HEART DISEASE PREDICTION USING MACHINE LEARNING

A PROJECT REPORT

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

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

BY

N.S.S HARSHITH	**K . DEVARAJU**
(20331A05C8)	**(20331A0592)**
K. YESWANTH TEJA	**G. VASU DEVARAJU**
(20331A0581)	**(20331A0567)**

Under the Supervision of
Dr. T. PAVAN KUMAR
Associate Professor, HOD CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAHARAJ VIJAYARAM GAJAPATHI RAJ COLLEGE OF ENGINEERING
(Autonomous)
(Approved by AICTE, New Delhi, and permanently affiliated to JNTUGV, Vizianagaram), Listed u/s 2(f) & 12(B) of UGC Act 1956.
Vijayaram Nagar Campus, Chintalavalasa, Vizianagaram-535005, Andhra Pradesh

APRIL, 2024

CERTIFICATE

This is to certify that the project report entitled "**HEART DISEASE PREDICTION USING MACHINE LEARNING**" being submitted by N.S.S Harshith (20331A05C8), K. Devaraju (20331A0592), K. Yeswanth Teja (20331A0581), G. Vasu Devaraju (20331A0567) in partial fulfillment for the award of the degree of "**Bachelor of Technology" in Computer Science and Engineering** is a record of bonafide work done by them under my supervision during the academic year 2023-2024.

Dr. T. Pavan Kumar
Associate Professor, HOD CSE
Supervisor,
Department of CSE,
MVGR College of Engineering(A),
Vizianagaram.

Dr. T. Pavan Kumar
Associate Professor,
Head of the Department,
Department of CSE,
MVGR College of Engineering(A),
Vizianagaram.

External Examiner

DECLARATION

We hereby declare that the work done on the dissertation entitled "**HEART DISEASE PREDICTION USING MACHINE LEARNING**" has been carried out by us and submitted in partial fulfillment for the award of credits in Bachelor of Technology in Computer Science and Engineering of MVGR College of Engineering (Autonomous) and affiliated to JNTUGV, Vizianagaram. The various contents incorporated in the dissertation have not been submitted for the award of any degree from any other institution or university.

ACKNOWLEDGEMENTS

We express our sincere gratitude to **Dr. T. Pavan Kumar** for his invaluable guidance and support as our mentor throughout the project. His unwavering commitment to excellence and constructive feedback motivated us to achieve our project goals. We are greatly indebted to him for his exceptional guidance.

Additionally, we extend our thanks to **Prof. P.S. Sitharama Raju (Director)**, **Prof. Ramakrishnan Ramesh (Principal)**, and **Dr. T. Pavan Kumar (Head of the Department)** for their unwavering support and assistance, which were instrumental in the successful completion of the project. We also acknowledge the dedicated assistance provided by all the staff members in the Department of Computer Science & Engineering. Finally, we appreciate the contributions of all those who directly or indirectly contributed to the successful execution of this endeavor.

N.S.S Harshith (**20331A05C8)**
K. Devaraju (**20331A0592)**
K. Yeswanth Teja (**20331A0581)**
G. Vasu Devaraju (**20331A0567)**

LAST MILE EXPERIENCE (LME)

PROJECT TITLE
HEART DISEASE PREDICTION USING MACHINE LEARNING

BATCH NUMBER – 10B

BATCH SIZE – 4

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Name:
N.S.S Harshith
Email:
nss.harshith02@gmail.com
Contact Number:
9705878597

Name:
K. Devaraju
Email:
devaraj20112002@gmail.co m
Contact Number:
9652060374

Name:
K. Yeswanth Teja
Email:
tejacherry0611@gmail.com
Contact Number:
8074121998

Name:
G. Vasu Devaraju
Email:
vasuvpk2330@gmail.com
Contact Number:
7075673186

Project Supervisor

Name:
Dr.T. Pavan Kumar
Designation:
Assistant Professor, H.O.D CSE
Email:
pavank3400@gmail.com
Contact Number:
9515687858

Project Objectives

1. **Develop Accurate predictive model**
2. **Reduce misdiagnosis rates**
3. **Enhance treatment planning.**
4. **Contribute to medical research and practice.**

Project Outcomes

The overall outcome of the cardiovascular disease prediction project is to develop accurate and reliable machine learning models that can effectively predict the occurrence of cardiovascular diseases based on patient data and medical history. By achieving this outcome, the project aims to:

Improve early detection and prevention

Reduce misdiagnosis rates

Enhance treatment planning.

Contribute to medical research and practice.

Domain of Specialisation

MACHINE LEARNING

How is your solution helping the domains?

- Improve patient care
- Enhance healthcare resource allocation
- Advancement in medical research
- Empowerment of healthcare practitioners
- Integration of machine learning in healthcare

List the Program Outcomes (POs) that are being met by doing the project work

PO9: Individual and teamwork	Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.
PO10: Communication	Communicate effectively on complex engineering activities with the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11: Project management and finance	Demonstrate knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work as a member and leader in a team, to manage projects and in multi-disciplinary environments.
PO12: Life-long learning	Recognize the need for, and have the preparation and ability to engage in, independent and life-long learning

End Users of Your Solution

- Healthcare professionals
- Healthcare administrators and policymakers
- Medical researchers and patients

ABSTRACT

The heart is the central part of the circulatory system and pumps blood throughout the body. According to the World Health Organization, 620 million of 8 billion people suffered from cardiovascular diseases in 2023. Examination and diagnosis of heart disease is an important task in medicine. It allows for accurate classification and helps cardiologists provide appropriate treatment to patients. The use of machine learning in medicine is increasing because it can identify patterns in data. Using machine learning to determine the occurrence of heart disease could help doctors reduce the number of misdiagnoses. In this study, we developed a model to predict cardiovascular disease to reduce deaths from cardiovascular disease.in this paper, we introduced a k-type clustering method that can improve classification accuracy. Machine learning methods Random Forest (RF), Decision Tree (DT) classifier, Multilayer Perceptron (MLP), and XG Boost (XGB) are utilized by GridSearchCV.

It is used to tune parameters to improve results. The proposed model was implemented on a real dataset consisting of 70,000 samples obtained from Kaggle. Data was split in 80:20 ratio and got the following accuracy: Decision Tree: 82.69% (with cross-validation) and 83.93%(without cross-validation), XGBoost: 83.31% (with cross-validation) and 84.46%(without cross-validation), Random Forest: 84.47% (with cross-validation) and 84.13% (without cross-validation), Multilayer Perceptron: 84.23% (with cross-validation) and 84.19% (without cross-validation). The results of this simple study in that the cross-validated random forest outperforms all other algorithms in accuracy. The highest accuracy reaches 84.47%.

CONTENTS

List of Abbreviations

WHO	–	World Health Organization
RF	–	Random Forest
DT	–	Decision Tree
XGBoost	–	Extreme Gradient Boosting
MLP	–	Multilayer Perceptron
CV	–	Cross-validation
CVD	–	Cardiovascular diseases
CT Scans	–	Computed Tomography Scans
USD	–	United States Dollar
ML	–	Machine Learning
MAFLD	–	Metabolic-Associated Fatty Liver Disease
AUC	–	Area Under the Curve
KKN	–	K-Nearest Neighbor
SVM	–	Support Vector Machine
UCI	–	University of California, Irvine
FRS	–	Framingham Risk Score
BMI	–	Body Mass Index
MAP	–	Mean Arterial Pressure
WCSS	–	Within-Cluster Sum of Squares
TP	–	True Positive
TN	–	True Negative
FP	–	False Positive
FN	–	False Negative
ap_hi	–	Systolic Blood Pressure
ap_lo	–	Diastolic Blood Pressure
k-means	–	k-means Clustering Algorithm
k-modes	–	k-modes Clustering Algorithm
F-measure	–	F1 Score
IQR	–	Interquartile Range
MSE	–	Mean Squared Error

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

A well-maintained heart not only supports the body's efficient operation but also represents a source of strength and liveliness, fundamental for a rewarding life journey. Globally, cardiovascular disease (CVDs) remains the leading cause of both morbidity and mortality, constituting over 70% of all deaths. According to research from 2017 on the Global Burden of Disease, CVDs are responsible for approximately 43% of all fatalities[1]. While high-income nations commonly face risk factors such as poor diet, smoking, excessive sugar intake, and obesity, low- and middle-income countries are also experiencing a surge in chronic illnesses[3]. The economic burden of cardiovascular diseases worldwide was projected to reach around USD 3.7 trillion between 2010 and 2015[5].

CVD accounts for over 70% of global deaths, with unhealthy diets, smoking, and obesity being common risk factors. Low and middle-income countries are also seeing increased rates of heart disease. Unfortunately, diagnostic tools like electrocardiograms and CT Scans are often too expensive for many, and this reason alone has resulted in the deaths of 17 million people[4]. 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[6].

Machine learning plays a crucial role in the medical field. Using machine learning, we can diagnose, detect, and predict various diseases. Recently, there has been a growing interest in using data mining and machine learning techniques to predict the likelihood of developing certain diseases[7]. The main goal of this paper is to accurately predict the possibility of heart disease in the human body.

Our project aims to predict individuals at risk of heart disease using their medical history. By analyzing factors like age, gender, cholesterol levels, glucose, smoking, activity, height, weight, etc, we can identify those likely to develop heart problems. We utilized advanced data mining techniques like Decision Trees, Random Forest[9], XGBoost[10], and Multi-value Perception to improve prediction accuracy, achieving an 84.47% success rate.

To increase the model performance, we preprocess and scale the dataset using k-mode clustering. The dataset used in our study is publicly available using Python on Google Colab. While previous studies have reported high accuracy rates in heart disease prediction, many have used small sample sizes, limiting generalizability. Our project is done using a larger dataset of 70,000 instances, aiming to produce more widely applicable results.

CHAPTER 2
LITERATURE SURVEY

In recent years, the healthcare sector has witnessed significant progress in leveraging data mining and machine learning techniques. These methods have gained widespread acceptance and have proven effective across various healthcare domains, notably in medical cardiology. The substantial accumulation of medical data has opened up unprecedented opportunities for researchers to devise and evaluate novel algorithms in this arena. Heart disease continues to be a leading cause of mortality, particularly in developing countries. Identifying risk factors and early indicators of the disease has emerged as a crucial area of investigation. The integration of data mining and machine learning approaches holds promise in facilitating early detection and prevention strategies for heart disease.

2.1. Drod et al. (2022):

Used machine learning techniques to identify significant risk variables for cardiovascular disease in patients with metabolic-associated fatty liver disease (MAFLD).Conducted blood biochemical analysis and subclinical atherosclerosis assessment on 191 MAFLD patients.Built a model to identify high-risk CVD patients using multiple logistic regression classifiers, univariate feature ranking, and principal component analysis (PCA). Achieved an AUC of 0.87 and correctly identified 85.11% of high-risk patients and 79.17% of low-risk patients[2].

2.2. Shah et al. (2020):

Aimed to develop a model for predicting cardiovascular disease using machine learning techniques.Employed data from the Cleveland heart disease dataset, consisting of 303 instances and 17 attributes.Tested various supervised classification methods, including naive Bayes, decision tree, random forest, and k-nearest neighbor (KKN).Found that the KKN model exhibited the highest accuracy of 90.8% in predicting cardiovascular disease[12].

