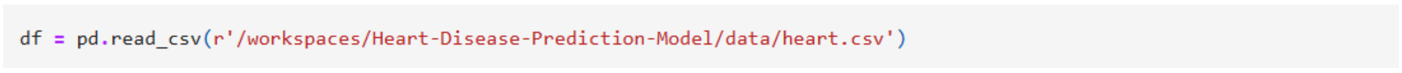
Monitoring and updating the model regularly is essential to ensuring its accuracy, reliability, and adaptability as new data becomes available and the understanding of heart disease evolves. Regular evaluations and updates help maintain the model's effectiveness over time.

3.2 Data Collection and Preprocessing

3.2.1 Data Collection

The standard heart disease dataset has been obtained from Kaggle. The dataset consists of 918 records and the following 11 features: age, sex, chest pain type, resting blood pressure, fasting blood sugar, cholesterol, resting electrocardiographic results, maximum heart rate, exercise-induced angina, ST depression induced by exercise, and the slope of the peak exercise ST segment.

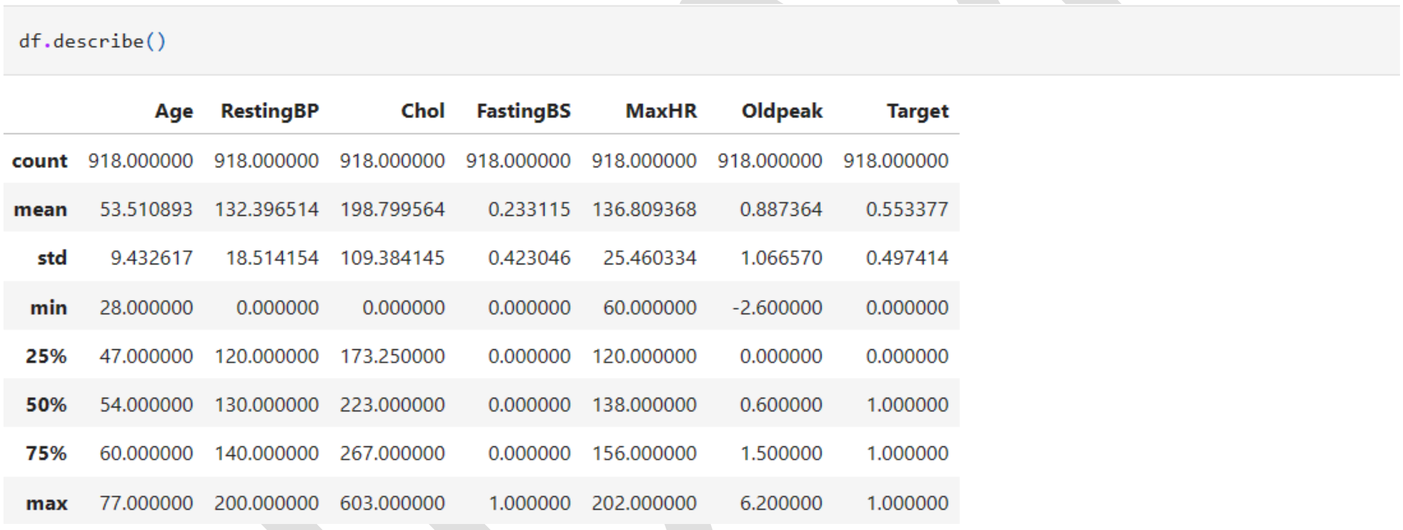
Once we have the .csv file, we use the Pandas library to read it [15].



```
df = pd.read_csv(r'/workspaces/Heart-Disease-Prediction-Model/data/heart.csv')
```

Fig. 2. Example to read .csv file using Pandas

We can gain introductory and statistical insights like shape of the dataset, count, mean, standard deviation, etc. of attributes by using functions like dataset.shape, dataset.describe(), etc.



```
df.describe()
```

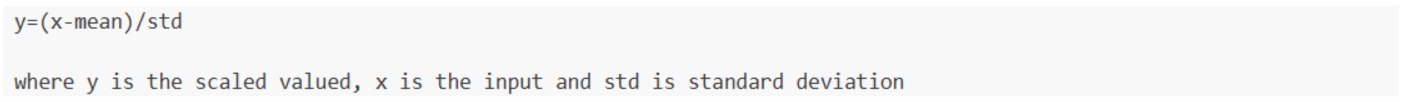
	Age	RestingBP	Chol	FastingBS	MaxHR	Oldpeak	Target
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Fig. 3. Example to generate statistical insights of the dataset [15]

3.2.2 Data Preprocessing

The first step in data preprocessing is cleaning the data. We need to address missing values, remove duplicate records, and handle outliers. This dataset does not have any missing values or duplicate records. In our case, we will focus on five numerical attributes: age, resting blood pressure (restingbp), cholesterol, maximum heart rate (maxhr), and ST depression induced by exercise relative to rest (oldpeak). Considering restingbp, it is not possible to have a resting blood pressure greater than 140/90mmHg and lesser than 90/60mmHg hence, we will remove problematic outliers in restingbp attribute.

The second step is feature scaling where we transform the values of different numerical features to fall within a fixed range. In our project, we have used the StandardScaler function to adjust the distribution of values such that the mean of observed values is zero and the standard deviation is one.



```
y=(x-mean)/std

where y is the scaled valued, x is the input and std is standard deviation
```

Fig. 4. Formula of StandardScaler()

The third step is encoding, a process to convert categorical data into numerical data. We have used the get_dummies function on those attributes that are categorical in nature namely sex, chestpaintype, cholesterol, restingecg, exerciseangina, and st_slope. This resulted in an increase in the number of columns from 11 to 21.

3.3 **Feature Selection**

Feature selection is a crucial step in supervised learning, where the presence of correlated features can impact model performance. While correlated features may not always worsen a model, they don't guarantee improvement either. There are three primary reasons for removing correlated features: to enhance the learning algorithm's speed, address the curse of dimensionality, and potentially decrease harmful bias.

Our dataset has a total of 12 attributes: 11 features and 1 target variable. The description of each attribute is provided below.

Features:

i. Age:
   a. Description: Age of the patient.
   b. Unit: Years
ii. Sex:
   a. Description: Sex of the patient.
   b. Categories: M (Male), F (Female)
iii. ChestPainType:
   a. Description: Chest pain type.
   b. Options: TA (Typical Angina), ATA (Atypical Angina), NAP (Non-Anginal Pain), ASY (Asymptomatic)
iv. RestingBP:
   a. Description: Resting blood pressure.
   b. Unit: mm Hg

v. Cholesterol:
   a. Description: Serum cholesterol.
   b. Unit: mm/dl

vi. FastingBS:
   a. Description: Fasting blood sugar.
   b. Options: 1 (if FastingBS > 120 mg/dl), 0 (otherwise)

vii. RestingECG:
   a. Description: Resting electrocardiogram results.
   b. Options: Normal (Normal), ST (ST-T wave abnormality - T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH (Probable or definite left ventricular hypertrophy by Estes' criteria)

viii. MaxHR:
   a. Description: Maximum heart rate achieved.
   b. Range: Numeric value between 60 and 202

ix. ExerciseAngina:
   a. Description: Exercise-induced angina.
   b. Options: Y (Yes), N (No)

x. Oldpeak:
   a. Description: Oldpeak = ST (Numeric value measured in depression)

xi. ST_Slope:
   a. Description: The slope of the peak exercise ST segment.
   b. Options: Up (Upsloping), Flat (Flat), Down (Downsloping)

Target:

a. Description: Output class.
b. Options: 1 (heart disease), 0 (normal)

In terms of speed, reducing the number of features often leads to improved efficiency, especially in scenarios where computation time is critical. However, if speed is not a primary concern, the decision to remove correlated features should be carefully weighed against potential harm to the model's performance.

Decreasing harmful bias is another consideration, particularly when correlated features are also correlated with the target variable. In such cases, retaining these features may provide valuable hints for accurate predictions. Algorithms like Naïve Bayes can directly benefit from positively correlated features, and others, like random forests, may indirectly benefit from them.

The probability of obtaining informative features is illustrated with an example of three features (A, B, and C), where A and B are highly correlated with each other and the target. The removal of such features may decrease the chances of selecting a "good" feature during sampling.

To summarise, removing correlated features may be necessary for speed, but careful consideration is required to avoid potential negative impacts on the algorithm's performance. Algorithms like decision trees often embed feature selection, and a recommended approach is to use a wrapper method for feature selection. Wrapper methods, albeit computationally expensive, selectively remove redundant features while preserving those that contribute directly to performance, thereby preventing overfitting.

In this specific scenario where features are not highly correlated, except for two, attempts to reduce the number of features resulted in an increase in model accuracy. The decision to retain all the features was made because of this outcome, highlighting the significance of adopting a meticulous approach to feature selection that is customised to the specific attributes of the dataset and the needs of the learning algorithm.

3.4 Model Selection

Model selection is the process of choosing the most appropriate model for a specific problem or dataset. It involves evaluating various models and selecting the one that provides the best performance in terms of accuracy, precision, recall, and other evaluation metrics. In this project, we have utilised both traditional and ensemble learning models for the purpose of classifying the data.

3.4.1 Traditional Classifiers

Traditional classifiers form the bedrock of machine learning algorithms, offering foundational approaches to address classification tasks. Naïve Bayes, a probabilistic classifier, simplifies the classification process by assuming feature independence, making it particularly effective for text classification and spam filtering. Decision Trees, on the other hand, leverage a tree-like structure where each node represents a decision based on feature values, making them intuitive to interpret. Logistic Regression, a linear model, is widely employed in binary and multiclass classification scenarios due to its simplicity and interpretability. K-Nearest Neighbours classifies instances based on the majority class among their nearest neighbours, making it suitable for both small and large datasets. Support Vector Machines construct hyperplanes to separate classes in high-dimensional spaces, demonstrating robustness in complex classification tasks.

