Predicting Cardiovascular Disease Using Machine Learning and Deep Neural Networks

Submitted By:

Prabhsimran Singh (1598386)

Daksh Karki (1571063)

Abstract

Cardiovascular diseases (CVD) are the major causes of mortality in the world, contributing to 17.9 million deaths annually, where most of these deaths are experienced in low and middle-income countries [1]. The traditional diagnostic models are generally unable to recognise the complex and nonlinear associations among clinical risk factors, as they are not useful in most cases. This report presents the application of machine learning (ML) models, including Artificial Neural Networks (ANN) and Random Forest (RF), to improve the reliability and accuracy of CVD prediction. The study will entail preprocessing a sufficiently big clinical dataset, conducting exploratory data analysis, applying various ML models, and measuring their accuracy with such functions as accuracy, F1-score, and ROC-AUC. As the results indicate, both ANN and RF models demonstrate good performance, whereas the ANN model has better accuracy (85.75%) and has more balanced results in all metrics. This report highlights the potential of machine learning, particularly neural networks, in enhancing early detection and reducing mortality associated with cardiovascular disease (CVD). Future directions must be aimed at enhancing model interpretation with the help of such tools as LIME and integration of real-time, longitudinal data to facilitate clinical applicability and trust.

Table of Contents

1. Introduction 1
1.1 Problem Statement 4
1.2 Project Objectives 4
2.2 Key ML Models Used in previous research: 6
2.2.1 Common Features Used in CVD Prediction 6
2.3.1 Advantages 7
2.3.2 Disadvantages 8
2.3.3 Key Applications in Health Care 8
2.4 Ensemble Learning (Random Forest) 8
2.4.1 Advantages 9
2.4.2 Disadvantages 9
2.4.3 Key Applications in Health Care: 9
3. Dataset Description: 10
4. System Architecture 10
5. Loading the Dataset and Initial Exploration 11
5.1 Library Imports 11
5.2 Dataset Loading 12
5.3 Initial Data Inspection 13
5.4 Exploratory Data Analysis (EDA) 13
5.4.1 Target Variable Distribution 13
5.4.2 Pairplot Analysis 14
5.4.3 Outlier Detection via Boxplots 15
6. Data Preprocessing 16
6.1 Removal of Outlier 16
6.2 Feature Engineering 17
6.2.1 Age Transformation and Grouping 17
6.2.2 Body Mass Index (BMI) 17
6.2.3 Mean Arterial Pressure (MAP) 18
6.3 Dropping Raw Numerical Columns 19
6.4 Categorical Encoding 19
7. Clustering Analysis using KModes 19
7.1 Elbow Method for Optimal Clusters 19
7.2 KModes Clustering 20

7.3 Adding Cluster Information to Dataset....21
7.4 Cluster-Based Visualizations and Insights....21
7.4.1 Correlation Matrix Analysis....21
7.4.2 Distribution of Cardiovascular Disease Across Clusters....22
7.5 Train-Test Split....23
8. Model Training....24
8.1 Random Forest Classifier....24
8.2 Artificial Neural Network....24
9. Results Evaluation....25
9.1 Evaluation of Random Forest....25
9.2 Evaluation of Artificial Neural Network (ANN)....26
10. Model Performance Comparison....27
11. Conclusion....28

1. Introduction

According to WHO, Cardiovascular diseases (CVD) continue to be the primary problem causing deaths in the world, with an average of 17.9 million deaths annually, or 32% of all deaths taking place in the world [1]. 75 percent of all these deaths are in the low and middle-income countries, and 85 per cent of these deaths are due to heart attacks and strokes, requiring urgent measures in preventing the deaths as well as early diagnosis of such deaths. CVD also has devastating impact on the economy, with total costs including treatment, productivity losses, and long-term care is estimated to reach USD 1 trillion by 2030. (World Economic Forum & WHO, 2022). This situation also gets worse as approximately 45 percent of heart attacks can be detected without causing any symptoms, i.e., people do not know about this event until some problems develop (American Heart Association, 2022). Moreover there is an increase of 13% heart attack rate over past 20 years among the individuals who are younger than 40 years of age. (JAMA Cardiology, 2021).