2.3. Hasan and Bao (2020):

Conducted a study to identify the most efficient feature selection approach for anticipating cardiovascular illness. Compare multiple feature selection methods (filter, wrapper, and embedding) and machine learning algorithms. Utilized various models, including random forest, support vector classifier, k-nearest neighbors, naive Bayes, and XGBoost.Found that the XGBoost classifier coupled with the wrapper technique provided the most accurate prediction results for cardiovascular illness, with an accuracy of 73.74%[14].

2.4. Alotalibi (2019):

Investigated the utility of machine learning techniques for predicting heart failure disease.Utilized a dataset from the Cleveland Clinic Foundation.Implemented various ML algorithms, including decision tree, logistic regression, random forest, naive Bayes, and support vector machine (SVM).Found that the decision tree algorithm achieved the highest accuracy of 93.19% in predicting heart disease[13].

2.5. Narain et al. (2016):

Developed a machine-learning-based cardiovascular disease (CVD) prediction system.Utilized a quantum neural network to improve upon the widely used Framingham risk score (FRS). Experimentally validated the proposed system with data from 689 individuals and a validation dataset from the Framingham research. Achieved an accuracy of 98.57% in forecasting CVD risk, surpassing existing techniques [11].

Table 1.1. Related work on heart disease prediction using large datasets.

Authors	**Novel Approach**	**Best Accuracy**	**Dataset**
Shorewall, 2021[4]	Stacking of KNN, random forest, and SVM outputs with logistic regression as the metaclassifier	75.1% (stacked model)	Kaggle cardiovascular disease dataset (70,000 patients, 12 attributes)
Our and ElSeddawy, 2021[15]	Repeated random with random forest	89.01%(random forest classifier)	UCI cardiovascular dataset (303 patients, 14 attributes)
Waigi et al., 2020[8]	Decision tree	72.77% (decision tree)	Kaggle cardiovascular disease dataset (70,000 patients, 12 attributes)
Khan and Mondal, 2020[16]	Cross-validation method with logistic regression (solver: lbfgs) where k = 30	72.72%	Kaggle cardiovascular disease dataset 1 (462 patients, 12 attributes)

Maiga et al., 2019[26]	-Random forest-Naive Bayes-Logistic regression-KNN	70%	Kaggle cardiovascular disease dataset (70,000 patients, 12 attributes)

CHAPTER 3
THEORETICAL BACKGROUND

3.1. Cardiovascular Disease Diagnosis and Prognosis: Cardiovascular diseases encompass a range of conditions affecting the heart and blood vessels, including coronary artery disease, heart failure, and stroke. Diagnosis and prognosis involve identifying the presence of these conditions, assessing their severity, and predicting their future course. Cardiologists rely on a combination of patient history, physical examination, laboratory tests, and imaging studies (such as electrocardiograms and echocardiograms) to make accurate diagnoses and prognostic assessments.

3.2. Machine Learning in Medicine: Machine learning algorithms have revolutionized the field of medicine by enabling computers to learn from data and make predictions or decisions without being explicitly programmed. In the context of cardiovascular disease, machine learning techniques can analyze large datasets of patient information to identify patterns and relationships that may not be apparent to human observers. These techniques can aid in risk stratification, early detection of disease, treatment planning, and outcome prediction.

3.3. Pattern Recognition and Data Mining: Data mining involves extracting valuable insights and knowledge from large datasets. In this project, data mining techniques are applied to identify patterns or clusters within the dataset that may be indicative of cardiovascular disease. For example, k-modes clustering with Huang starting is used to partition the data into groups based on similarities in patient characteristics. This helps to improve the accuracy of classification by grouping instances with similar features.

3.4. BMI (Body Mass Index): BMI is a measure commonly used to assess a person's body weight relative to their height. It is calculated by dividing a person's weight in kilograms by the square of their height in meters (BMI = weight / (height)^2). The resulting value is expressed in units of kg/m^2. BMI is often used as a screening tool to classify individuals such as underweight, normal weight, overweight, or obese. While BMI does not directly measure body fat, it is correlated with body fat percentage and serves as a convenient and inexpensive method for assessing weight status and potential health risks associated with weight.

Underweight: BMI less than 18.5.

Normal weight: BMI 18.5 to 24.9

Overweight: BMI 25 to 29.9

Obesity Class I (Moderate): BMI 30 to 34.9

Obesity Class II (Severe): BMI 35 to 39.9

Obesity Class III (Very Severe or Morbid Obesity): BMI 40 or greater

3.5. MAP (Mean Arterial Pressure): MAP is a calculated value that represents the average blood pressure in the arteries during a single cardiac cycle. It is an important parameter in hemodynamics and reflects the perfusion pressure required to deliver oxygen and nutrients to vital organs. MAP is typically calculated using the formula: MAP = (2/3 * Diastolic BP) + (1/3 * Systolic BP). Diastolic blood pressure (DBP) represents the pressure in the arteries when the heart is at rest between beats, while systolic blood pressure (SBP) represents the pressure during a heartbeat when the heart contracts and pumps blood into the arteries. MAP provides a more accurate estimate of perfusion pressure than systolic or diastolic blood pressure alone and is often used in clinical settings to guide treatment decisions for conditions such as hypertension and shock.

HypoTension Crisis: Below 70

HypoTension : 70 to 80

Normal: 80 to 90

Elevated/Pre HyperTension: 90 to 100

Hyper Tension Stage I: 100 to 110

Hyper Tension StageII: 110 to 120

Hyper Tension Crisis: Above 120

3.6. Model Selection and Hyperparameter Tuning: Choosing the right machine learning model and optimizing its parameters are crucial steps in developing an effective predictive model. This project evaluates several popular machine learning algorithms, including random forest, decision tree classifier, multilayer perceptron, and XGBoost. Hyperparameter tuning techniques, such as GridSearchCV, are employed to systematically search for the best combination of model parameters that maximize predictive performance.

3.7. Binning: Binning, also known as discretization, is a data preprocessing technique used to transform continuous numerical variables into discrete categorical bins or intervals. This process involves dividing the range of the variable into a predetermined number of intervals and assigning each data point to its corresponding bin based on its value. Binning can help simplify complex datasets, reduce noise, and improve the performance of certain machine-learning algorithms by making the data more manageable and interpretable. Common methods of binning include equal-width binning (where bins have the same width) and equal-frequency binning (where bins have approximately the same number of data points).

3.8. Machine Learning Models

3.8.1. Random Forest (RF): Random Forest is an ensemble learning method that constructs multiple decision trees during training. Each tree in the forest is trained on a random subset of the training data, and the final prediction is made by averaging the predictions of all individual trees (for regression tasks) or by taking a majority vote (for classification tasks). Random Forest is known for its robustness to overfitting, high accuracy, and ability to handle large datasets with high dimensionality.

3.8.2. Decision Tree Classifier (DT): Decision Tree Classifier is a non-parametric supervised learning method used for classification tasks. It recursively splits the dataset into subsets based on the most significant attribute, resulting in a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents a class label. Decision trees are easy to interpret and can handle both numerical and categorical data, but they are prone to overfitting, especially with complex datasets.

3.8.3. Multilayer Perceptron (MP): Multilayer Perceptron is a type of artificial neural network characterized by multiple layers of interconnected neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron applies a non-linear activation function to the weighted sum of its inputs, allowing the network to learn complex patterns in the data. Multilayer Perceptron is effective for tasks involving non-linear relationships and can approximate any continuous function given enough hidden units.

3.8.4. XGBoost (XGB): XGBoost, short for Extreme Gradient Boosting, is an optimized implementation of gradient boosting machines, which are ensemble learning methods that build a series of weak learners (typically decision trees) sequentially. XGBoost combines the advantages of gradient boosting with several regularization techniques to prevent overfitting and improve predictive performance. It employs a scalable and efficient algorithm that is widely used for structured/tabular data and has won numerous machine-learning competitions for its high accuracy and speed.

3.9. GridSearchCV: GridSearchCV is a method for hyperparameter tuning in machine learning, particularly for algorithms that have hyperparameters that need to be optimized for optimal performance. It is part of the sci-kit-learn library in Python. GridSearchCV systematically searches through a predefined grid of hyperparameters for a specified machine-learning model. For each combination of hyperparameters, GridSearchCV performs cross-validation to evaluate the model's performance using a scoring metric (e.g., accuracy, AUC). The combination of hyperparameters that yields the best performance on the validation set is selected as the optimal set of hyperparameters for the model. GridSearchCV helps automate the process of hyperparameter tuning and can improve the generalization and performance of machine learning models.

3.10. Elbow Curve method: The elbow curve method is a technique used to determine the optimal number of clusters in a dataset for clustering algorithms like k-means. By plotting the within-cluster sum of squares (WCSS) against the number of clusters and identifying the "elbow" point where the rate of decrease in WCSS slows down significantly, practitioners can select the appropriate number of clusters. This point represents a trade-off between clustering performance and complexity, helping to achieve meaningful and interpretable clusters in the data.

3.11. Confusion matrix: A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. All the measures can be calculated by using left most four parameters. So, let's talk about those four parameters first.

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Figure 1: Confusion matrix with 4 parameters

True positives and true negatives are the observations that are correctly predicted. We want to minimize false positives and false negatives. These terms are a bit confusing. So, let's take each term one by one and understand it fully.

True Positives (TP) - These are the correctly predicted positive values which means that the value of the actual class is yes and the value of the predicted class is also yes.

True Negatives (TN) - These are the correctly predicted negative values which means that the value of the actual class is no and the value of the predicted class is also no.

False positives and false negative, these values occur when your actual class contradicts with the predicted class.

False Positives (FP) – When the actual class is no and the predicted class is yes.

False Negatives (FN) – When the actual class is yes but the predicted class is no.

3.12. Evaluation Metrics

3.12.1. Accuracy: Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observations to the total observations.

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

3.12.2. Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision = TP/TP+FP

3.12.3. Recall/ Sensitivity/ True Positive Rate (TPR): Recall is the ratio of correctly predicted positive observations to all observations in actual class - yes.

Recall = TP/TP+FN

3.12.4. F1 Score: The F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. F1 Score = 2*(Recall * Precision) / (Recall + Precision)**.**

CHAPTER 4
APPROACH DESCRIPTION

4.1. Data Collection:

- Gather a diverse and representative dataset that includes relevant patient information. This data is obtained from an open source available in Kaggle.
- The dataset includes a mix of demographic information (age, gender), medical history (past heart-related issues), lifestyle factors (smoking, exercise), and clinical measurements (blood pressure, cholesterol levels, etc.).

4.2. Data Preprocessing:

- Clean the dataset by handling missing values, outliers, and data inconsistencies. This may involve imputing missing values, normalizing or standardizing features, and addressing data imbalances if present.
- This step involves identifying and removing the outliers, null values, and missing values with the help of boxplots.

4.3. Feature Selection and Engineering:

- Analyze the dataset to identify the most relevant features for predicting heart disease. Feature selection techniques like correlation analysis, feature importance from tree-based models, or domain expertise can help in choosing the right features.
- Engineer new features if necessary, such as BMI (Body Mass Index) calculated from height and weight and MAP(mean arterial pressure) calculated from ap_hi and ap_lo.

4.4. Clustering:

- Clustering is a machine-learning technique used to group instances based on similarity measures.
- While k-means is a common algorithm for clustering, it's ineffective with categorical data. The k-modes algorithm, introduced by Huang in 1997, addresses this limitation by using dissimilarity measures for categorical data and replacing means with modes. This makes it effective with categorical data.