A. Logistic Regression

Logistic Regression, a supervised machine learning algorithm, is instrumental in classification tasks, predicting the probability that an instance belongs to a given class. This algorithm's name derives from its classification focus, and it utilises a sigmoid function to estimate class probabilities based on the output of the linear regression function. Unlike linear regression, logistic regression is tailored for classification problems, predicting probabilities between 0 and 1 for categorical outcomes. It produces a probability of success, given the values of the feature variables, rather than just a predicted class, which enables sorting the observations by probability of success and setting an arbitrary cutoff for classification [10]. Analysing the relationship between independent variables and a binary dependant variable, logistic regression fits an "S"-shaped logistic function, indicating the likelihood of an event. This versatile algorithm is applicable to various data types, providing probabilities and facilitating feature selection for classification tasks. With its ability to classify new data and determine effective variables for decision-making, logistic regression stands as a powerful tool in machine learning.

The sigmoid function, characterised by an "S"-shaped curve, is a mathematical tool crucial in logistic regression for mapping predicted values into probabilities. This function transforms any real value to a range between 0 and 1, enforcing a limit on logistic regression's predicted values and forming the characteristic "S"-shaped curve.

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

Known also as the logistic function, this curve is central to logistic regression's probabilistic interpretation, where values above a threshold tend toward 1, and those below the threshold tend toward 0. In logistic regression, the sigmoid function plays a pivotal role in converting the linear combination of input features into probabilities, facilitating binary classification decisions based on a chosen threshold (commonly 0.5). Understanding the significance of the sigmoid function is fundamental in comprehending how logistic regression models translate input features into probability estimates for binary classification problems.

B. K-Nearest Neighbours

K-Nearest Neighbours (KNN) stands out as a fundamental and versatile classification algorithm in Machine Learning, belonging to the supervised learning domain. Its applications span pattern recognition, data mining, and intrusion detection, making it widely applicable in real-life scenarios. KNN is a non-parametric algorithm [9]. Non-Parametric means either there are no parameters or fixed number of parameters irrespective of size of data. Instead, parameters would be determined by the size of the training dataset. An essential characteristic of KNN is its non-parametric nature, avoiding assumptions about the underlying data distribution, unlike algorithms such as Gaussian Mixture Model. This algorithm relies on prior data, termed training data, where coordinates are classified into groups, based on attributes. KNN makes predictions by considering the majority class among the k-nearest neighbours of a given data point, offering an intuitive and less complex model training approach. The choice of a distance metric, such as Euclidean distance, plays a crucial role in determining the similarity between data points. KNN is known for its flexibility, suitable for both classification and regression tasks, with the careful selection of 'k' (number of neighbours) being a critical parameter based on the dataset's characteristics.

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

i. Euclidean Distance

Euclidean distance, a fundamental distance metric in the K-Nearest Neighbours (KNN) algorithm, measures the Cartesian distance between two points in a plane or hyperplane. It is represented as the length of the straight line joining the two points and is commonly used in various applications, including machine learning algorithms like KNN, where it assesses the similarity or dissimilarity between data points. The formula for Euclidean Distance between two points in a 2D space is given by:

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Key points include its visualisation as a straight-line distance and its role in calculating the net displacement or straight-line distance between two states of an object.

ii. Manhattan Distance

Manhattan Distance, another crucial distance metric for KNN, is employed when the total distance travelled by an object is of interest rather than just the displacement. This metric sums the absolute differences between the coordinates of points in n-dimensions, considering the horizontal and vertical distances travelled between two points. The formula for Manhattan Distance in a 2D space is given by:

$$Distance = |x_2 - x_1| + |y_2 - y_1|$$

For higher-dimensional spaces, it generalises by summing the absolute differences along each dimension. Manhattan Distance is applicable in scenarios like navigation or route planning, emphasising movement along grid-like paths.

iii. Minkowski Distance

Minkowski distance serves as a generalised form encompassing both Euclidean and Manhattan distances. The formula for Minkowski Distance [9] is given by:

$$D_{Mink} = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

The formula demonstrates that when $p = 2$ it aligns with the Euclidean distance, and when $p = 1$, it corresponds to the Manhattan distance. The Minkowski distance accommodates various scenarios by adjusting the value of p, offering flexibility in capturing different distance metrics. This adaptability makes it a valuable tool in distance calculations, providing a unified perspective on Euclidean and Manhattan distances within the context of KNN and other applications.

C. Support Vector Machine

Support Vector Machine (SVM) is a versatile supervised machine learning algorithm applicable to both classification and regression, although it is predominantly associated with classification tasks. The primary goal of SVM is to discover the optimal hyperplane in an N-dimensional space, effectively separating data points into different classes within the feature space. This hyperplane serves as a decision boundary, maximising the margin, which represents the distance between the closest points of different classes to the hyperplane. The dimension of the hyperplane corresponds to the number of features, making visualisation challenging beyond three features. Support Vectors, the data points closest to the hyperplane, play a crucial role in determining its position and orientation. SVM efficiently handles non-linear decision boundaries through kernel tricks, transforming the feature space to a higher dimension. Widely used in domains such as image classification, text classification, bioinformatics, and finance, SVM's versatility extends to both linear and non-linear classification tasks, making it suitable for a diverse range of problems. However, as the number of features increases, visualising the hyperplane and decision boundaries becomes increasingly challenging.

The learning problem setting for SVMs [11] is as follows: there is some un-known and nonlinear dependency (mapping, function) $y = f(x)$ between some high-dimensional input vector x and scalar output y (or the vector output y as in the case of multiclass SVMs). There is no information about the underlying joint probability functions. Thus, one must perform a distribution-free learning. The only information available is a training data set $D = \{(x_i, y_i) \in X \times Y\}, i = 1, l$ where l stands for the number of the training data pairs and is therefore equal to the size of the training data set D. Often, y_i is denoted as d_i, where d stands for a desired (target) value.

D. Decision Tree

A Decision Tree is a versatile supervised learning technique applicable to both classification and regression problems, with a predominant use in solving classification tasks. Its tree structure comprises internal nodes representing dataset features, branches embodying decision rules, and leaf nodes holding output labels or values. The tree's branches represent decision rules based on feature values, guiding different paths within the tree. While Decision Trees can address regression tasks, they excel in classification due to their interpretability and strengths. They offer insights into feature importance for decision-making and provide a visual representation facilitating understanding. Known for capturing non-linear relationships, Decision Trees find utility in diverse domains, including finance, healthcare, and marketing. Their interpretability and ease of understanding make them valuable, and they can serve as base learners in ensemble methods like Random Forests or Gradient Boosting to enhance predictive performance. However, Decision Trees are susceptible to overfitting, especially in deep structures, and mitigation techniques like pruning or ensemble methods are employed. The goal is to produce a tree that can generalise to predict unseen samples [7].

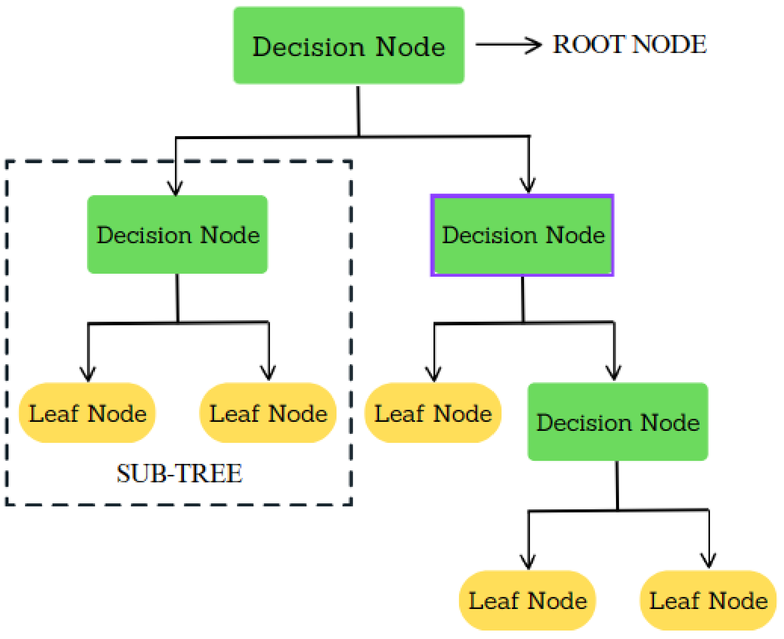


Fig. 5. Working of a Decision Tree

E. Naïve Bayes Classifier

The Naïve Bayes algorithm, a supervised learning technique rooted in Bayes' theorem, specialises in solving classification problems, with a particular focus on text classification involving high-dimensional training datasets. Recognised for its simplicity and efficiency, Naïve Bayes is an effective tool for constructing rapid machine learning models capable of swift predictions. Operating as a probabilistic classifier, it predicts based on the probability of an object belonging to a specific class. Some notable applications of Naïve Bayes encompass spam filtration, sentiment analysis, and article classification. The term "Naïve" in Naïve Bayes stems from the assumption that the occurrence of a particular feature is independent of the occurrence of other features.

Bayes' Theorem:

The term "Bayes" in Naïve Bayes comes from Bayes' Theorem, a fundamental principle used to determine the probability of a hypothesis with prior knowledge. It relies on conditional probability.