Although widely used, the traditional risk prediction models do not capture the complex and nonlinear interactions between clinical factors, leading to less accurate predictions. Thus, machine learning is applied in this study to improve the prediction of CVD based on clinical and demographic factors. Now, large volumes of clinical data can be analysed, and raw non-linear patterns can be identified, indicating whether a patient is at risk of CVD. ML models analyse clinical features including age, blood pressure, cholesterol and lifestyle habits to produce timely and precise risk assessments.

This project investigates how ML algorithms can be used in predicting cardiovascular diseases and compares the different techniques in aspects of accuracy, reliability and clinical usefulness. It aims to leverage ML capabilities for improving disease prevention methods while maximising clinical resource efficiency and reducing mortality rates.

1.1 Problem Statement

Cardiovascular disease remains the leading cause of mortality worldwide, yet current diagnostics and risk prediction methods are often not up to the mark. Traditional models like the Logistic regression model are not the best choice due to their limitation to linear data assumptions and inability to process complex interactions aiming multiple nonlinear risk metrics. This leads to lower effectiveness in identifying individuals with CVDS across a vast, varied population.

This project addresses the critical question:

How can machine learning models be leveraged to predict cardiovascular disease more accurately and efficiently using clinical and demographic data, compared to traditional risk assessment methods?

1.2 Project Objectives

The key objectives of this project are:

- To collect and pre-process a reliable CVD dataset
- To perform exploratory data analysis (EDA)

- To develop and compare multiple ML models
- To evaluate model performance
- To interpret model predictions
- To identify the best-performing model

2. Literature Review

In early researches related to cardiovascular diseases, the main focus was in the improvement of data quality and completeness before model development. Therefore one such study proposed a machine learning framework that focuses on data imputation to overcome missing values and reduce bias presents in clinical datasets. The models that were trained further outperformed the traditional machine learning approaches in prediction of stroke and heart failure [2].

Following this research there was another study that focuses on identifying the effectiveness conventional machine learning algorithms on clinical metrics such as BMI, blood pressure, cholesterol, and smoking habits. In particular BMI was found to be crucial factor when used in models like ANN, Decision trees and KNN [3].

In subsequent studies there was the introduction of hybrid techniques by combining traditional algorithms, with deep learning based feature extraction. A notable study which implements Random Forest for feature ranking via Gini impurity, followed by a attention based neural network for classification. As a result this model outperformed the traditional models such as Decision Trees, CNN and SVM with an accuracy of 98.72% [4].

Another study implemented the use of ensemble learning technique in prediction of cardiovascular diseases. It implemented and evaluated various algorithms such as Random Forest, SVM, Decision trees, KNN and XGBoost on the publicly available dataset present on kaggle. In result it is seen that XGBoost outperformed other ensemble based models reaching an accuracy of 93%, which confirmed the effectiveness of boosting algorithms in handling structured clinical data [5].

In parallel directions researches also started using multi modal clinical data which consists of genetic inputs and imaging. A study on Ischemic Heart Disease (IHD) uses a multimodal approach in combining data from Electronic Health Records (EHR) and imaging features. As a result fusion model has achieved an AUC of .086, which demonstrates the effectiveness of multi modal data over unimodal data [6].

Further researches focused on the implementation of Generative AI models. One study introduced a GAN based approach and compared its performance with LSTM and RNN. As a result it has seen that GAN based model achieved the highest accuracy of 93% and AUC of 0.953, showing its effectiveness in multi-risk level classification (no, mild, moderate, high) [7].

To improve the diagnostic performance further, researchers further explored the computer vision techniques. In this study several models such as Inception V3, VGG and ResNet, were trained and evaluated. It is founded Inception V3 outperforms all other architectures, with an accuracy of 99%, suggesting the deployment of this model in scalable diagnostics [8].