4.5. Modelling:

- A training dataset (80%) and a testing dataset (20%) are created from the dataset. A model is trained using the training dataset, and its performance is assessed using the testing dataset.
- Different classifiers, such as decision tree classifier, random forest classifier, multilayer perceptron, and XGBoost, are applied to the clustered dataset to assess

their performance. The performance of each classifier is then evaluated using accuracy, precision, recall, and F-measure scores.

4.6. Model Training:

- Train the selected models using the training dataset. During training, the model learns the patterns and relationships in the data that are indicative of heart disease.

4.7. Hyperparameter Tuning:

- Optimize the hyperparameters of the selected models using techniques like grid search or random search. This step helps in finding the best configuration for each model.

4.8. Model Evaluation:

- Evaluate the models' performance using the validation dataset. Common evaluation metrics for classification tasks include accuracy, precision, recall, and F1-score.
- Use cross-validation to assess model stability and generalization.

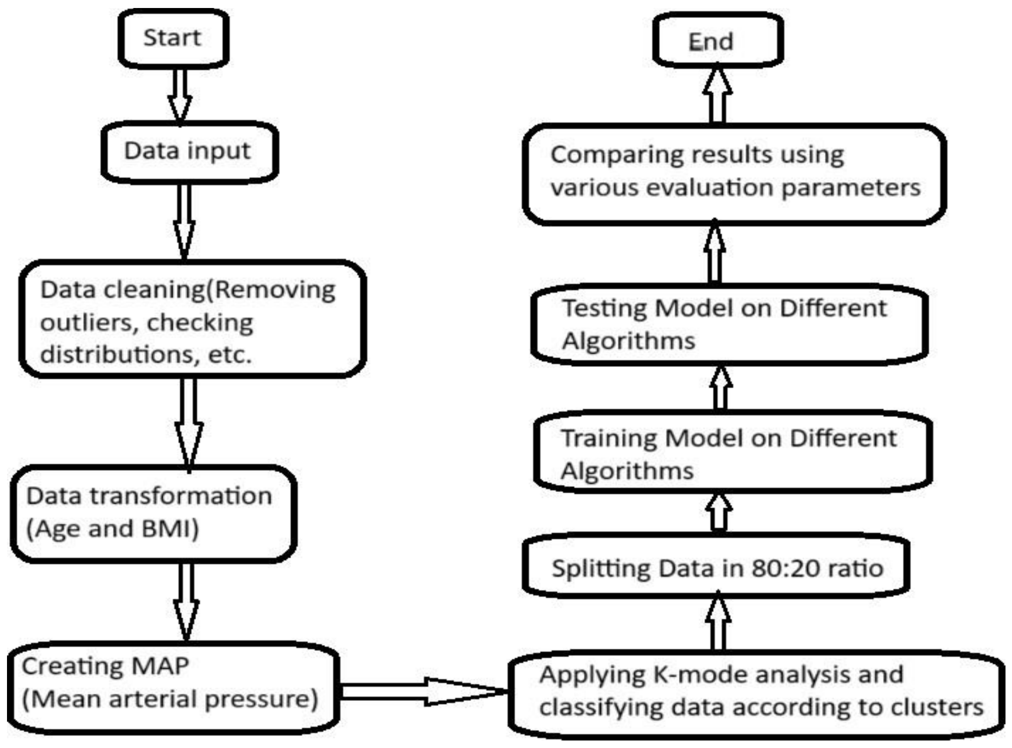

Start
Data input
Data cleaning(Removing outliers, checking distributions, etc.
Data transformation (Age and BMI)
Creating MAP (Mean arterial pressure)
Applying K-mode analysis and classifying data according to clusters
Splitting Data in 80:20 ratio
Training Model on Different Algorithms
Testing Model on Different Algorithms
Comparing results using various evaluation parameters
End

Figure 2. Flow diagram of Model

CHAPTER 5
DATA EXPLORATION

5.1 Data Source

The dataset utilized in this study[17] consists of 70,000 patient records, surrounded by 12 distinct features outlined in Table 2. These features involve metrics like age,gender, Height, Weight, systolic blood pressure, diastolic blood pressure, cholesterol, Smoking, Physical Activities, Glucose level, and Alcohol. The target class, labeled "cardio," distinguishes between patients with cardiovascular disease (coded as 1) and those without (coded as 0).

Table 2. Dataset attributes.

Feature	Variable	Min and Max Values
Age	age	Min: 10,798 and Max: 23,713
Height	height	Min: 55 and Max: 250
Weight	weight	Min: 10 and Max: 200
Gender	gender	1: female and 2: male
Systolic blood pressure	ap_hi	Min: -150 and Max: 16,020
Diastolic blood pressure	ap_lo	Min: -70 and Max: 11,000
Cholesterol	cholesterol	Categorical value=1(min) to 3(max)
Glucose	gluc	Categorical value=1(min) to 3(max)
Smoking	smoke	1: yes and 0: no
Alcohol intake	alco	1: yes and 0: no
Physical activity	active	1: yes and 0: no
Presence or absence of cardiovascular disease	cardio	1: yes and 0: no

5.2 Data Cleaning

5.2.1 Identifying outliers

The provided code aims to identify outliers within a data frame using boxplots. It begins by determining the number of rows and columns needed to display all the features in the DataFrame, with each subplot accommodating up to 12 columns. Subsequently, it creates subplots, with each subplot containing a boxplot for a specific feature in the data frame. By visualizing the distribution of each feature using boxplots, outliers can be identified based on their deviation from the bulk of the data. Outliers typically appear as individual points or clusters outside the whiskers of the boxplot. This approach facilitates the identification and examination of potential outliers across multiple features simultaneously, aiding in data exploration and anomaly detection.

Here, the presence of outliers in the dataset is evident, which are observed in "height", "weight", "ap_hi" and "ap_lo". These outliers may have been the result of errors in data entry. The removal of these outliers has the potential to improve the performance of our predictive model.

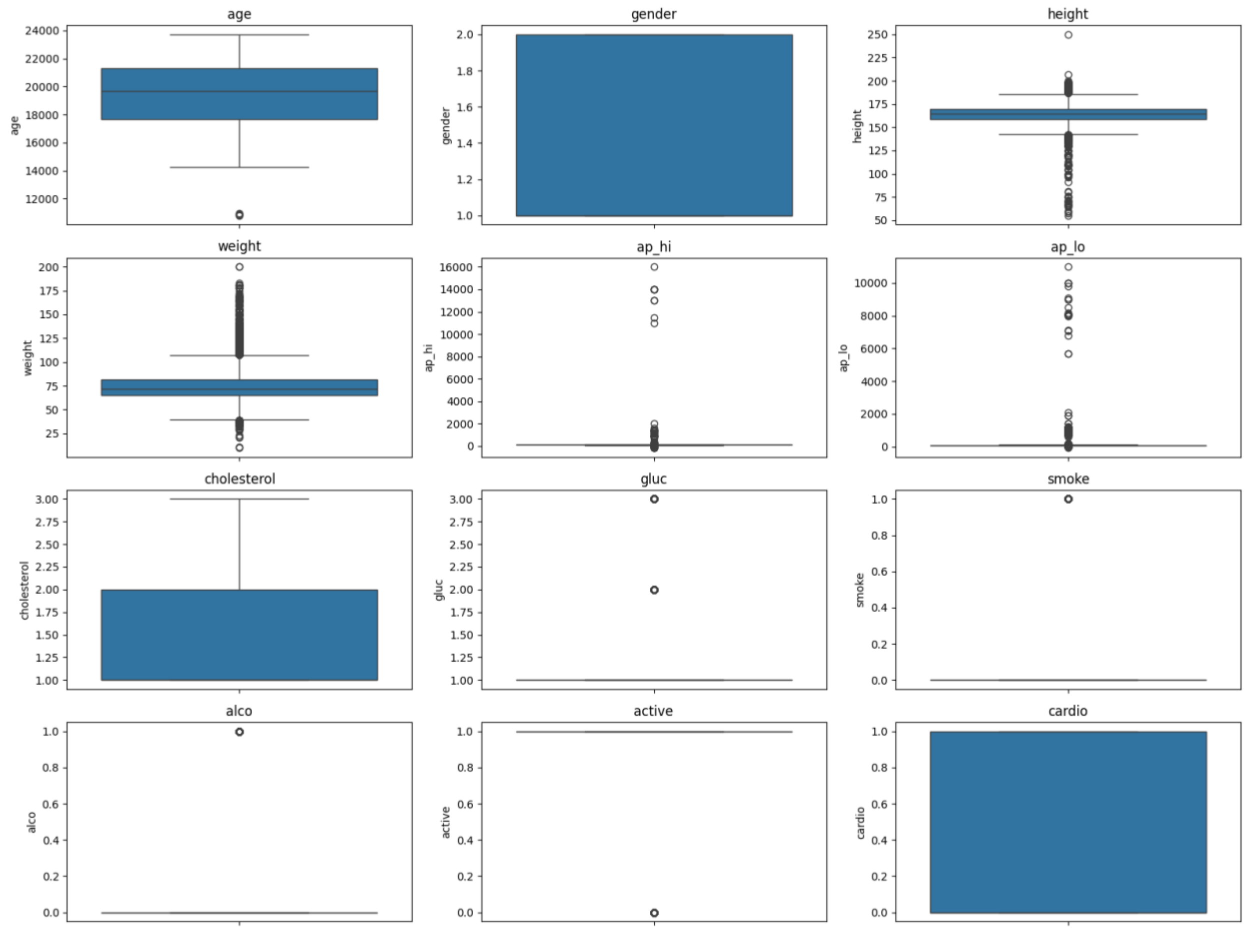


Figure 3. Identifying outliers through boxplots

5.2.2 Removing outliers

In the project, an essential aspect of data preprocessing involved the computation of statistical information for four key attributes: height, weight, systolic blood pressure (ap_hi), and diastolic blood pressure (ap_lo). This information was crucial for understanding the distribution of these attributes within the dataset. The function `statistic_Info` was designed to calculate significant statistical measures such as quartiles and interquartile range (IQR), which are fundamental in outlier detection.

Following the calculation of statistical measures, outliers were identified based on the IQR method. These outliers represent data points that significantly deviate from the typical distribution of the attribute values. In this project, a total of 7,495 outliers were detected across the dataset, indicating potential anomalies or erroneous data entries.

To ensure the robustness and reliability of subsequent analyses, these outliers were systematically removed from the dataset. This data cleaning process aimed to improve the quality and integrity of the dataset by eliminating potentially misleading or erroneous data points. After the removal of outliers, the dataset consisted of 62,505 rows, providing a more accurate representation of the underlying data distribution.

By conducting thorough data preprocessing, including outlier detection and removal, the project ensured that subsequent analyses and modeling tasks were based on a cleaner and more reliable dataset. This approach enhances the validity and trustworthiness of the project's findings and conclusions, ultimately contributing to more accurate insights into the factors influencing cardiovascular health.

5.3 Missing Values and Null Values

We should check whether there are missing values or null values in any of the columns and should handle them properly. Generally missing values are either filled with random values leading to less accuracy or they are ignored which is not good if half of the data contains missing values. The efficient way to fill these missing values is by finding the correlation between the other attributes. missing values and null values are common challenges in data analysis, requiring careful handling to ensure the integrity and accuracy of results. Effective strategies for dealing with missing and null values are essential to derive meaningful insights and make reliable decisions based on the data.