The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B,

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true,

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

3.4.2 Ensemble Classifiers

Ensemble algorithm [7] is a machine learning technique that combines multiple models to improve the accuracy and robustness of the prediction. Ensemble classifiers represent a paradigm shift in machine learning, emphasising the strength of collective decision-making. Random Forests, an ensemble of decision trees, mitigate overfitting by aggregating predictions from multiple trees. Gradient Boosting Machines, such as XGBoost, sequentially build weak learners to iteratively correct errors, achieving high predictive accuracy. AdaBoost combines weak classifiers to form a robust ensemble, dynamically adjusting weights to emphasise misclassified instances. Bagging techniques aggregate predictions from independently trained models to enhance overall performance. Voting classifiers harness the diversity of multiple classifiers, with each having equal or weighted influence on the final decision. Stacking further elevates ensemble learning by training a meta-model to combine predictions from diverse base models. These ensemble approaches capitalise on the synergy of individual classifiers, fostering improved generalisation and adaptability across a spectrum of classification tasks.

A. Random Forest

Random Forest is a highly regarded supervised learning algorithm designed for both classification and regression tasks in machine learning. Grounded in the concept of ensemble learning, it leverages multiple classifiers to enhance overall performance. By employing numerous decision trees, each trained on diverse subsets of the dataset, Random Forest mitigates overfitting and achieves higher accuracy. Instead of relying on a single decision tree, the algorithm computes the average predictions from all trees, determining the final output through majority voting. The customisation options, such as the number of trees and tree depth, contribute to its adaptability. Renowned for its versatility, Random Forest can handle various data types and feature spaces, finding applications in domains like finance, healthcare, and image classification.

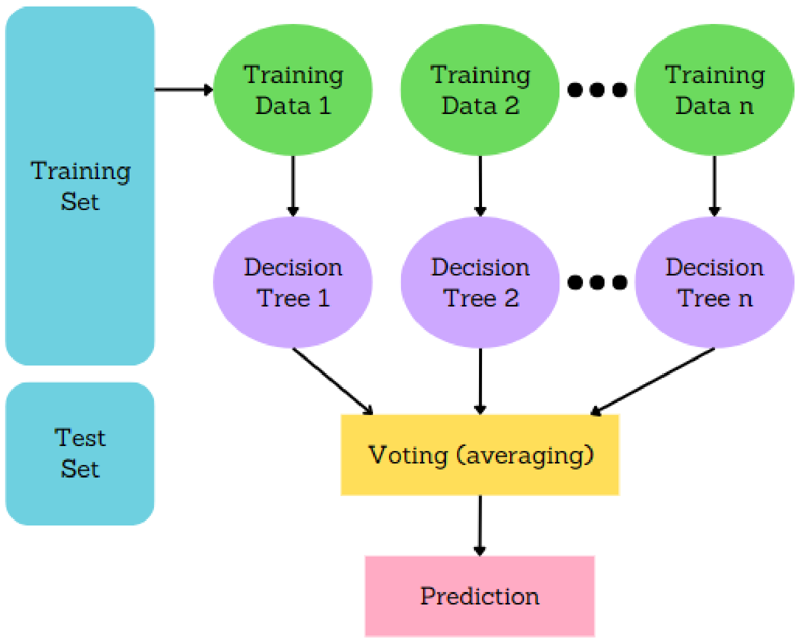


Fig. 6. Working of a Random Forest Classifier

B. Bagging

Bagging, short for Bootstrap Aggregation, is a powerful ensemble learning technique designed to diminish the variance of decision trees, particularly beneficial for unstable models. The process begins with a dataset D containing d tuples. At each iteration i, a training set Di is sampled with replacement from D, a technique known as bootstrapping. Subsequently, a classifier model Mi is trained independently on each Di. These classifiers collectively contribute to the bagged classifier M*, which aggregates their predictions through a voting mechanism, ultimately assigning the class with the highest votes to an unknown sample X.

[12] The implementation of bagging involves several key steps. Firstly, multiple subsets are generated from the original dataset, each containing an equal number of tuples, with observations selected through replacement. Next, a base model is constructed on each of these subsets, and crucially, each model is trained in parallel, independent of one another. The final predictions are then determined by amalgamating the individual predictions from all the models. This process ensures robustness and stability in the resulting model, effectively mitigating overfitting and enhancing predictive performance.

In bagging, the ensemble approach aims to address the limitations of weak models by aggregating their predictions. The three main steps involve sampling equal-sized subsets with replacement, training weak models independently and concurrently on each subset, and finally, combining the outcomes to produce a result. The aggregation can take different forms, such as averaging the results in regression tasks or selecting the majority class in classification tasks. Bagging is particularly advantageous for reducing variance, making it a popular choice for tree-based machine learning models like decision trees and random forests, where its impact on stability and generalisation is most pronounced.

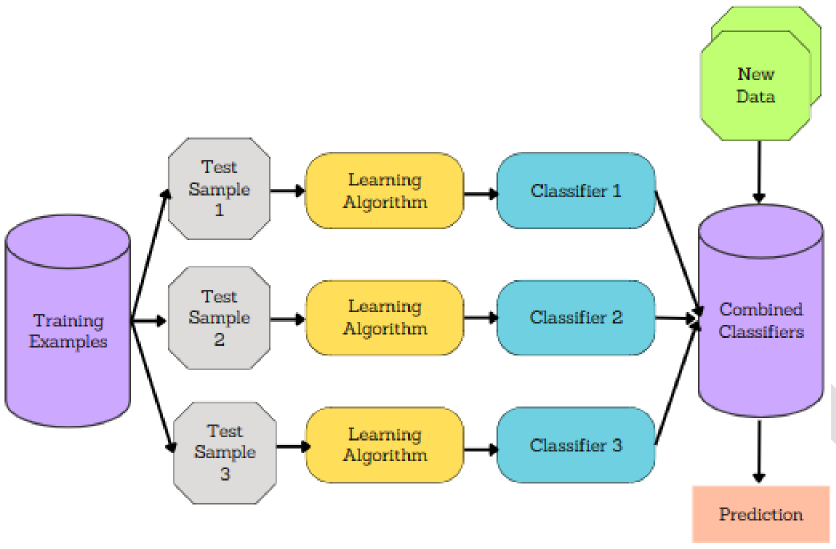


Fig. 7. Working of Bagging Classifier [12]

C. Adaptive Boosting

AdaBoost, short for Adaptive Boosting, is an ensemble learning technique used for predictive modelling in machine learning. As an Ensemble Method, AdaBoost combines the strength of multiple weak learners to create a robust and accurate model. Decision trees with just one level, often termed Decision Stumps, are the most employed estimators in AdaBoost. This approach initiates the model construction by assigning equal weights to all data points and iteratively applies larger weights to incorrectly categorised points. The subsequent models prioritise points with greater weights, leading to a refined and optimised ensemble.

AdaBoost's distinctive feature lies in its ability to adapt to the complexities of the dataset by assigning varying weights to data points based on their classification accuracy. In each iteration, the algorithm focuses on the previously misclassified points, assigning them higher weights to rectify the errors. This iterative process continues until the model achieves a lower error rate, contributing to the overall strength and accuracy of the ensemble. AdaBoost's efficacy is particularly notable when dealing with weak learners, enhancing their individual predictive capabilities through strategic weighting and aggregation. This adaptability makes AdaBoost a powerful tool for addressing diverse problem statements in machine learning.

SurTech

D. Extreme Gradient Boosting

XGBoost, or Extreme Gradient Boosting, is a highly optimised and distributed gradient boosting library crafted for efficient and scalable machine learning model training. Renowned for its ensemble learning method, XGBoost aggregates predictions from multiple weak models, establishing itself as one of the most widely utilised algorithms in the field. A standout characteristic of XGBoost is its adeptness at handling missing values in real-world datasets, eliminating the need for extensive pre-processing. This capability significantly streamlines the management of diverse and incomplete datasets, contributing to the algorithm's widespread popularity [13].

The efficiency of XGBoost in handling large datasets is further amplified by its built-in support for parallel processing, facilitating the training of models on substantial datasets within reasonable timeframes. This scalability is essential for tackling complex machine learning tasks effectively. XGBoost finds applications in various domains, including classification, regression, Kaggle competitions, recommendation systems, and click-through rate prediction. Highly customisable, XGBoost provides practitioners with the flexibility to fine-tune various model parameters, tailoring the algorithm to meet specific requirements across diverse applications. Initially proposed by researchers at the University of Washington and implemented in C++, XGBoost optimises the training process for gradient boosting, ensuring efficiency, scalability, and the ability to handle missing values are integral to its design. XGBoost's innovation extends to its dynamic determination of decision tree depth, mitigating overfitting through penalisation parameters and proportional shrinking of leaf nodes. Leveraging Newton's tree boosting for optimised learning and incorporating randomisation parameters further bolsters its performance. The amalgamation of these features positions XGBoost as a state-of-the-art solution for various machine learning tasks. Its dynamic adjustment of tree depth, control over leaf node size, utilisation of advanced learning methods, and incorporation of randomisation parameters collectively establish it as a versatile and powerful algorithm in the realm of gradient boosting.

E. MaxVoting Classifier

The MaxVoting Classifier is an ensemble learning algorithm that combines the predictions of multiple base classifiers in order to make a final decision. It operates by selecting the class label that receives the majority vote from the individual classifiers. The MaxVoting Classifier can be applied to both classification and regression problems. It is particularly useful when the base classifiers have different strengths and weaknesses, as it allows for the exploitation of their complementary characteristics. This information is based on my knowledge and understanding of ensemble learning algorithms.