Finally an advanced research was done in Real-time cardiovascular risk assessment. It implemented reservoir computing, which enables continuous prediction by integrating time-series EMR data with structured features. The performance was enhanced as it uses fusion based modelling across all metrics such as ROC, F1 suggesting its deployment in emergency and telehealth environments [9].

2.2 Key ML Models Used in previous research:

Model	Strengths	Limitations
Logistic Regression	Interpretability	Poor with nonlinear data
Decision Trees	Easy to interpret, non-linear	Overfitting
Random Forest	Robust ensemble, high accuracy	Less interpretable
SVM	Handles high-dimensional data	Computationally expensive
Neural Networks	High accuracy, can handle complex patterns	Black-box nature
Gradient Boosting (XGBoost, LightGBM)	Best performance in many studies	Complex, needs tuning
Deep Learning (CNN, RNN)	Real-time & unstructured data	Requires large datasets

2.2.1 Common Features Used in CVD Prediction

Below are the key factors that most researches agree upon determining the risk of heart diseases.

- Age
- Gender
- Resting Blood Pressure
- Serum Cholesterol
- Chest Pain Type
- Fasting Blood Sugar
- ECG Results
- Maximum Heart Rate
- Exercise-induced Angina

- Smoking and Alcohol History
- Body Mass Index (BMI)

2.3 Neural Network (MLP):

Artificial Neural Networks (ANNs) are a group of machine learning architectures whose concept is based on how the human brain works. These networks are comprised of artificial neurons connected in layers that pass the input information through the weighted connections and activation functions as seen in Fig 1. The most impressive property of ANNs is that they can learn complex patterns and nonlinear connections in data, and therefore, the training of ANNs is most effective when it comes to tasks requiring classification, prediction, and pattern recognition, including those concerning the sphere of healthcare. Data in healthcare is usually large, multifaceted and noisy. Non-neural models, typically either based on rules or statistics, might not have the detail needed to handle such data well, but ANNs can infer using both structured (e.g. lab results, vitals) and unstructured data (e.g. clinical notes, imaging). These are scalable and adaptable, and hence they can have many medical applications.