CHAPTER 6
DATA TRANSFORMATION

6.1. Feature Selection and Reduction

In the phase of feature selection and reduction, we believed in the utilization of binning as a technique to transform continuous input variables, such as age, into categorical inputs. Binning is the process of transforming numerical variables into their categorical data. This approach aims to enhance both the performance and understandability of the algorithms. Bycategorizing continuous variables into distinct bins or groups, the algorithm can effectively differentiate between various data classes based on specific values of the input variables. Moreover, converting continuous inputs into categorical ones through binning can facilitate the interpretation of results, as it simplifies the understanding of the relationship between input variables and output classes. In contrast, utilizing continuous inputs, such as numericalvalues, in classification algorithms can pose challenges, as the algorithm may struggle to determine optimal boundaries between different classes or categories[18].

In this analysis, we implemented binning on the age attribute within a patient dataset. Initially recorded in days, the patient ages were transformed into years for better analysis and prediction by dividing by 365. The age data were later on segmented into 5-year bins, spanning from 0–20 to 95–100. With the dataset's minimum age being 30 years and the maximum 65,the bins were labeled accordingly, starting from 0 for the 30–35 bin and ending at 6 for the 60–65 bin.

Furthermore, continuous attributes such as height, weight, ap_hi, and ap_lo were alsoconverted into categorical values. In our research, data exhibited a significant lifetime risk fordeveloping CVD, particularly among those classified as overweight or obese. Compared to individuals with a normal BMI, obese individuals showed an earlier onset of incident CVD[19]. This underscores the potential value of converting attributes like height and weight into **body mass index (BMI)** to enhance the performance of our heart disease prediction model.Subsequently, BMI values were categorized for further analysis.

$$\text{BMI} = \frac{\text{weight (kg/lb)}}{\text{height}^2\left(\text{m}^2/\text{in}^2\right)}$$

In the medical field, **mean arterial pressure (MAP)** denotes the average blood pressure experienced during a single cardiac cycle. A typical mean arterial pressure (MAP) ranges from 70 to 100 mmHg. When the MAP falls below 60 mmHg, there arises a worry regarding the adequacy of pressure to perfuse critical organs such as the brain. Conversely, if the MAP exceeds 100 mmHg, it may indicate elevated arterial pressure in the patient. Research involving individuals with type 2 diabetes revealed a direct correlation between MAP and CVDrisk, with a 13% increase in risk observed for every 13 mmHg rise in MAP. Moreover, elevatedMAP levels were linked to a higher incidence of CVD hospitalizations among individuals with type 2 diabetes, further underscoring the connection between MAP and CVD[20]. Mean Arterial Pressure (MAP) = (2 Diastolic Blood Pressure + Systolic Blood Pressure) / 3 For our study, we computed MAP using the diastolic blood pressure (ap_lo) and systolic blood pressure (ap_hi) data for each instance. Similar to the approach taken with the age attribute,we segmented the MAP data into 10 intervals, spanning from 70–80 to 110–120. Each intervalwas then assigned a categorical label for ease of analysis, as detailed in Table 3.

Table 3. MAP categorical values

MAP Values	Category
≥70 and <80	1
≥80 and <90	2
≥90 and <100	3
≥100 and <110	4
≥110 and <120	5

Now, we drop attributes that had continuous data like height, weight, ap_hi, ap_lo, and age.we transformed all attribute values into categorical representations. This categorization of the data played a crucial role in enabling the model to produce more accurate predictions.

Table 4. Final attributes after feature selection and reduction

Feature	Variable	Min and Max Values
Gender	gender	1: female and 2: male
Age	age_group	Categorical values = 0(min) to 6(max)
BMI	bmi	Categorical values = 0(min) to 5(max)
Mean arterial pressure	map	Categorical values = 0(min) to 5(max)
Cholesterol	cholesterol	Categorical value=1(min) to 3(max)
Glucose	gluc	Categorical value=1(min) to 3(max)
Smoking	smoke	1: yes and 0: no
Alcohol intake	alco	1: yes and 0: no
Physical activity	active	1: yes and 0: no
Presence or absence of cardiovascular disease	cardio	1: yes and 0: no

6.2 Clustering:

Clustering is a machine-learning technique used to group instances based on similarity measures. While k-means is a common algorithm for clustering, it's ineffective with categorical data. The k-modes algorithm, introduced by Huang[21] in 1997, addresses this limitation by using dissimilarity measures for categorical data and replacing means with modes. This makes it effective with categorical data. To find the optimal number of clusters, we use the elbow curve method with Huang initialization. This method identifies a "knee" or inflection point in the plot of costs to determine the optimal number of clusters. Splitting the dataset by gender can be advantageous due to significant biological disparities. For instance, heart disease manifests differently in men and women, with men having a higher risk of coronary artery disease. Utilizing the elbow curve method separately for male and female datasets showed that 2 clusters were optimal for both genders.

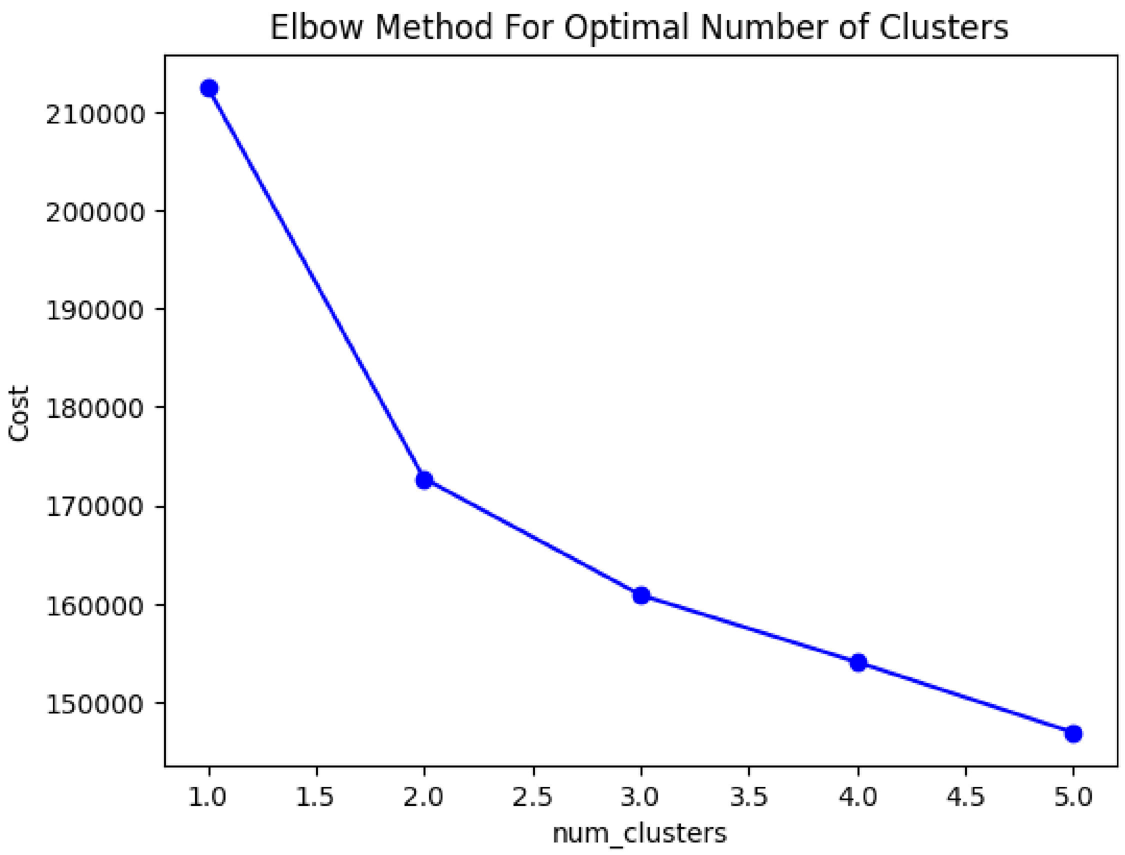
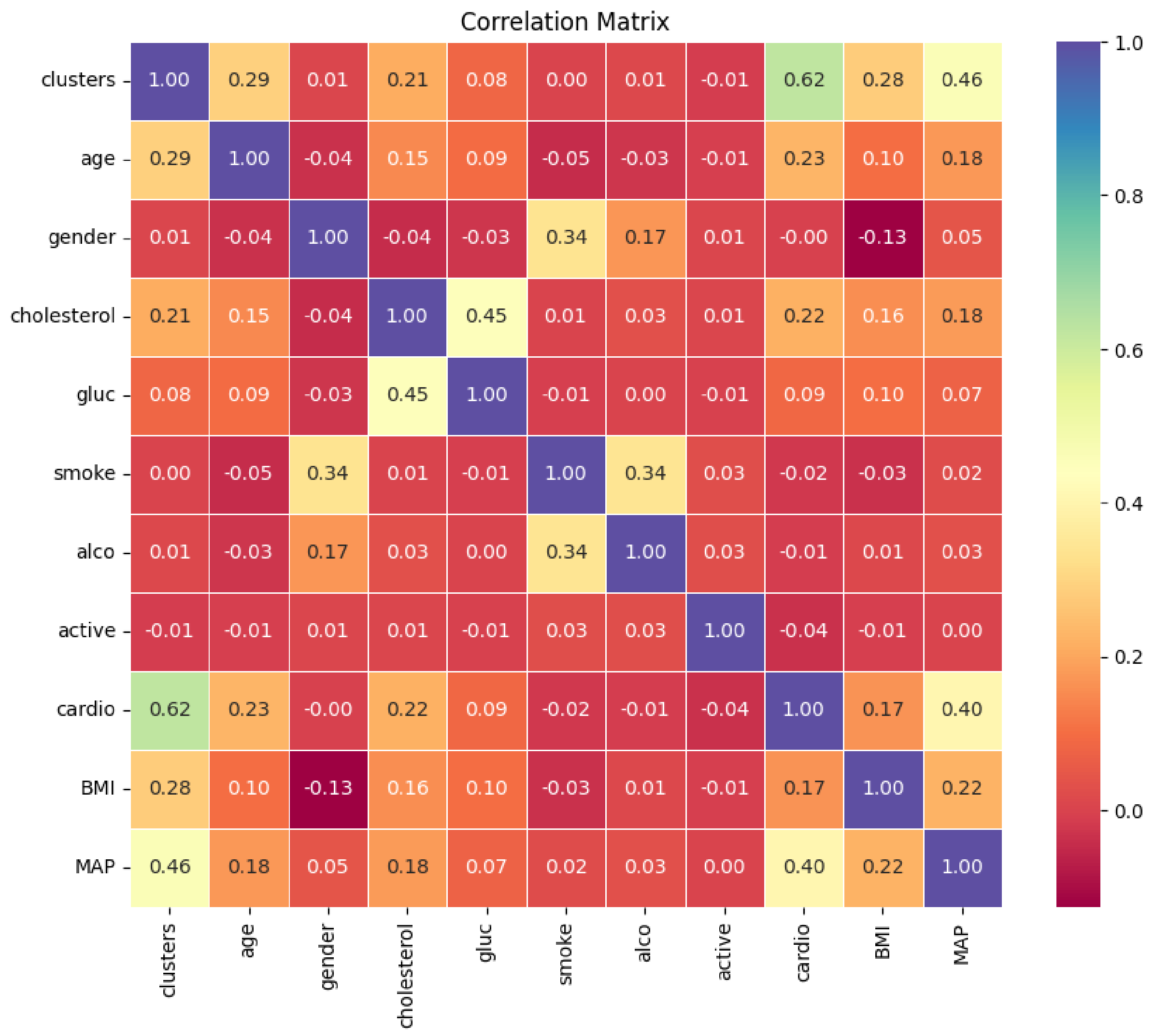


Figure 4. Elbow Curve

6.3 Correlation Table

A correlation table is prepared to determine the correlation between different categories. Intra-feature dependency can also be looked upon with the help of this matrix.