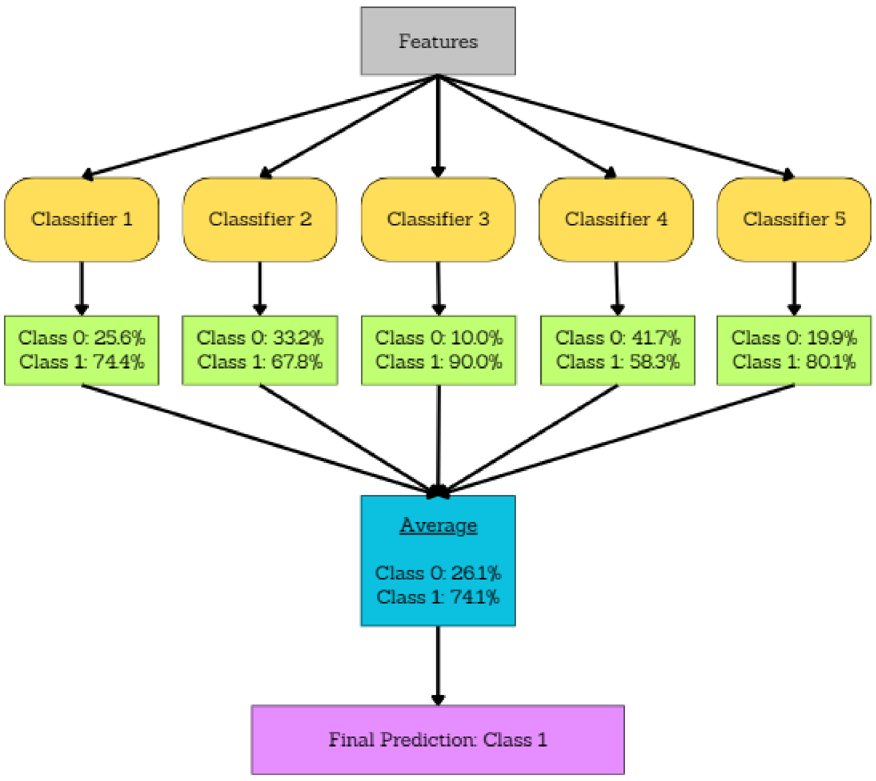


Fig. 8. Working of MaxVoting Classifier

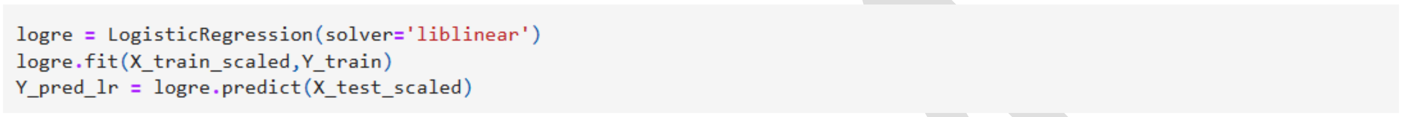
3.5 Training and Evaluation

3.5.1 Model Training

The training process of the model begins with the pivotal step of splitting the dataset into distinct training and testing sets. This division is important to assess the model's performance accurately and ensure its generalisation to new, unseen data. Allocating 80 percent of the dataset for training and reserving the remaining 20 percent for testing strikes a balance between providing the model with sufficient examples to learn patterns, correlations, and attributes while maintaining a robust evaluation metric.

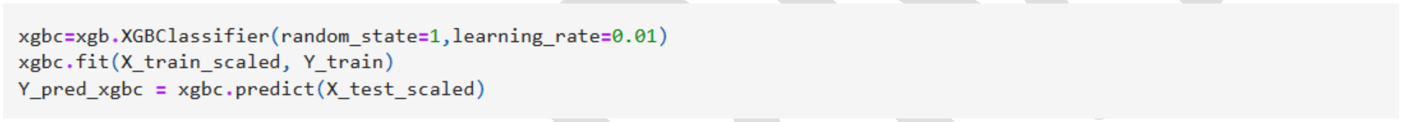
During the training phase, the model extensively utilises the training dataset, employing sophisticated algorithms to comprehend the intricate relationships between input features and target outcomes. By exposing the model to diverse instances, it hones its ability to recognise patterns, adapt to variations, and make informed predictions. This iterative learning process involves adjusting internal parameters, optimising the model's predictive capabilities.

The rationale behind the 80-20 split lies in the need to strike a delicate equilibrium between training sufficiency and effective evaluation. Allocating a substantial portion to training allows the model to grasp the underlying structures within the data, enhancing its ability to make accurate predictions. Simultaneously, reserving a portion for testing serves as a litmus test for the model's performance on unseen data, guarding against overfitting and gauging its ability to generalise beyond the training set. This strategic partitioning of data is foundational in the pursuit of robust, reliable machine learning models.



```
logre = LogisticRegression(solver='liblinear')
logre.fit(X_train_scaled,Y_train)
Y_pred_lr = logre.predict(X_test_scaled)
```

Fig. 9. Example of Model Training – Logistic Regression [15]



```
xgbc=xgb.XGBClassifier(random_state=1,learning_rate=0.01)
xgbc.fit(X_train_scaled, Y_train)
Y_pred_xgbc = xgbc.predict(X_test_scaled)
```

Fig. 10. Example of Model Training – XGBoost Classifier [15]

3.5.2 Model Evaluation

Model evaluation is a pivotal phase involving the use of diverse metrics to assess the effectiveness and limitations of a machine learning model. This critical process plays a key role in evaluating the model's performance during the initial research stages and continuously monitoring its functionality over time. In the context of assessing a classification model, common metrics such as accuracy, precision, AUC (area under the ROC curve), and the confusion matrix prove invaluable. In our specific project, we have used accuracy, precision and confusion matrix as our primary evaluation metrics. Accuracy, a foundational measure, quantifies the ratio of correct predictions to the total predictions made by the classifier, providing a high-level overview of the model's overall performance.

The utilisation of the confusion matrix metrics goes a step further, offering a detailed breakdown of both correct and incorrect classifications for each class. This granular analysis proves particularly useful in comprehending the distinctions between classes, a critical aspect when the cost of misclassification varies across different classes or when there is a significant disparity in test data distribution. This type of in-depth analysis becomes especially crucial in applications such as medical diagnosis, where the consequences of a false positive or false negative can have distinct and severe implications. Overall, model evaluation is an iterative and essential process that contributes to refining the model's accuracy and reliability for real-world applications.

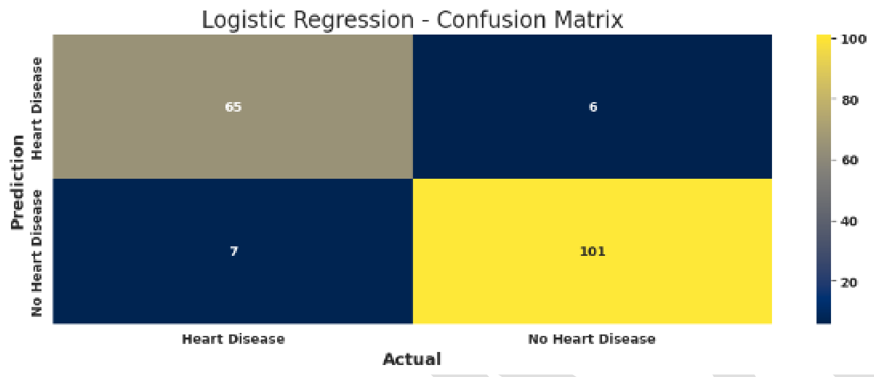


Fig. 11. Example of Confusion Matrix – Logistic Regression (80/20 split) [15]

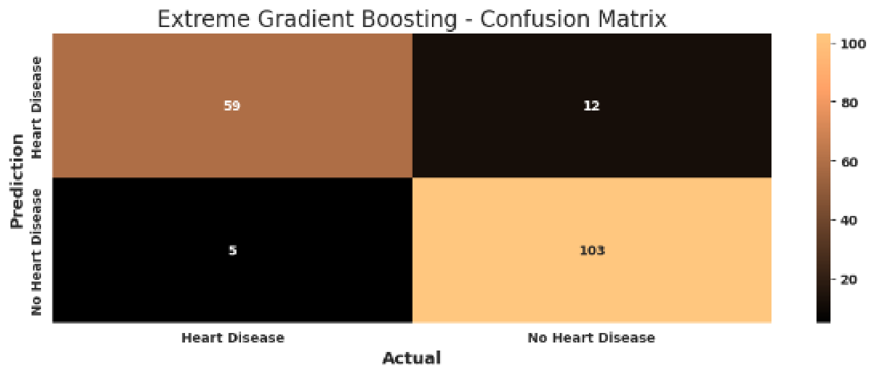


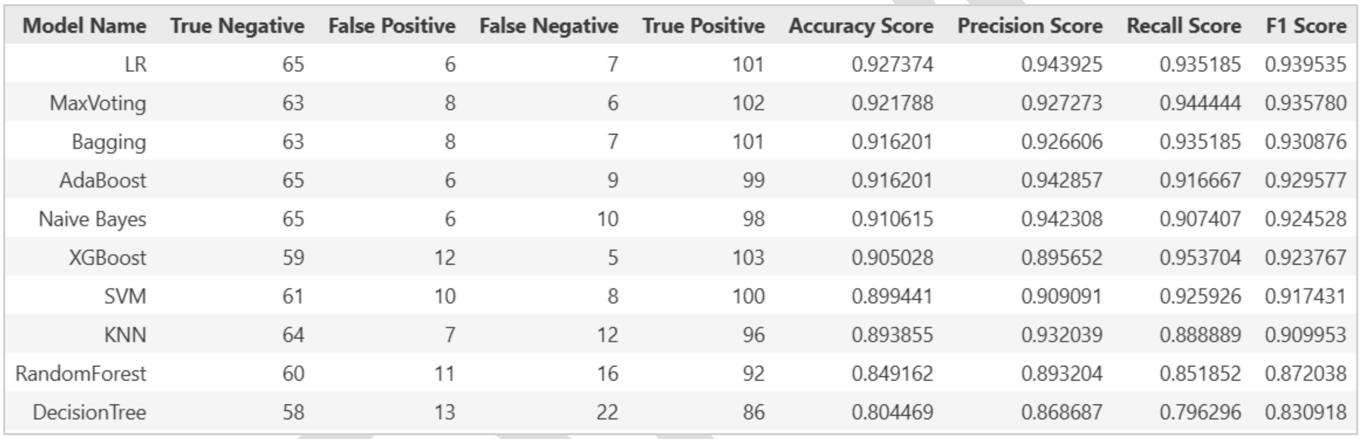
Fig. 12. Example of Confusion Matrix – XGBoost Classifier (80/20 split) [15]