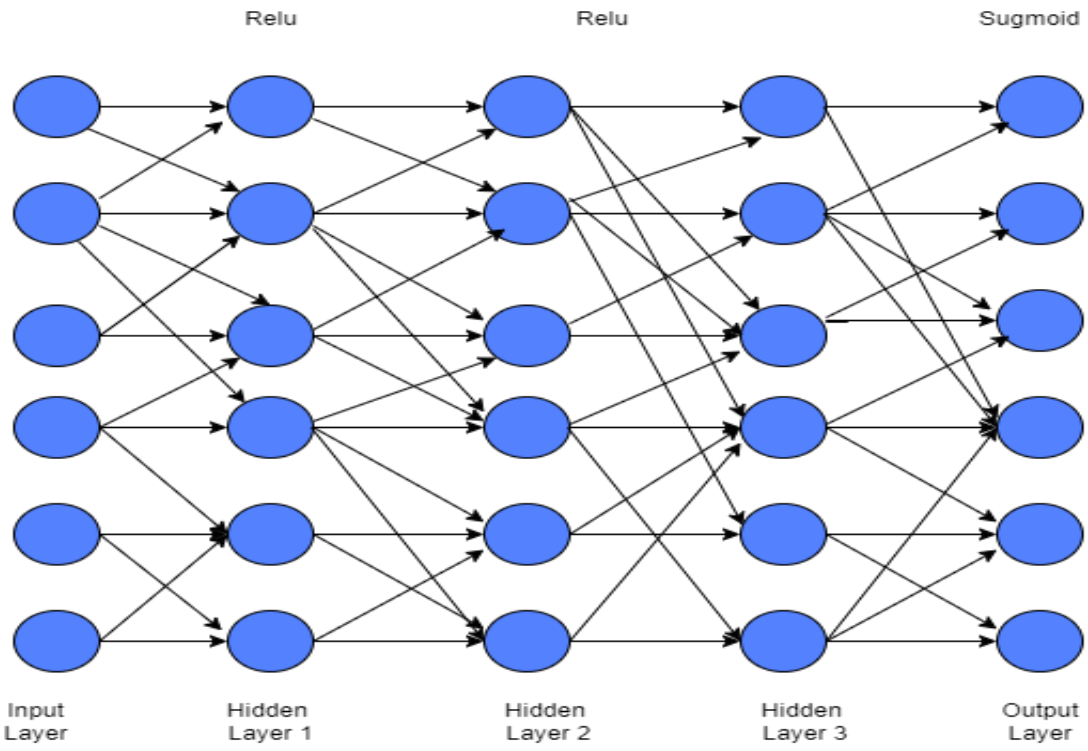


Fig 1: Working of Neural Network (MLP)

2.3.1 Advantages

1. **Adaptability:** One of the biggest specialisations of neural networks is that they learn and adapt easily with new data, which makes them much more flexible than traditional methods [10].
2. **Non-linear Modelling capability:** Complex non-linear relationships in data can be learned, which gives it an upper hand in processing complex medical data, where most of the traditional algorithms struggle [11].

3. **Fault Tolerance:** These models are generally immune to noisy and missing data, which makes them more reliable in real-world scenarios.
4. **Generalisation:** It can generalise to new data that they have not been trained on, making them versatile than traditional algorithms.

2.3.2 Disadvantages

1. **Black Box nature:** Interpretability is limited; they are often opaque and difficult to understand why they make specific predictions [12].
2. **Data requirements:** They require large and well-labelled datasets to avoid situations like overfitting and underperformance.
3. **Computational Costs:** Training these MLPs can be computationally very intense, requiring high-power GPUs.
4. **Hyperparameter Sensitivity:** Their performance highly relies on how effectively the hyperparameters, like learning rate, number of layers, and neurons, are tuned [13].

2.3.3 Key Applications in Health Care

1. Disease Diagnosis and Prediction
2. Individualised Treatment and Drug Development
3. Medical Image Segmentation and Classification
4. Robotics and Surgical Assistance

2.4 Ensemble Learning (Random Forest)

Ensemble learning is a machine learning method derived by uses a variety of models, sometimes called base learners or weak learners, and merges them to generate a more robust and precise forecasting model. The basic concept of ensemble methods is that individual models can have flaws or biases, whereas an averaging or combination of their outputs can achieve large performance improvements, minimise variance, and improve generalisability. Such a combined approach enables ensemble models to achieve better results than single learners do, especially on noisy or uneven datasets (for ex-Healthcare data). The Random Forest algorithm is one of the most popular ensemble methods. Random Forest constructs a set of decision trees during the training process and combines their predictions through majority vote (in the classification problem) or averages (in the regression problem) as seen in Fig 2. Diversity gives variety to each decision tree in the forest, training it on a randomly selected subset of the training data and features, which helps minimise the occurrence of overfitting, which is common in individual trees. This makes the model capable of generalising well to unforeseen data, which makes it more reliable and easier to interpret.