'gender' has a correlation of 0 to our target 'cardio', and 'smoke' correlates '0.01'. We will remove those features to increase performance.



	clusters	age	gender	cholesterol	gluc	smoke	alco	active	cardio	BMI	MAP
clusters	1.00	0.29	0.01	0.21	0.08	0.00	0.01	-0.01	0.62	0.28	0.46
age	0.29	1.00	-0.04	0.15	0.09	-0.05	-0.03	-0.01	0.23	0.10	0.18
gender	0.01	-0.04	1.00	-0.04	-0.03	0.34	0.17	0.01	-0.00	-0.13	0.05
cholesterol	0.21	0.15	-0.04	1.00	0.45	0.01	0.03	0.01	0.22	0.16	0.18
gluc	0.08	0.09	-0.03	0.45	1.00	-0.01	0.00	-0.01	0.09	0.10	0.07
smoke	0.00	-0.05	0.34	0.01	-0.01	1.00	0.34	0.03	-0.02	-0.03	0.02
alco	0.01	-0.03	0.17	0.03	0.00	0.34	1.00	0.03	-0.01	0.01	0.03
active	-0.01	-0.01	0.01	0.01	-0.01	0.03	0.03	1.00	-0.04	-0.01	0.00
cardio	0.62	0.23	-0.00	0.22	0.09	-0.02	-0.01	-0.04	1.00	0.17	0.40
BMI	0.28	0.10	-0.13	0.16	0.10	-0.03	0.01	-0.01	0.17	1.00	0.22
MAP	0.46	0.18	0.05	0.18	0.07	0.02	0.03	0.00	0.40	0.22	1.00

Figure 5. Correlation Matrix

CHAPTER 7
MODELLING

A training dataset (80%) and a testing dataset (20%) are created from the dataset. A model is trained using the training dataset, and its performance is assessed using the testing dataset. Different classifiers, such as decision tree classifier, random forest classifier, multilayer perceptron, and XGBoost, are applied to the clustered dataset to assess their performance. The performance of each classifier is then evaluated using accuracy, precision, recall, and F-measure scores.

7.1 Decision Tree Classifier: Decision trees are treelike structures that are used to manage large datasets. They are often depicted as flowcharts, with outer branches representing the results and inner nodes representing the properties of the dataset. Decision trees are popular because they are efficient, reliable, and easy to understand. The projected class label for a decision tree originates from the tree's root. The following steps in the tree are decided by comparing the value of the root attribute with the information in the record. Following a jump on the next node, the matching branch is followed to the value shown by the comparison result. Entropy changes when training examples are divided into smaller groups using a decision tree node. The measurement of this change in entropy is information gain.

Entropy(S) = $\sum$ ci=1 −(Pi log2 Pi)

Information Gain (S, A) = Entropy(S) − $\sum$v∈values(A) |Sv| |S| Entropy(Sv)

An accuracy of 73.0% has been achieved by the decision tree. In research by [4], 72.77% accuracy was achieved by the decision tree classifier[8].

7.2 Random Forest: The random forest[9] algorithm belongs to a category of supervised classification technique that consists of multiple decision trees working together as a group. The class with the most votes becomes the prediction made by our model. Each tree in the random forest makes a class prediction, which eliminates the limitations of the decision tree algorithm. This improves accuracy and reduces the overfitting of the dataset. When used on large datasets, the random forest approach may still provide the same results even if a significant portion of record values are missing. The samples produced by the decision tree may be saved and used with various data types[22]. In the previous research[26], random forest achieved a test accuracy of 73% and a validation accuracy of 72% with 500 estimators, 4 maximum depths, and 1 random state.

7.3 Multilayer Perceptron: The multilayer perceptron (MLP) is a type of artificial neural network that consists of multiple layers. Single perceptron can only solve linear problems, but MLP is better suited for nonlinear examples. MLP is used to tackle complex issues. A feedforward neural network with many layers is an example of an MLP[23]. Other activation functions beyond the step function are usually used by MLP. The buried layer neurons often perform sigmoid functions. As with step functions, smooth transitions rather than rigid decision limits are produced using sigmoid functions[24]. In MLPs, learning also comprises adjusting the perceptron's weights to obtain the lowest possible error. This is accomplished via the backpropagation technique, which reduces the MSE.

7.4 XGBoost: XGBoost[10] is a version of gradient-boosted decision trees. This algorithm involves sequentially creating decision trees. All the independent variables are allocated weights, which are subsequently used to produce predictions by the decision tree. If the tree makes a wrong prediction, the importance of the relevant variables is increased and used in the next decision tree. The output of each of these classifiers/predictors is then merged to produce a more robust and accurate model. In a study by [25], the XGBoost model achieved 73% accuracy with the parameters 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 100, 'cross-validation': 10 folds including 49,000 training and 21,000 testing data instances on 70,000 CVD dataset.

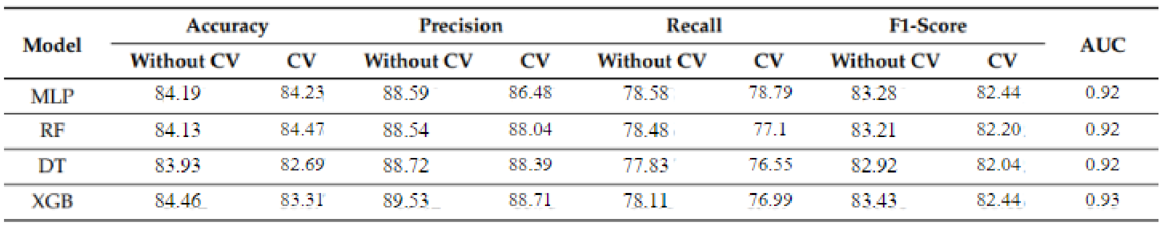
CHAPTER 8
RESULTS AND CONCLUSIONS

8.1. Results

This review utilized Google Colab on an Intel Core i5 processor and 8 GB of RAM. The dataset comprises 70,000 instances with 12 varying attributes, which are reduced to 62,505 instances and 11 attributes. Outliers are successfully removed to improve model efficiency. Several performance metrics are calculated over different classifiers, such as random forest, decision tree, multilayer perceptron, and XGBoost classifier.

Trained 80% of the dataset for each algorithm, employing an automated hyper-tuning technique, called GridSearchCV. The evaluation metrics of the different classifiers are given in the below table. The result indicates that random forest with cross-validation achieves the highest accuracy of 84.47%, outperforming all other algorithms.

Table 5. The evaluation metrics resulting from different classifiers.



Model	Accuracy		Precision		Recall		F1-Score		AUC
	Without CV	CV	Without CV	CV	Without CV	CV	Without CV	CV	
MLP	84.19	84.23	88.59	86.48	78.58	78.79	83.28	82.44	0.92
RF	84.13	84.47	88.54	88.04	78.48	77.1	83.21	82.20	0.92
DT	83.93	82.69	88.72	88.39	77.83	76.55	82.92	82.04	0.92
XGB	84.46	83.31	89.53	88.71	78.11	76.99	83.43	82.44	0.93

8.2. Conclusion

The key objective of this review paper was to classify the occurrence of heart diseases using different machine-learning models, on a real-world dataset. Employing a k-mode clustering approach on this huge dataset was to predict the presence of heart disease after several preprocessing approaches were applied. Feature reduction and selection have been achieved by binning methodology, which improved the efficiency of data transformation, from categorical data into numerical data.

The elbow Curve method was utilized to obtain an optimal number of clusters, for precise classification of the model. The correlation matrix, in hand, provided the correlations between the different categories. Employing different classifiers on 80% of the dataset enhanced the accuracy of the outputs obtained, which resulted in a huge rise in accuracy among all the algorithms. The accuracies of all the algorithms achieved were higher than 82%, where, random forest with cross-validation outperformed all other algorithms with an accuracy of 84.47%,

Limitations: Despite being highly accurate algorithms with over 70,000 instances of the dataset, this paper doesn’t ensure to achieve similar accuracies when dealing with other datasets. Furthermore, this review didn’t take into account other risk factors, such as lifestyle factors or genetic predispositions. On account of the several limitations, it was advised to conduct further research to address these limitations and to better understand the potential risk factors that play a major role in the occurrence of heart diseases.

REFERENCES

[1] Estes, C.; Anstee, Q.M.; Arias-Loste, M.T.; Bantel, H.; Bellentani, S.; Caballeria, J.; Colombo, M.; Craxi, A.; Crespo, J.; Day, C.P.; et al. Modeling NAFLD disease burden in China, France, Germany, Italy, Japan, Spain, United Kingdom, and United States for the period 2016–2030. J. Hepatol. 2018, 69, 896–904.

[2] Drozd˙ z, K.; Nabrdalik, K.; Kwiendacz, H.; Hendel, M.; Olejarz, A.; Tomasik, A.; Bartman, W.; Nalepa, J.; Gumprecht, J.; Lip, G.Y.H. ˙ Risk factors for cardiovascular disease in patients with metabolic-associated fatty liver disease: A machine learning approach. Cardiovasc. Diabetol. 2022, 21, 240.

[3] Murthy, H.S.N.; Meenakshi, M. Dimensionality reduction using neuro-genetic approach for early prediction of coronary heart disease. In Proceedings of the International Conference on Circuits, Communication, Control and Computing, Bangalore, India, 21–22 November 2014; pp. 329–332.

[4] Shorewala, V. Early detection of coronary heart disease using ensemble techniques. Inform. Med. Unlocked 2021, 26, 100655.

[5] Mozaffarian, D.; Benjamin, E.J.; Go, A.S.; Arnett, D.K.; Blaha, M.J.; Cushman, M.; de Ferranti, S.; Després, J.-P.; Fullerton, H.J.; Howard, V.J.; et al. Heart disease and stroke statistics—2015 update: A report from the American Heart Association. Circulation 2015, 131, e29–e322.

[6] Purushottam; Saxena, K.; Sharma, R. Efficient Heart Disease Prediction System. Procedia Comput. Sci. 2016, 85, 962–969.

[7] Soni, J.; Ansari, U.; Sharma, D.; Soni, S. Predictive Data Mining for Medical Diagnosis: An Overview of Heart Disease Prediction. Int. J. Comput. Appl. 2011, 17, 43–48.

[8] Waigi, R.; Choudhary, S.; Fulzele, P.; Mishra, G. Predicting the risk of heart disease using advanced machine learning approach. Eur. J. Mol. Clin. Med. 2020, 7, 1638–1645.

[9] Breiman, L. Random forests. Mach. Learn. 2001, 45, 5–32.

[10] Chen, T.; Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the KDD '16: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 785–794.

[11] Narin, A.; Isler, Y.; Ozer, M. Early prediction of Paroxysmal Atrial Fibrillation using frequency domain measures of heart rate variability. In Proceedings of the 2016 Medical Technologies National Congress (TIPTEKNO), Antalya, Turkey, 27–29 October 2016.