The confusion matrix [8] shows the total number of cases where:

- The model predicted *no heart disease* and the actual label is *no heart disease* (*true negatives*, bottom right)
- The model predicted *heart disease* and the actual label is *heart disease* (*true positives*, top left)
- The model predicted *no heart disease* and the actual label is *heart disease* (*false negatives*, bottom left)
- The model predicted *heart disease* and the actual label is *no heart disease* (*false positives*, top right)

From these core values, you can calculate a range of other metrics that can help you evaluate the performance of the model. For example:

- Accuracy: (TP+TN)/(TP+TN+FP+FN) - out of all the predictions, how many were correct?
- Recall: TP/(TP+FN) - of all the cases that *are* positive, how many did the model identify?
- Precision: TP/(TP+FP) - of all the cases that the model predicted to be positive, how many are actually positive?



Model Name	True Negative	False Positive	False Negative	True Positive	Accuracy Score	Precision Score	Recall Score	F1 Score
LR	65	6	7	101	0.927374	0.943925	0.935185	0.939535
MaxVoting	63	8	6	102	0.921788	0.927273	0.944444	0.935780
Bagging	63	8	7	101	0.916201	0.926606	0.935185	0.930876
AdaBoost	65	6	9	99	0.916201	0.942857	0.916667	0.929577
Naive Bayes	65	6	10	98	0.910615	0.942308	0.907407	0.924528
XGBoost	59	12	5	103	0.905028	0.895652	0.953704	0.923767
SVM	61	10	8	100	0.899441	0.909091	0.925926	0.917431
KNN	64	7	12	96	0.893855	0.932039	0.888889	0.909953
RandomForest	60	11	16	92	0.849162	0.893204	0.851852	0.872038
DecisionTree	58	13	22	86	0.804469	0.868687	0.796296	0.830918

Fig. 13. Model Comparison Table sorted by Accuracy Score (80/20 Split) [15]

From Fig.13., we can infer that Logistic Regression has the highest accuracy at .927 out of 1.

3.6 <u>Deploy the model</u>

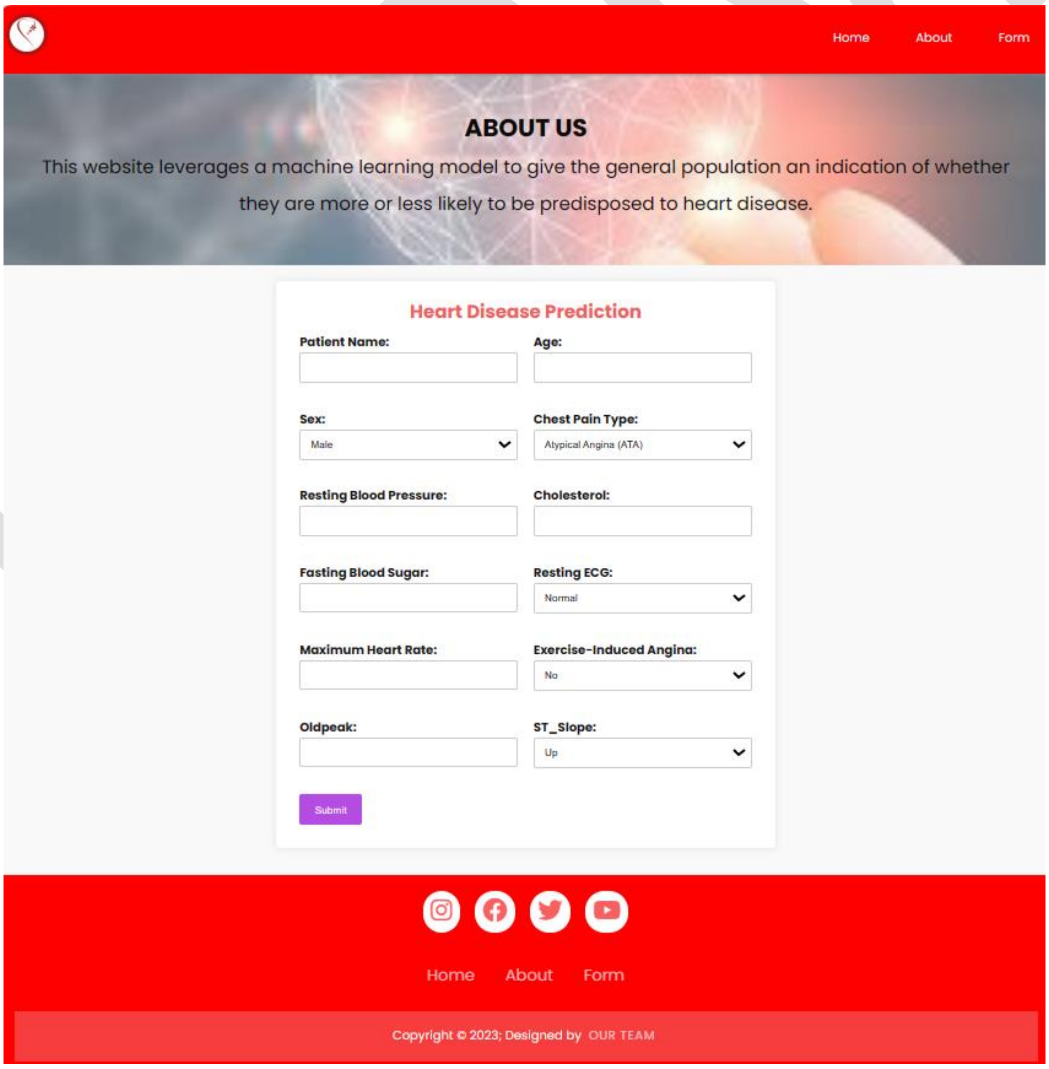
To deploy the heart disease prediction model for practical use, the most accurate model (Logistic Regression) is saved in a pickle file, a serialised binary file format in Python. This pickle file essentially encapsulates the trained model, preserving its parameters and structure. During deployment, this file is integrated into a website, particularly using Flask, HTML, CSS, and Python.

Our deployment process involves creating a user-friendly interface where input for prediction can be entered. This is typically done through a form on the website, allowing users to input relevant health attributes. Flask, a web framework for Python, is utilised to handle the backend processing. The form inputs are then fed into the stored model using Python scripts.

After processing the input through the saved model, the output prediction is generated and displayed on the website. This user-friendly approach allows individuals to easily interact with the heart disease prediction system without requiring knowledge of the underlying machine learning model or its complexities.

This deployment setup not only ensures accessibility but also facilitates scalability and maintenance. The Flask web application serves as the interface, while the underlying model, stored as a pickle file, seamlessly integrates with the website, making predictions on new data. This enables real-time prediction capabilities for users interacting with the deployed heart disease prediction system.

In essence, the deployment involves combining the machine learning model's predictive capabilities with a user-friendly web interface, providing a practical and accessible tool for predicting heart disease based on input health attributes.



ABOUT US

This website leverages a machine learning model to give the general population an indication of whether they are more or less likely to be predisposed to heart disease.

Heart Disease Prediction

Patient Name:	Age:
Sex: Male	Chest Pain Type: Atypical Angina (ATA)
Resting Blood Pressure:	Cholesterol:
Fasting Blood Sugar:	Resting ECG: Normal
Maximum Heart Rate:	Exercise-Induced Angina: No
Oldpeak:	ST_Slope: Up

Copyright © 2023; Designed by OUR TEAM

Fig. 14. Heart Disease Prediction Model Website [16]

CHAPTER 4:

RESULTS

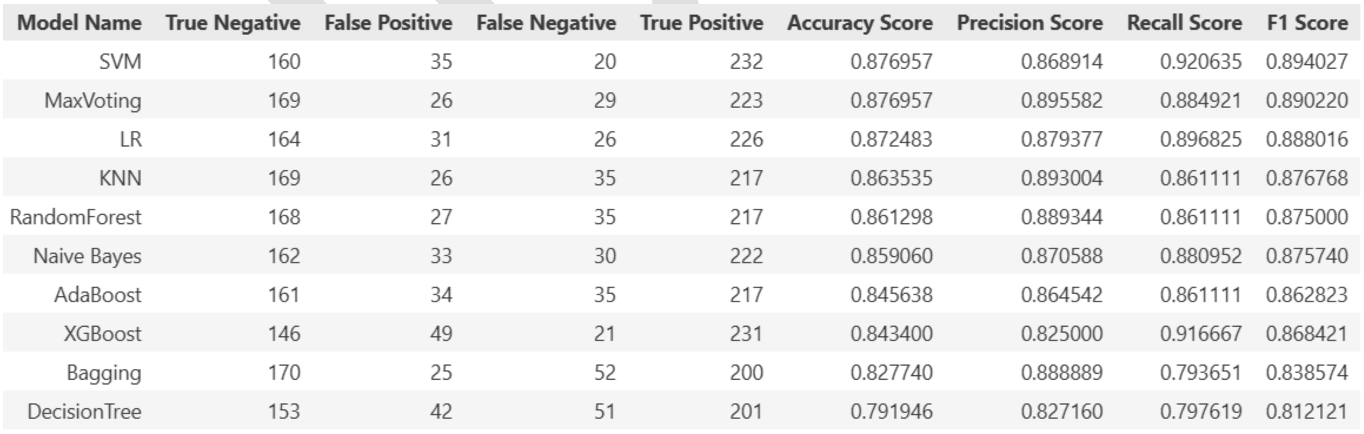
4.1 **Performance Metrics**

Performance metrics are fundamental tools in the evaluation of classifier models, offering valuable insights into their effectiveness and reliability. In the domain of machine learning, where machine learning algorithms like Support Vector Machine, Logistic Regression, and more play a crucial role in predicting outcomes and making decisions, assessing their performance is essential for refining their accuracy and ensuring their suitability for specific tasks.

Performance metrics provide quantifiable measures that shed light on various aspects of a classifier's behaviour. From assessing the accuracy of predictions to understanding a classifier's ability to handle diverse scenarios and minimise errors, these metrics serve as essential benchmarks for gauging model performance.

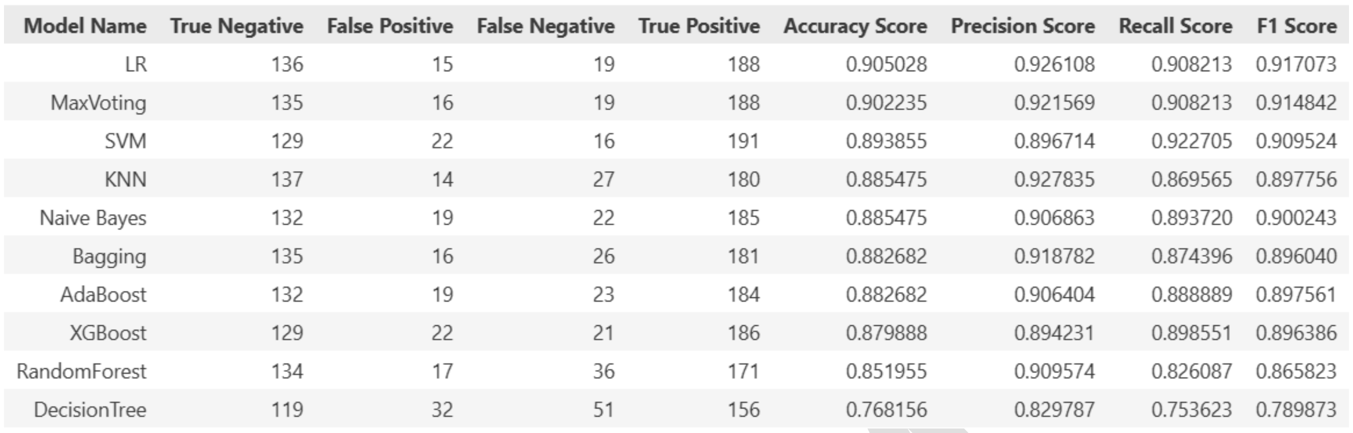
Performance metrics serve as a guide for our team, offering a systematic way to evaluate and compare different models. By analysing these metrics, we were able to gain a nuanced understanding of a classifier's strengths and limitations, enabling them to make informed decisions about model optimisation and fine-tuning.

Below we have the model comparison table across different train/test split ratios. The data splitting ratios in considerations are 50/50(50% training set, 50% test set), 60/40(60% training set, 40% test set), 70/30(70% training set, 30% test set), 80/20(80% training set, 20% test set) and 90/10(90% training set, 10% test set).



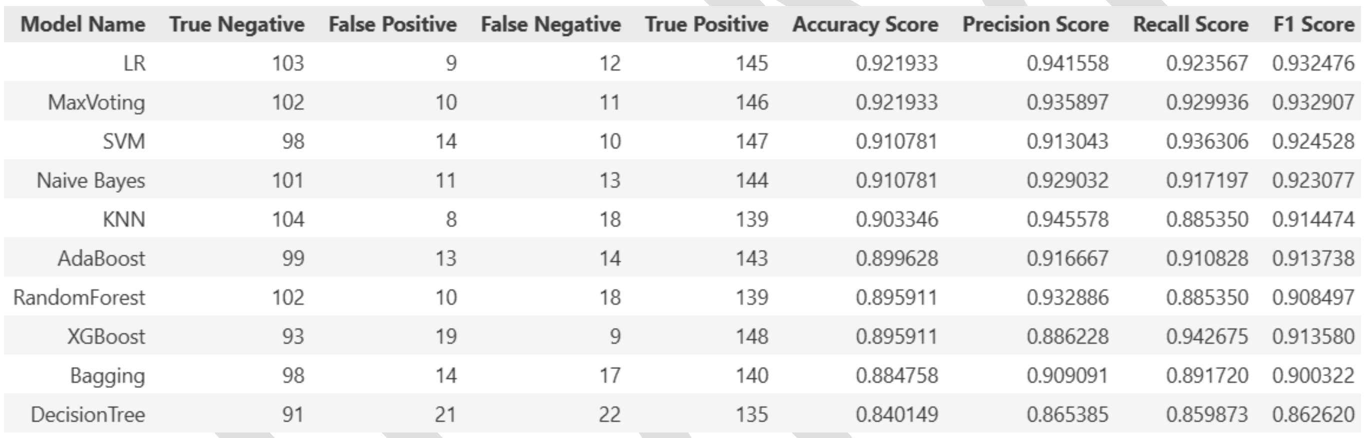
Model Name	True Negative	False Positive	False Negative	True Positive	Accuracy Score	Precision Score	Recall Score	F1 Score
SVM	160	35	20	232	0.876957	0.868914	0.920635	0.894027
MaxVoting	169	26	29	223	0.876957	0.895582	0.884921	0.890220
LR	164	31	26	226	0.872483	0.879377	0.896825	0.888016
KNN	169	26	35	217	0.863535	0.893004	0.861111	0.876768
RandomForest	168	27	35	217	0.861298	0.889344	0.861111	0.875000
Naive Bayes	162	33	30	222	0.859060	0.870588	0.880952	0.875740
AdaBoost	161	34	35	217	0.845638	0.864542	0.861111	0.862823
XGBoost	146	49	21	231	0.843400	0.825000	0.916667	0.868421
Bagging	170	25	52	200	0.827740	0.888889	0.793651	0.838574
DecisionTree	153	42	51	201	0.791946	0.827160	0.797619	0.812121

Fig. 15. Model Comparison Table (50/50 Split)



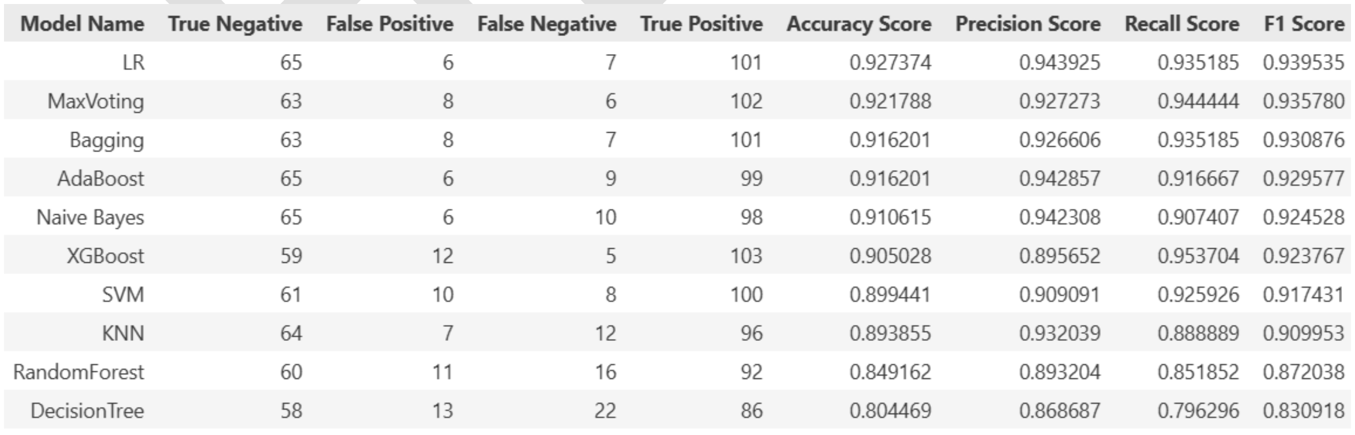
Model Name	True Negative	False Positive	False Negative	True Positive	Accuracy Score	Precision Score	Recall Score	F1 Score
LR	136	15	19	188	0.905028	0.926108	0.908213	0.917073
MaxVoting	135	16	19	188	0.902235	0.921569	0.908213	0.914842
SVM	129	22	16	191	0.893855	0.896714	0.922705	0.909524
KNN	137	14	27	180	0.885475	0.927835	0.869565	0.897756
Naive Bayes	132	19	22	185	0.885475	0.906863	0.893720	0.900243
Bagging	135	16	26	181	0.882682	0.918782	0.874396	0.896040
AdaBoost	132	19	23	184	0.882682	0.906404	0.888889	0.897561
XGBoost	129	22	21	186	0.879888	0.894231	0.898551	0.896386
RandomForest	134	17	36	171	0.851955	0.909574	0.826087	0.865823
DecisionTree	119	32	51	156	0.768156	0.829787	0.753623	0.789873

Fig. 16. Model Comparison Table (60/40 Split)