[12] Shah, D.; Patel, S.; Bharti, S.K. Heart Disease Prediction using Machine Learning Techniques. SN Comput. Sci. 2020, 1, 345.

[13] Alotaibi, F.S. Implementation of Machine Learning Model to Predict Heart Failure Disease. Int. J. Adv. Comput. Sci. Appl. 2019, 10, 261–268

[14] Hasan, N.; Bao, Y. Comparing different feature selection algorithms for cardiovascular disease prediction. Health Technol. 2020, 11, 49–62.

[15] Ouf, S.; ElSeddawy, A.I.B. A proposed paradigm for intelligent heart disease prediction system using data mining techniques. J. Southwest Jiaotong Univ. 2021, 56, 220–240.

[16] Khan, I.H.; Mondal, M.R.H. Data-Driven Diagnosis of Heart Disease. Int. J. Comput. Appl. 2020, 176, 46–54.

[17] Kaggle Cardiovascular Disease Dataset. Available online: https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset.

[18] Rivero, R.; Garcia, P. A Comparative Study of Discretization Techniques for Naive Bayes Classifiers. IEEE Trans. Knowl. Data Eng. 2009, 21, 674–688.

[19] Khan, S.S.; Ning, H.; Wilkins, J.T.; Allen, N.; Carnethon, M.; Berry, J.D.; Sweis, R.N.; Lloyd-Jones, D.M. Association of body mass index with lifetime risk of cardiovascular disease and compression of morbidity. JAMA Cardiol. 2018, 3, 280–287. [CrossRef]

[20] Yu, D.; Zhao, Z.; Simmons, D. Interaction between Mean Arterial Pressure and HbA1c in Prediction of Cardiovascular Disease Hospitalisation: A Population-Based Case-Control Study. J. Diabetes Res. 2016, 2016, 8714745

[21] Huang, Z. A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining. DMKD 1997, 3, 34–39.

[22] Bhunia, P.K.; Debnath, A.; Mondal, P.; D E, M.; Ganguly, K.; Rakshit, P. Heart Disease Prediction using Machine Learning. Int. J. Eng. Res. Technol. 2021, 9.

[23] Mohanty, M.D.; Mohanty, M.N. Verbal sentiment analysis and detection using recurrent neural network. In Advanced Data Mining Tools and Methods for Social Computing; Academic Press: Cambridge, MA, USA, 2022; pp. 85–106.

[24] Menzies, T.; Kocagüneli, E.; Minku, L.; Peters, F.; Turhan, B. Using goals in model-based reasoning. In Sharing Data and Models in Software Engineering; Morgan Kaufmann: San Francisco, CA, USA, 2015; pp. 321–353.
[25] Fayez, M.; Kurnaz, S. Novel method for diagnosis diseases using advanced high-performance machine learning system. Appl. Nanosci. 2021.
[26] Maiga, J.; Hungilo, G.G.; Pranowo. Comparison of Machine Learning Models in Prediction of Cardiovascular Disease Using Health Record Data. In Proceedings of the 2019 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), Jakarta, Indonesia, 24–25 October 2019; pp. 45–48.

Appendix: A- Packages, Tools Used & Working Process

1.Python Programming language

Python is a high-level Interpreter based programming language used especially for general-purpose programming. Python features a dynamic type of system and supports automatic memory management.

It supports multiple programming paradigms, including object-oriented, functional, and Procedural, and also has a large and comprehensive standard library. Python has two versions. They are Python 2 and Python 3.

This project uses the latest version of Python, i.e., Python 3. This Python language uses different types of memory management techniques such as reference counting and a cycle-detecting garbage collector for memory management. One of its features is late binding (dynamic name resolution), which binds method and variable names during program execution. Python offers a design that supports some of the things that are used for functional programming in the Lisp tradition. It has vast usage of functions for faster results such as filter, map, split, list comprehensions, dictionaries, sets, and expressions. The standard library of Python language has two modules itertools and functools that implement functional tools taken from Standard machine learning.

2.Libraries

NumPy

Numpy is the basic package for scientific calculations and computations used along with Python. NumPy was created in 2005 by Travis Oliphant. It is open source so can be used freely. NumPy stands for Numerical Python. It is used for working with arrays and mathematical computations.

Using NumPy in Python gives you much more functional behavior comparable to MATLAB because they both are interpreted, and they both allow the users to quickly write fast programs as far as most of the operations work on arrays and matrices instead of scalars. Numpy is a library consisting of array objects and a collection of routines for processing those arrays.

Numpy has also functions that mostly work on linear algebra, Fourier transforms, arrays, and matrices. In a general scenario, the working of NumPy in the code involves searching, joining, splitting, reshaping, etc. operations using NumPy.

The syntax for importing the NumPy package is → import NumPy as np indicates NumPy is imported alias np.

Pandas

Pandas are used whenever working with matrix data, time series data, and mostly tabular data. Pandas is also an open-source library that provides high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

This helps extremely in handling large amounts of data with the help of data structures like Series, Data Frames, etc. It has inbuilt methods for manipulating data in different formats like CSV, HTML, etc.,

Simply we can define pandas as used for data analysis and data manipulation and extremely works with data frame objects in our project, where a data frame is a dedicated structure for two-dimensional data, and it consists of rows and columns similar to database tables and Excel spreadsheets.

In our code, we first import pandas package alias pd and use pd to read the CSV file and assign it to the data frame and in the further steps. We work on the data frames by manipulating them and we perform data cleaning by using functions on the data frames such as df.isna().sum(). So, finally, the whole code depends on the data frames that are to be acquired by coordinating with pandas. So, this package plays a key role in our project.

Matplotlib

Matplotlib is a library used for plotting in the Python programming language and it is a numerical mathematical extension of NumPy. Matplotlib is most commonly used for visualization and data exploration in a way that statistics can be known using different visual structures by creating basic graphs like bar plots, scatter plots, histograms, etc.

Matplotlib is a foundation for every visualizing library and the library also offers great flexibility with regards to formatting and styling plots. We can choose freely certain assumptions like ways to display labels, grids, legends, etc.

In our code firstly we import the matplotlib.pyplot alias plt, This plt comes into the picture in the exploratory data analysis part to analyze and summarize datasets into visual methods, we use plt to add some characteristics to figures such as titles, legends, and labels on the x and y axis as said earlier, to understand more clearly we can also use different plots.

Seaborn

Seaborn is used for drawing attractive statistical graphics with just a few lines of code. In other words, we can say Seaborn is a data visualization library based on matplotlib and closely combined with Pandas data structures in Python. Visualization is the central theme of Seaborn which helps in the exploration and understanding of data. Plots are used for visualizing the relationship between variables. Those variables can be numerical or categorical.

Using Seaborn, we can also plot wide varieties of plots like Distribution plots, boxplots, Pie charts and bar charts, Scatter plots, Pair plots, and Heat maps. In our project, Seaborn's boxplots serve as a visual tool to explore the distribution characteristics of numerical variables within your dataset. A boxplot, or box-and-whisker plot, offers a succinct summary of the central tendency, spread, and presence of outliers in the data. The central line within the box represents the median, offering insight into the typical value of the variable under consideration. Meanwhile, the length of the box indicates the interquartile range (IQR), representing the middle 50% of the data's distribution. A longer box suggests higher variability within this range. The whiskers extending from the box illustrate the data's range, excluding outliers. Outliers, plotted individually as points beyond the whiskers, are data points that fall significantly beyond the expected range. Detecting outliers is crucial for understanding the data's distribution and identifying potential anomalies that may affect subsequent analysis or modeling efforts. Furthermore, comparing boxplots of different variables side by side facilitates the comparison of their central tendencies, dispersions, and outlier distributions, enabling insights into potential relationships or disparities between variables. Overall, Seaborn's boxplots provide valuable insights into the distribution characteristics of numerical variables related to cardiovascular health, aiding in exploratory data analysis and informing subsequent modeling endeavors.

Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. Scikit Learn is an efficient, and beginner, user-friendly tool for predictive data analysis and it provides a selection of tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistent interface in Python. This library is built upon different libraries such as NumPy, SciPy, and Matplotlib.

Scikit learning is used when identifying to which category an object likely belongs, predicting continuous values, and grouping similar objects into clusters.

In our project, the scikit-learn library (sklearn) plays a fundamental role in facilitating various machine-learning tasks, including data preprocessing, model training, evaluation, and hyperparameter tuning. Several modules from sci-kit-learn are utilized in your project for different purposes:

1. `metrics` Module: This module provides functions for evaluating model performance using various metrics such as accuracy, precision, recall, F1-score, and ROC-AUC score. These metrics help quantify the effectiveness of the predictive models trained on your dataset. In your

project, these metrics are essential for assessing the accuracy and robustness of your cardiovascular disease prediction models.

2. `preprocessing` Module: The preprocessing module offers functionalities for preparing and preprocessing the dataset before training machine learning models. This includes tasks such as standardization (scaling) of numerical features, encoding categorical variables, handling missing values, and splitting the dataset into training and testing sets. In your project, preprocessing tasks are crucial for ensuring the quality and compatibility of the data with the machine learning algorithms used.

3. `model_selection` Module: The model_selection module provides utilities for tasks related to model selection and evaluation, including cross-validation, grid search for hyperparameter tuning, and train-test splitting. Cross-validation ensures that the model's performance is robust and reliable across different subsets of the data, while grid search helps identify the optimal hyperparameters for your models. Train-test splitting allows for the creation of independent training and testing sets, enabling unbiased evaluation of model performance. These functionalities are essential for optimizing the predictive models in your project.

4. Specific Model Classes: Your project imports specific model classes such as LogisticRegression, RandomForestClassifier, MLPClassifier, DecisionTreeClassifier, SVC, and XGBClassifier. These classes represent different machine learning algorithms/models used for predicting cardiovascular diseases based on the dataset. Each model class has its own set of parameters and methods for training, predicting, and evaluating the model's performance.

Kmodes

The `kmodes` library is a Python package designed for clustering categorical data using the k-modes algorithm. Unlike traditional clustering algorithms such as k-means, which are tailored for numerical data, k-modes are specifically suited for datasets with categorical or nominal attributes. The k-modes algorithm extends the principles of k-means by operating on categorical data, calculating the mode (most frequent value) for each cluster centroid. It iteratively assigns data points to the nearest cluster centroid based on a dissimilarity measure, such as the matching dissimilarity or Hamming distance. It updates the centroids until convergence is reached.

This library offers an efficient and scalable implementation of the k-modes algorithm, enabling users to perform clustering on large datasets with categorical attributes. It provides various functionalities for clustering categorical data, including different initialization methods for initializing cluster centroids, such as random initialization, Huang's method, and Cao's method.

Additionally, it supports alternative clustering algorithms like k-prototypes, which can handle datasets with a mixture of numerical and categorical attributes.
One of the key advantages of the `modes` library is its scalability and memory efficiency, making it suitable for clustering large datasets with thousands or even millions of data points. Furthermore, it offers flexibility in parameter customization, allowing users to specify the number of clusters (k), the dissimilarity measure, and other algorithmic parameters according to their specific requirements.