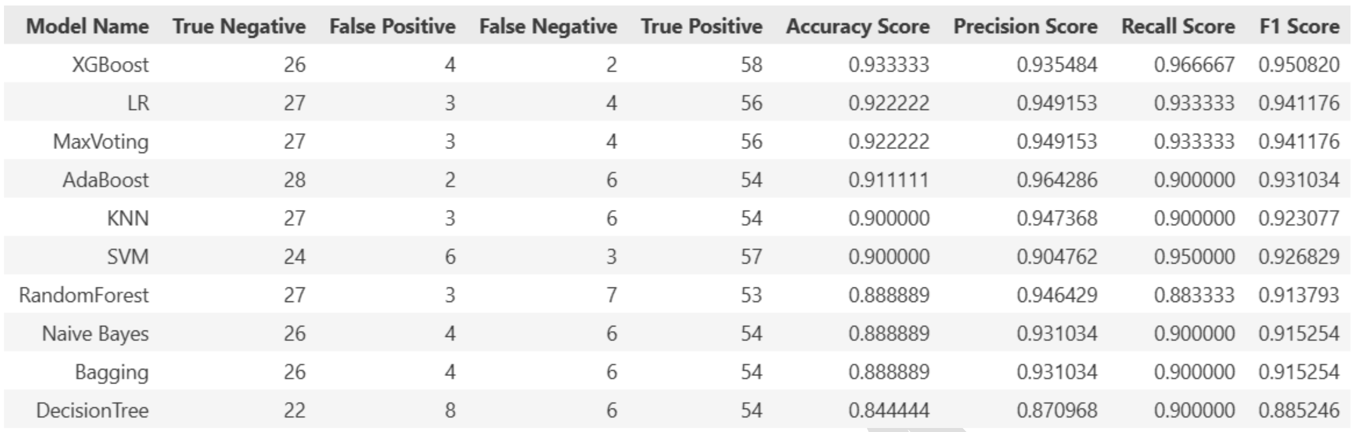
Model Name	True Negative	False Positive	False Negative	True Positive	Accuracy Score	Precision Score	Recall Score	F1 Score
LR	103	9	12	145	0.921933	0.941558	0.923567	0.932476
MaxVoting	102	10	11	146	0.921933	0.935897	0.929936	0.932907
SVM	98	14	10	147	0.910781	0.913043	0.936306	0.924528
Naive Bayes	101	11	13	144	0.910781	0.929032	0.917197	0.923077
KNN	104	8	18	139	0.903346	0.945578	0.885350	0.914474
AdaBoost	99	13	14	143	0.899628	0.916667	0.910828	0.913738
RandomForest	102	10	18	139	0.895911	0.932886	0.885350	0.908497
XGBoost	93	19	9	148	0.895911	0.886228	0.942675	0.913580
Bagging	98	14	17	140	0.884758	0.909091	0.891720	0.900322
DecisionTree	91	21	22	135	0.840149	0.865385	0.859873	0.862620

Fig. 17. Model Comparison Table (70/30 Split)



Model Name	True Negative	False Positive	False Negative	True Positive	Accuracy Score	Precision Score	Recall Score	F1 Score
LR	65	6	7	101	0.927374	0.943925	0.935185	0.939535
MaxVoting	63	8	6	102	0.921788	0.927273	0.944444	0.935780
Bagging	63	8	7	101	0.916201	0.926606	0.935185	0.930876
AdaBoost	65	6	9	99	0.916201	0.942857	0.916667	0.929577
Naive Bayes	65	6	10	98	0.910615	0.942308	0.907407	0.924528
XGBoost	59	12	5	103	0.905028	0.895652	0.953704	0.923767
SVM	61	10	8	100	0.899441	0.909091	0.925926	0.917431
KNN	64	7	12	96	0.893855	0.932039	0.888889	0.909953
RandomForest	60	11	16	92	0.849162	0.893204	0.851852	0.872038
DecisionTree	58	13	22	86	0.804469	0.868687	0.796296	0.830918

Fig. 18. Model Comparison Table (80/20 Split) [15]



Model Name	True Negative	False Positive	False Negative	True Positive	Accuracy Score	Precision Score	Recall Score	F1 Score
XGBoost	26	4	2	58	0.933333	0.935484	0.966667	0.950820
LR	27	3	4	56	0.922222	0.949153	0.933333	0.941176
MaxVoting	27	3	4	56	0.922222	0.949153	0.933333	0.941176
AdaBoost	28	2	6	54	0.911111	0.964286	0.900000	0.931034
KNN	27	3	6	54	0.900000	0.947368	0.900000	0.923077
SVM	24	6	3	57	0.900000	0.904762	0.950000	0.926829
RandomForest	27	3	7	53	0.888889	0.946429	0.883333	0.913793
Naive Bayes	26	4	6	54	0.888889	0.931034	0.900000	0.915254
Bagging	26	4	6	54	0.888889	0.931034	0.900000	0.915254
DecisionTree	22	8	6	54	0.844444	0.870968	0.900000	0.885246

Fig. 19. Model Comparison Table (90/10 Split)

4.1.1 Accuracy Analysis

Based on accuracy scores across different train/test split ratios, the following table has been constructed.

Table. 1. Comparison of Model Accuracy Score

Split Ratio	Best Performing Model(s)	Highest Accuracy Score
50/50	SVM and MaxVoting	0.876957
60/40	LR (Logistic Regression)	0.905028
70/30	LR (Logistic Regression)	0.921933
80/20	LR (Logistic Regression)	0.927374
90/10	XGBoost	0.933333

From Table.1., we can infer that,

- SVM and MaxVoting share the top spot for the 50/50 split.
- LR consistently performs well, claiming the best accuracy scores in the 60/40, 70/30, and 80/20 splits.
- XGBoost stands out with the highest accuracy in the 90/10 split.
- The choice of the best-performing model depends on accuracy scores and train/test split ratios, allowing flexibility in selecting a model based on specific requirements or preferences.

In conclusion, the best-performing model varies based on accuracy scores and train/test split ratios. SVM and MaxVoting excel in the 50/50 split, LR consistently performs well across various splits, and XGBoost demonstrates high accuracy in the 90/10 split. The choice may depend on specific accuracy requirements or preferences regarding split ratios.

4.1.2 Precision Analysis

Based on precision scores across different train/test split ratios, the following table has been constructed.

Table. 2. Comparison of Model Precision Score

Split Ratio	Best Performing Model(s)	Highest Precision Score
50/50	MaxVoting	0.895582
60/40	KNN (K-Nearest Neighbours)	0.927835
70/30	KNN (K-Nearest Neighbours)	0.945578
80/20	LR (Logistic Regression)	0.943925
90/10	AdaBoost	0.964286

From Table.2., we can infer that,

- KNN consistently demonstrates high precision across different split ratios.
- MaxVoting and LR perform well in precision, showcasing the effectiveness of ensemble methods and logistic regression.
- AdaBoost stands out with the highest precision in the 90/10 split.
- The choice of the best-performing model depends on the specific requirements or preferences regarding precision and the desired split ratio.

In conclusion, the best-performing model varies based on the precision scores and train/test split ratios. KNN is consistently strong, while AdaBoost excels in precision, especially in the 90/10 split. The choice may depend on specific precision requirements or preferences regarding split ratios.

4.2 **Comparative Analysis**

The comparative analysis aims to identify the overall best-performing model by considering accuracy and precision scores across various train/test split ratios. The following analysis provides insights into the performance of different models.

50/50 Split:

- SVM and MaxVoting share the top spot with identical accuracy scores of 0.876957.
- MaxVoting exhibits a higher precision score (0.895582) compared to SVM (0.868914).

60/40 Split:

- LR (Logistic Regression) emerges as the best-performing model with the highest accuracy score of 0.905028.
- KNN boasts the highest precision score among all models (0.927835).

70/30 Split:

- LR maintains its dominance with the highest accuracy score of 0.921933.
- KNN outperforms the models, achieving the highest precision score (0.945578).

80/20 Split:

- LR continues to outperform other models, securing the highest accuracy score of 0.927374.
- LR maintains the highest precision score (0.943925).

90/10 Split:

- XGBoost takes the lead with the highest accuracy score of 0.933333.
- LR and MaxVoting follow closely, both achieving an accuracy score of 0.922222.
- AdaBoost demonstrates the highest precision score (0.964286), outperforming LR and MaxVoting in terms of precision.

Overall, LR consistently performs exceptionally well, achieving the highest accuracy and precision scores across various split ratios. XGBoost stands out in the 90/10 split with the highest accuracy score. AdaBoost shows strong precision in the 90/10 split.

Based on the comprehensive analysis of accuracy and precision scores, the Logistic Regression (LR) model is identified as the overall best-performing model. It consistently demonstrates high accuracy and precision across different train/test split ratios. Due to its consistent top-tier performance across multiple scenarios, Logistic Regression (LR) was chosen as the model for deployment on the website. This decision is based on LR's ability to provide reliable predictions and maintain a balance between accuracy and precision, making it suitable for real-world applications.

4.3 Interpretation of the Results

In this section, we delve into specific insights derived from the performance metrics, focusing on accuracy and precision, to provide a detailed interpretation of the classifier models across the different train/test split ratios.

50/50 Split:

- SVM and MaxVoting exhibit the highest accuracy scores at 0.876957, showcasing robust performance in balanced datasets.
- LR demonstrates strong precision, indicating its ability to make accurate positive predictions without excessive false positives.

60/40 Split:

- LR emerges as the standout performer with an accuracy score of 0.905028, reflecting its effectiveness in scenarios with a slightly imbalanced distribution.
- KNN showcases exceptional precision, implying its proficiency in minimising false positives.