XGBoost

The XGBoost library, or eXtreme Gradient Boosting, is a renowned open-source machine learning library widely recognized for its exceptional performance in supervised learning tasks, notably regression and classification. It operates within the gradient boosting framework, where it sequentially builds multiple weak predictive models, often decision trees, with each subsequent model aiming to rectify the errors of its predecessors. By aggregating the predictions of these weak learners, weighted by their performance, XGBoost delivers the final prediction. One of its standout features is the incorporation of regularization techniques like L1 (Lasso) and L2 (Ridge) regularization, which help prevent overfitting by penalizing the complexity of the model. Furthermore, XGBoost offers built-in support for cross-validation, enabling users to assess model performance and fine-tune hyperparameters effectively. Its design for parallel and distributed computing ensures scalability and efficiency, making it suitable for handling large datasets. Additionally, XGBoost provides advanced tree pruning and split-finding algorithms to optimize model construction and enhance predictive accuracy. With customizable objective functions and evaluation metrics, users can tailor the model to specific tasks and domains. Finally, XGBoost offers interpretability features such as feature importance scores, facilitating the understanding of individual feature contributions to model predictions. Overall, the XGBoost library stands as a robust and versatile tool for building accurate predictive models, widely utilized across various industries and machine learning applications.

Appendix: B

Sample Source Code with Execution

```
import numpy as np

import pandas as pd

from sklearn import metrics, preprocessing

from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.neural_network import MLPClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

!pip install kmodes

from kmodes.kmodes import KModes

from sklearn.metrics import classification_report

from sklearn. preprocessing import LabelEncoder

import matplotlib. pyplot as plt

import seaborn as sb

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score

from xgboost import XGBClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import roc_curve, auc


df = pd.read_csv("/content/cardio_train.csv")                    #read the dataset
```

DATA CLEANING

Identifying

```
num_cols = len(df.columns)              # Determine number of columns and rows needed

num_rows = (num_cols - 1)


for row in range(num_rows):                    # Create subplots

plt.figure(figsize=(16, 12))

for i, column in enumerate(df.columns[row*12:min((row+1)*12, num_cols)]):

plt.subplot(4, 3, i + 1)

sb.boxplot(data=df[column])

plt.title(column)

plt.tight_layout()

plt.show()
```

Removal of outliers

```
def statistic_Info(flier):

Q1 = np.percentile(df[flier],25)

Q3 = np.percentile(df[flier],75)

IQR = Q3 - Q1

lower_Whisker = np.max([df[flier].min(), Q1 - 1.5*IQR])

upper_Whisker = np.min([df[flier].max(), Q3 + 1.5*IQR])

return Q1, Q3, IQR, lower_Whisker, upper_Whisker

#Height

q1,q3,k,low,high = statistic_Info("height")

print(q1, q3, k, low, high)                          # 159.0 170.0 11.0 142.5 186.5

numFliers = df[(df['height'] > high) | (df['height'] < low)].shape[0]  # Finding Outliers
```

```
print("No. of fliers: ", numFliers)              #519

totalFliers = numFliers                          # Total Fliers

print("Total fliers: ", totalFliers)                                       #519

df = df[(df['height'] <= high) & (df['height'] >= low)]                    # Trim outliers

# Checking if outliers are successfully removed

numFliers = df[(df['height'] > high) | (df['height'] < low)].shape[0]

print("No. of fliers after trimming: ", numFliers)             #0


#Weight

q1,q3,k,low,high = statistic_Info("weight")

print(q1, q3, k, low, high)              # 65.0 82.0 17.0 39.5 107.5

numFliers = df[(df['weight'] > high) | (df['weight'] < low)].shape[0] # Finding Outliers

print("Number of outliers in 'weight' column:", numFliers)           #1758

totalFliers = totalFliers + numFliers                  # Total Fliers

print("Total number of outliers so far:", totalFliers)               #2277

print("Number of rows before trimming outliers:", df.shape[0])# Checking available rows


df = df[(df['weight'] >= low) & (df['weight'] <= high)]              # Trimming outliers

print("Number of rows after trimming outliers:", df.shape[0])        #69481

# Checking if outliers are successfully removed

numFliers = df[(df['weight'] > high) | (df['weight'] < low)].shape[0]

print("Number of outliers in 'weight' column after trimming:", numFliers)          #0


#ap_hi

q1,q3,k,low,high = statistic_Info("ap_hi")
```

```
print(q1, q3, k, low, high)                                                    # 120.0 140.0 20.0 90.0 170.0

numFliers = df[(df['ap_hi'] > high) | (df['ap_hi'] < low)].shape[0]    # Finding Outliers

print("Number of outliers in 'ap_hi' column:", numFliers)              #1309

totalFliers = totalFliers + numFliers                                        # Total Fliers

print("Total number of outliers so far:", totalFliers)                    #3586

# Checking available rows

print("Number of rows before trimming in 'ap_hi' column:", df['ap_hi'].shape[0])  #67723

df = df[(df['ap_hi'] <= high) & (df['ap_hi'] >= low)]                          # Trimming outliers

print("Number of rows after trimming outliers in 'ap_hi' column:", df.shape[0])    #66414

# Checking if outliers are successfully removed

numFliers = df[(df['ap_hi'] > high) | (df['ap_hi'] < low)].shape[0]

print("Number of outliers in 'ap_hi' column after trimming:", numFliers)           #0


#ap_lo

q1,q3,k,low, high = statistic_Info("ap_lo")

print(q1,q3,k,low, high)                    # 80.0 90.0 10.0 65.0 105.0

numFliers = df[(df['ap_lo']<low) | (df['ap_lo']>high)].shape[0]       #Finding outliers

print("Number of outliers in 'ap_lo' column:", numFliers)              #3909

totalFliers = totalFliers + numFliers                                        #total fliers

print("Total number of outliers so far:", totalFliers)                    #7495

# Checking available rows

print("Number of rows before trimming in 'ap_lo' column:", df['ap_lo'].shape[0])  #66414

df = df[(df['ap_lo']>=low) & (df['ap_lo']<=high)]                            #trimming outliers

print("Number of rows after trimming outliers in 'ap_lo' column:", df.shape[0])    #62505
```

#checking if outliers are successfully removed

```
numFliers = df[(df['ap_lo']>high) | (df['ap_lo']<low)].shape[0]
print("Number of outliers in 'ap_lo' column after trimming:", numFliers)    #0


df.isna().sum()                   #Checking null values
```

FEATURE SELECTION AND REDUCTION

Age

```
df["age"] = round(df["age"]/365.0)          #Converting number of days into years
min_Age = int(df["age"].min())              #Calculating interval
max_Age = int(df["age"].max())
print(min_Age, max_Age)                     #30 65
interval = 5                                #defining interval, bins, and labels
age_bins = [30,35,40,45,50,55,60,70]
age_labels = list(i for i in range(len(age_bins)-1))
print(f"{age_bins}\n{age_labels}")   #[30, 35, 40, 45, 50, 55, 60, 70] [0, 1, 2, 3, 4, 5, 6]
df["age"] = pd.cut(df["age"],bins=age_bins, labels=age_labels, include_lowest=True,
right=False)
```

BMI

#Converting units of centimeters into meters for attribute "height"

```
df["height"] = df["height"]/100.0
df["BMI"] = round(df["weight"]/(df["height"]**2),1)              #calculate BMI
df = df.drop(["height", "weight"], axis=1)
min_bmi = min(df["BMI"])
max_bmi = max(df["BMI"])
```

print(f"{min_bmi}\t{max_bmi}") **#13.5 50.9**

bmi_bins = [0,18.5,25,30,35,40,100] **# Defining bins based on the known BMI ranges**

bmi_labels = [i for i in range(len(bmi_bins)-1)]

print(f"{bmi_bins}\n{bmi_labels}") **# [0, 18.5, 25, 30, 35, 40, 100] [0, 1, 2, 3, 4, 5]**

df["BMI"]=pd.cut(df["BMI"],bins=bmi_bins,labels=bmi_labels,right=False,include_lowest=True)

MAP

df["MAP"] = round(((2*df["ap_lo"])+df["ap_hi"])/3)

df = df.drop(["ap_lo","ap_hi"],axis=1)

df.head(10)

min_map = min(df["MAP"])

max_map = max(df["MAP"])

print(f"{min_map}\t{max_map}") **#73.0 126.0**

map_bins = [0,70,80,90,100,110,140]**# Defining bin values based on the above criteria**

map_labels = list(i for i in range(len(map_bins)-1))

print(f"{map_bins}\n{map_labels}") **# [0, 70, 80, 90, 100, 110, 140] [0, 1, 2, 3, 4, 5]**

df["MAP"] = pd.cut(df['MAP'],bins=map_bins,labels=map_labels,include_lowest=True,right=False)

DATA TRANSFORMATION

labelEncoder = preprocessing.LabelEncoder()

df = df.apply(labelEncoder.fit_transform)

nan_rows = df[df.isna().any(axis=1)] **# Identify rows containing NaN values**

CLUSTERING AND MODELLING

Elbow Curve Method
cost = []
num_clusters = range(1,6) # 1 to 5

```
for i in list(num_clusters):
    kmode = KModes(n_clusters=i, init = "Huang", n_init = 5, verbose=0,random_state=1)
    kmode.fit_predict(df)
    cost.append(kmode.cost_)

plt.plot(num_clusters, cost, 'bo-')
plt.xlabel('num_clusters')
plt.ylabel('Cost')
plt.title('Elbow Method For Optimal Number of Clusters')
plt.show()
```

K-modes Clustering

```
km = KModes(n_clusters=2, init = "Huang", n_init = 5,random_state=1)
clusters = km.fit_predict(df)
```

Correlation matrix

```
df.insert(0,"clusters",clusters,True)
df.head(10)

plt.figure(figsize=(10, 8))
# Draw correlation matrix
sb.heatmap(df.corr(), annot=True, cmap='Spectral', fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix')            # Show the figure
plt.show()

sb.countplot(x='clusters', hue='cardio', data=df)
plt.title('Distribution of Cardiovascular Disease within Clusters')
plt.show()
```

TRAINING

```
x = df.drop(['cardio',' gender', 'alco'], axis=1)
y = df['cardio']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=1)
```

```
x_train_xg,x_test_xg,y_train_xg,y_test_xg=train_test_split(x,y,test_size=0.20,random_state=1)
x_train_mlp,x_test_mlp,y_train_mlp,y_test_mlp=train_test_split(x,y,test_size=0.20,random_
state=1)
x_train_rf,x_test_rf,y_train_rf,y_test_rf=train_test_split(x,y,test_size=0.20,random_state=1)
x_train_dt,x_test_dt,y_train_dt,y_test_dt=train_test_split(x,y,test_size=0.20,random_state=1)
```

MODELLING

Random Forest

```
x = df.drop(['cardio', 'gender', 'alco'], axis=1)          # Define features (x) and target (y)
y = df['cardio']

rf_model = RandomForestClassifier(random_state=1)         # create model
rf_model.fit(x_train, y_train)                            # Fit the model
rf_pred = rf_model.predict(x_test)                        # Make predictions
rf_accuracy = metrics.accuracy_score(y_test, rf_pred) * 100          # Accuracy

# Define a simplified parameter grid for Random Forest
param_grid_rf = {'n_estimators': [100, 200], 'max_depth': [10, 20],'min_samples_split': [5,
10], 'min_samples_leaf': [1, 2], 'max_features': ['sqrt', None],}

# Create grid search for Random Forest
rf_gridsearch = GridSearchCV(estimator=rf_model, param_grid=param_grid_rf, cv=5,
scoring='accuracy', n_jobs=-1)
rf_gridsearch.fit(x_train, y_train)                       # Fit grid search for Random Forest
best_params = rf_gridsearch.best_params_
best_estimator = rf_gridsearch.best_estimator_
print(f"Best Parameters : {best_params}")
print(f"Best Estimator  : {best_estimator}")

# Best Parameters : {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1,
'min_samples_split': 5, 'n_estimators': 100}
#Best Estimator  : RandomForestClassifier(max_depth=10, max_features=None,
min_samples_split=5,random_state=1)
```

rf_pred_CV = best_estimator.predict(x_test_rf)

rf_accuracy_cv = metrics.accuracy_score(y_test_rf, rf_pred_CV)*100

classification_report_str = classification_report(y_test, rf_pred_CV, digits=4)

Multilayer Perceptron

mlpModel = MLPClassifier(random_state=1) **# build MLP model**

mlpModel.fit(x_train_mlp, y_train_mlp) **# Fit the model**

mlp_pred = mlpModel.predict(x_test_mlp) **# Make predictions**

mlp_accuracy = metrics.accuracy_score(y_test_mlp, mlp_pred)*100 **# accuracy**

mlp_best_params = {'activation': ['tanh'], 'alpha': [0.01], 'hidden_layer_sizes': [(50, 50)], 'max_iter': [300], 'solver': ['adam'],} **# Best parameters for MLP**

mlp_gridsearch = GridSearchCV(estimator=mlpModel, param_grid=mlp_best_params, cv=5, scoring='accuracy', n_jobs=-1) **# Create grid search**

mlp_gridsearch.fit(x_train_mlp, y_train_mlp) **# Fit grid search**

mlp_best_params = mlp_gridsearch.best_params_

mlp_best_estimator = mlp_gridsearch.best_estimator_

print(f"Best Parameters : {mlp_best_params}")

print(f"Best Estimator : {mlp_best_estimator}")

#Best Parameters: {'activation': ['tanh'], 'alpha': [0.01], 'hidden_layer_sizes': [(50, 50)], 'max_iter': [300], 'solver': ['adam']}

#Best Estimator : MLPClassifier(activation='tanh', alpha=0.01, hidden_layer_sizes=(50, 50),max_iter=300, random_state=1)

mlp_pred_CV = mlp_best_estimator.predict(x_test_mlp)

mlp_accuracy_cv = metrics.accuracy_score(y_test_mlp, mlp_pred_CV)*100

classification_report_str = classification_report(y_test, mlp_pred_CV, digits=4)

Decision Tree

dtModel = DecisionTreeClassifier(random_state=1) **# build MLP model**

dtModel.fit(x_train_dt, y_train_dt) **# Fit the model**

```
dt_pred = dtModel.predict(x_test_dt)                    # Make predictions

dt_accuracy = metrics.accuracy_score(y_test_dt, dt_pred)*100                # accuracy

dt_best_params = {'criterion': ['entropy'], 'max_depth': [6], 'n_estimators': [100], 'subsample':
[0.8], 'colsample_bytree': [0.8],}        # Best parameters for dt

dt_gridsearch = GridSearchCV(estimator=dtModel, param_grid=dt_best_params, cv=5,
scoring='accuracy', n_jobs=-1)          # Create grid search

dt_gridsearch.fit(x_train_dt, y_train_dt)               # Fit grid search

dt_best_estimator = dt_gridsearch.best_estimator_

print(f"Best Parameters : {dt_best_params}")

print(f"Best Estimator  : {dt_best_estimator}")

# Best Parameters: {'criterion': ['entropy'], 'max_depth': [6]}

#Best Estimator : DecisionTreeClassifier(criterion='entropy', max_depth=6,
random_state=1)

dt_pred_CV = dt_best_estimator.predict(x_test_dt)
dt_accuracy_cv = metrics.accuracy_score(y_test_dt, dt_pred_CV)*100
```

XGBoost

```
xgbmodel = XGBClassifier(random_state=1)          #build the model

xgbmodel.fit(x_train_xg, y_train_xg)              # Fit the model

xgbpred = xgbmodel.predict(x_test_xg)             # Make predictions

xgbaccuracy = metrics.accuracy_score(y_test_xg, xgbpred)*100                # accuracy

param_grid = {'learning_rate': [0.01, 0.1, 0.2], 'max_depth': [3, 5, 7], 'n_estimators': [50, 100,
200], 'subsample': [0.8, 1.0], 'colsample_bytree': [0.8, 1.0],}

best_xgbparam = {'learning_rate': [0.1], 'max_depth': [3], 'n_estimators': [100], 'subsample':
[0.8], 'colsample_bytree': [0.8],}

grid_search = GridSearchCV(estimator=xgbmodel, param_grid=best_xgbparam, cv=5,
scoring='accuracy', n_jobs=-1)

grid_search.fit(x_train_xg, y_train_xg)

best_params = grid_search.best_params_
```

```
best_model = grid_search.best_estimator_


xgb_pred_CV = best_model.predict(x_test_xg)

xgb_accuracy_cv = metrics.accuracy_score(y_test_xg, xgb_pred_CV)*100
```

PERFORMANCE METRIX EVALUATION

Accuracy without CV

```
print(f"RF accuracy without CV : {rf_accuracy:.2f}")          #84.13

print(f"mlp accuracy without CV : {mlp_accuracy:.2f}")        #84.19

print(f"Dt accuracy without CV : {dt_accuracy:.2f}")          #83.93

print(f"XGB accuracy without CV: {xgbaccuracy:.2f}")          #84.46
```

Accuracy with CV

```
print(f"RF accuracy with CV : {rf_accuracy_cv:.2f}")          #84.47

print(f"mlp accuracy with CV : {mlp_accuracy_cv:.2f}")        #84.23

print(f"Dt accuracy with CV : {dt_accuracy_cv:.2f}")          #82.69

print(f"XGB accuracy with CV: {xgbaccuracy_cv:.2f}")          #83.31
```

AUC scores without CV

```
rf_probs = rf_model.predict_proba(x_test)[:, 1]

dt_probs = dtModel.predict_proba(x_test)[:, 1]

mlp_probs = mlpModel.predict_proba(x_test)[:, 1]

xgb_probs = xgbmodel.predict_proba(x_test)[:, 1]


rf_auc = roc_auc_score(y_test_rf, rf_probs)
```

```
dt_auc = roc_auc_score(y_test_dt, dt_probs)

mlp_auc = roc_auc_score(y_test_mlp, mlp_probs)

xgb_auc = roc_auc_score(y_test_xg, xgb_probs)


print("Random Forest AUC:", round(rf_auc,2))                    #0.92

print("Decision Tree AUC:", round(dt_auc,2))                    #0.92

print("Multi-Layer Perceptron AUC:", round(mlp_auc,2))          #0.92

print("XGBoost AUC:", round(xgb_auc,2))                         #0.93
```

F1 scores without CV

```
rf_f1 = f1_score(y_test_rf, rf_pred)

dt_f1 = f1_score(y_test_dt, dt_pred)

mlp_f1 = f1_score(y_test_mlp, mlp_pred)

xgb_f1 = f1_score(y_test_xg, xgbpred)


print("Random Forest F1 score without CV:", round(rf_f1*100,2))             #83.21

print("Decision Tree F1 score without CV:", round(dt_f1*100,2))             #82.92

print("Multi-Layer Perceptron F1 score without CV:", round(mlp_f1*100,2))   #83.28

print("XGBoost F1 score without CV:", round(xgb_f1*100,2))                  #83.43
```

F1 scores with CV

```
rf_f1_cv = cross_val_score(rf_model, x, y, cv=5, scoring='f1').mean()

dt_f1_cv = cross_val_score(dtModel, x, y, cv=5, scoring='f1').mean()

mlp_f1_cv = cross_val_score(mlpModel, x, y, cv=5, scoring='f1').mean()

xgb_f1_cv = cross_val_score(xgbmodel, x, y, cv=5, scoring='f1').mean()
```

```
print("Random Forest Cross-Validated F1 score:", round(rf_f1_cv*100,2))          #82.20
print("Decision Tree Cross-Validated F1 score:", round(dt_f1_cv*100,2))          #82.04
print("Multi-Layer Perceptron Cross-Validated F1 score:", round(mlp_f1_cv*100,2))#82.44
print("XGBoost Cross-Validated F1 score:", round(xgb_f1_cv*100,2))               #82.44
```

Recall without CV

```
rf_recall = recall_score(y_test_rf, rf_pred)
dt_recall = recall_score(y_test_dt, dt_pred)
mlp_recall = recall_score(y_test_mlp, mlp_pred)
xgb_recall = recall_score(y_test_xg, xgbpred)

print("Random Forest Recall without CV:", round(rf_recall*100,2))                #78.48
print("Decision Tree Recall without CV:", round(dt_recall*100,2))                #77.83
print("Multi-Layer Perceptron Recall without CV:", round(mlp_recall*100,2))      #78.58
print("XGBoost Recall without CV:", round(xgb_recall*100,2))                     #78.11
```

Recall with CV

```
f_recall_cv = cross_val_score(rf_model, x, y, cv=5, scoring='recall').mean()
dt_recall_cv = cross_val_score(dtModel, x, y, cv=5, scoring='recall').mean()
mlp_recall_cv = cross_val_score(mlpModel, x, y, cv=5, scoring='recall').mean()
xgb_recall_cv = cross_val_score(xgbmodel, x, y, cv=5, scoring='recall').mean()
print("Random Forest Cross-Validated Recall:", round(f_recall_cv*100,2))         #77.10
print("Decision Tree Cross-Validated Recall:", round(dt_recall_cv*100,2))        #76.55
print("Multi-Layer Perceptron Cross-Validated Recall:", round(mlp_recall_cv*100,2))#78.79
print("XGBoost Cross-Validated Recall:", round(xgb_recall_cv*100,2))        #76.99
```

Precision without cv

```
rf_precision = precision_score(y_test_rf, rf_pred)
dt_precision = precision_score(y_test_dt, dt_pred)
mlp_precision = precision_score(y_test_mlp, mlp_pred)
xgb_precision = precision_score(y_test_xg, xgbpred)

print("Random Forest Precision without CV:", round(rf_precision*100,2))     #88.54
print("Decision Tree Precision without CV:", round(dt_precision*100,2))     #88.72
print("Multi-Layer Perceptron Precision without CV:", round(mlp_precision*100,2))#88.59
print("XGBoost Precision: without CV", round(xgb_precision*100,2))          #89.53
```

Precision with CV

```
rf_precision_cv = cross_val_score(rf_model, x, y, cv=5, scoring='precision').mean()
dt_precision_cv = cross_val_score(dtModel, x, y, cv=5, scoring='precision').mean()
mlp_precision_cv = cross_val_score(mlpModel, x, y, cv=5, scoring='precision').mean()
xgb_precision_cv = cross_val_score(xgbmodel, x, y, cv=5, scoring='precision').mean()
print("RF Cross-Validated Precision:", round(rf_precision_cv*100,2))        #88.04
print("DT Cross-Validated Precision:", round(dt_precision_cv*100,2))        #88.39
print("MLP Cross-Validated Precision:", round(mlp_precision_cv*100,2))      #86.48
print("XGBoost Cross-Validated Precision:", round(xgb_precision_cv*100,2))  #88.71
```

Paper Publications (if any)

Attach the complete published paper