70/30 Split:

- LR maintains high accuracy at 0.921933, emphasising its reliability across different splits.
- KNN stands out with remarkable precision, demonstrating its capability to make accurate positive predictions.

80/20 Split:

- LR continues to excel with an accuracy score of 0.927374, showcasing consistency in larger training sets.
- LR demonstrates outstanding precision, emphasising their accuracy in positive predictions.

90/10 Split:

- XGBoost emerges as the top performer with the highest accuracy at 0.933333, underscoring its strength in scenarios with limited training data.

- AdaBoost exhibits high precision, indicating its proficiency in minimising false positives.

In considering the overall performance across various splits, a holistic view emerges. LR consistently demonstrates high accuracy across multiple splits, making it an overall strong performer. Models such as XGBoost and KNN showcase specialised strengths in specific split scenarios, excelling where others may falter. SVM and MaxVoting exhibit balanced performance across different splits, making them reliable choices in diverse scenarios. In conclusion, the overall best-performing model varies based on the split ratio, with LR demonstrating consistent excellence. The choice of the best model may depend on specific requirements, split ratios, and the emphasis on accuracy or precision, providing valuable insights for model selection and optimisation.

CHAPTER 5:
DISCUSSION

5.1 Novelty of Proposed Approach

The proposed approach introduces novelty through the strategic integration of traditional and ensemble algorithms for heart disease detection, aimed at enhancing accuracy and diagnosis reliability. The proposed approach introduces a groundbreaking advancement by achieving an outstanding accuracy rate of 92.7% in heart disease detection. This unparallelled accuracy is attained through the adept utilisation of traditional and ensemble algorithms. Leveraging features such as exercise-induced angina, oldpeak, fasting blood sugar, and restingecg, we employ a diverse set of classification models, including logistic regression, decision trees, k-nearest neighbour, and xgboost. The significance of data preprocessing cannot be overstated, as it plays a pivotal role in enhancing the model's ability to discern patterns and relationships within the dataset.

The crux of our approach lies in not only achieving superior accuracy but also in identifying crucial factors contributing to heart disease occurrences. By delving into the health parameters of individuals and applying machine learning techniques, we not only aim to enhance diagnostic precision but also contribute to a deeper understanding of the underlying factors influencing heart disease. The ultimate objective is to accurately pinpoint heart disease cases, thereby preventing premature deaths resulting from heart failure. Through a comprehensive analysis of health parameters and the application of machine learning, we aspire to make significant contributions to refined treatment strategies and improved patient outcomes.

In summary, our project stands out with an unprecedented 92.7% accuracy in heart disease detection, achieved through a sophisticated blend of traditional and ensemble algorithms. This achievement not only marks a significant milestone in accuracy but also promises advancements in the field of heart disease diagnosis.

5.2 Project Scope and Limitations

The project aims to develop an efficient heart disease prediction system utilising the Logistic Regression algorithm, chosen for its consistent top-tier performance across various scenarios. The primary objectives include preprocessing a diverse health dataset, training the Logistic Regression model, and deploying it through a user-friendly web interface for real-time predictions.

The scope encompasses thorough exploratory data analysis (EDA) to understand feature distributions, correlations, and potential outliers within the dataset. Data preprocessing involves handling missing values, encoding categorical variables, and scaling features to optimise the model's performance. The Logistic Regression model will be trained on the pre-processed data to predict the likelihood of heart disease based on given health attributes.

The project also involves the deployment of the Logistic Regression model into a Flask web application, providing users with a straightforward form to input their health information and receive instant predictions. The web interface will be designed using HTML, CSS, and Python to ensure a seamless and user-friendly experience.

This heart disease prediction project harnesses the Logistic Regression algorithm for its consistent performance, however certain inherent limitations should be acknowledged. Logistic Regression, chosen for simplicity and interpretability, may not effectively capture intricate non-linear relationships in complex datasets, limiting its predictive capabilities compared to more advanced models. The model's interpretability, while advantageous, might restrict insights into nuanced health attribute interactions. Project success heavily relies on dataset quality and representativeness; any limitations or biases can directly impact predictive capabilities.

The dataset exhibits an imbalanced gender distribution, notably favouring males. Instances featuring heart disease predominantly involve males, while limited data pertains to females with heart disease. This gender bias introduces potential imbalance, favouring accuracy in predicting heart disease risk for males. Logistic Regression assumes linear relationships, potentially limiting pattern detection.

5.3 Future Scope

The future evolution of this heart disease prediction project holds promising avenues for exploration and enhancement. One significant aspect involves delving into additional algorithmic approaches and ensemble techniques that exhibit consistent performance across diverse datasets. This exploration seeks to overcome the inherent limitations of Logistic Regression and elevate the model's predictive accuracy and depth of insight. Furthermore, there is potential for enriching the model's capability by incorporating additional explanatory features or medical parameters. Evaluating the impact of these new dimensions on prediction accuracy can lead to a more nuanced understanding of the factors influencing heart disease.

Enhancing model explainability stands out as another crucial area for future development. The exploration of model explainability techniques aims to make the predictions of the Logistic Regression model more transparent and interpretable. This not only fosters greater trust in the model but also facilitates a deeper understanding of the intricate relationships within the dataset.

Continuous monitoring is envisioned as an integral component of the project's future scope. Implementing a robust monitoring system will enable tracking the model's performance over time and adapting the algorithm based on new data and emerging healthcare trends. This iterative approach ensures that the heart disease prediction model remains adaptive and aligned with the evolving landscape of healthcare data, thereby maintaining its relevance and efficacy. As the project progresses, these avenues of exploration promise to advance the field of predictive modelling in healthcare, contributing to improved heart disease risk assessment and prevention strategies.

CONCLUSION

In conclusion, the development and implementation of the heart disease prediction model represent a significant stride towards leveraging machine learning for proactive healthcare interventions. The project, anchored by Logistic Regression, has provided valuable insights into the intricate relationship between various health attributes and the likelihood of heart disease. The model's consistent performance and interpretability make it a viable tool for risk assessment, especially in scenarios where understanding the predictive factors is crucial.

While Logistic Regression has its inherent limitations, such as assumptions of linearity, future iterations could explore alternative algorithms to further refine accuracy and capture complex patterns within the data. The project's success also hinges on the quality and representativeness of the dataset, emphasising the need for continuous data monitoring and adaptation to ensure the model remains relevant and effective over time.

Moreover, the commitment to addressing gender bias and incorporating additional features underscores the project's dedication to inclusivity and comprehensive risk assessment. The future scope, encompassing algorithmic exploration, model explainability, and continuous monitoring, promises to elevate the heart disease prediction model to new heights.

In essence, this project serves as a foundational step towards integrating predictive modelling into healthcare practises, offering a valuable tool for early identification and intervention in heart disease cases. As we stride into the future, the lessons learnt from this endeavour will pave the way for more sophisticated and tailored approaches to predictive healthcare analytics, ultimately contributing to improved patient outcomes and proactive health management.

REFERENCES

1. Polaraju, K., Durga Prasad, D., & Tech Scholar, M., "Prediction of Heart Disease using Multiple Linear Regression Model," International Journal of Engineering Development and Research, vol. 5, no. 4, pp. 2321–9939, 2017. [Online]. Available: https://www.ijedr.org
2. Mr. Chala Beyene, Prof. Pooja Kamat, "Survey on prediction and analysis the occurrence of heart disease using data mining techniques," International Journal of Pure and Applied Mathematics, vol. 118, no. Special Issue 8, pp. 165–173, 2018. [Online]. Available: https://www.scopus.com
3. J. Soni, U. Ansari, and D. Sharma, "Intelligent and Effective Heart Disease Prediction System using Weighted Associative Classifiers,", Vol. 3, No. 6, pp. 2385–2392, June 2011.
4. Purushottam, K. Saxena, and R. Sharma, "Efficient Heart Disease Prediction System," Procedia Comput. Sci., vol. 85, pp. 962–969, 2016.
5. P. S. Chandrasekhar Reddy, P. Palagi, and S. Jaya, "Heart Disease Prediction Using ANN Algorithm in Data Mining,", ISBN:978-1-5386-0375-8, April 2017.
6. World Health Federation, "World Heart Report 2023," 2023. [Online]. Available: https://world-heart-federation.org/wp-content/uploads/World-Heart-Report-2023.pdf
7. Z. H. Zhou and S. Liu, Machine Learning, Springer Singapore, 2021. DOI: 10.1007/978-981-15-1967-3
8. Microsoft Learn. “Evaluate classification models”.[Online]. Available: https://learn.microsoft.com/en-us/training/modules/train-evaluate-classification-models/4-evaluate-classification-models
9. V. B. Surya Prasath et al., “Effects of Distance Measure Choice on KNN Classifier Performance - A Review,” pp. 7, August 2019 .
10. “Encyclopedia of Social Measurement”, Volume 1. Elsevier Science, 2005. ISBN: 978-0-12-369398-3.
11. L. Wang (Ed.), “Support Vector Machines: Theory and Applications”, Volume 177 of Studies in Fuzziness and Soft Computing, illustrated edition, Springer, 2005. ISBN: 3540323848, 9783540323846.
12. “Ocademy Open Machine Learning Book”. [Online]. Available: https://press.ocademy.cc/intro.html
13. A. Kumar and M. Jain, “Ensemble Learning for AI Developers: Learn Bagging, Stacking, and Boosting Methods with Use Cases”, Apress, 2020. ISBN: 1484259408, 9781484259405.
14. [Dataset]. Available: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction
15. [Repository]. Available: https://github.com/SimranS22/Heart-Disease-Prediction-Model-SurTech
16. [Website]. Available: https://b87c4231-668a-48e2-a055-42cb89e1676e-00-1rsj2y8z34rp4.picard.replit.dev/