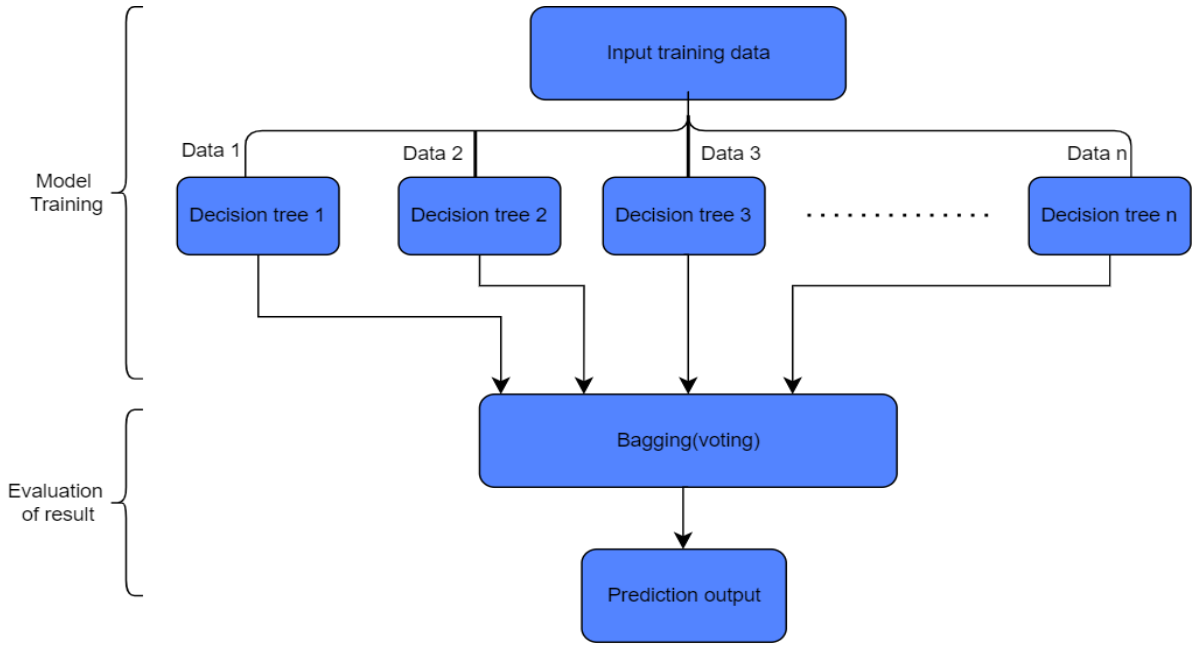


Fig 2: Working of Random Forest (Ensemble Learning)

2.4.1 Advantages

1. **High Accuracy:** Combines multiple decision trees to reduce error occurrence and improve prediction performance [14].
2. **Robustness to Overfitting:** By averaging multiple trees trained on different data samples, overfitting is reduced significantly.
3. **Features Importance:** Helps identify which features/variables are most influential in decision making, which aids interpretability [15].
4. **Handles missing Data and Outliers:** Performs well with partial data and adapts easily to unbalanced class distributions.

2.4.2 Disadvantages

1. **Computationally Intensive:** Training and prediction can be slow due to large datasets or many trees, which makes it a non-ideal solution for real real-time prediction system [16].
2. **Tuning Required:** While it works well, fine-tuning the number of trees, depth and other hyperparameters is required for optimal performance.
3. **Memory Usage:** It consumes a significant memory and storage, increasing with the number of trees [16].
4. **Less Transparency:** Even with higher interpretability than most of the deep learning models, it is still considered a black box compared to single decision trees.

2.4.3 Key Applications in Health Care:

1. Hypertension Prediction
2. Health Diagnosis and Classification
3. Selectivity of features and interpretability
4. Predictive Outcome of Patients

3. Dataset Description:

The dataset that we have used consists of 70,000 unique values. It consists of categorical target variable named cardio and other 12 unique attributes, which are as follows [17]:

We have three input features categorized as:

1. Objective features

These are the true values that are collected during physical checkup, such as:

Feature	Description
Age	This feature indicates the age of a person represented in days and has the type as integer.
Height	This feature gives the height of an individual in centimeters and the type is integer.
Weight	Provide the weight of an individual in kilograms and the type is float.
Gender	It is used in determining whether an individual is male or female and is categorical data so represented by 1 and 2.

2. Examination features

These are the results that we got after getting medically examined and they are as follows:

Feature	Description
Blood pressure	In this feature, ap_hi represents the upper blood pressure and ap_lo represents the lower blood pressure and both are of integer type.
Cholesterol	In this we have three categories such as normal, above normal, and well above normal represented by 1, 2 and 3 respectively.
Glucose	It is a categorical data where 1 is used for normal, 2 for above normal and 3 for above normal category.

3. Surjective feature

This includes the information given by the patient regarding their lifestyle.

Feature	Description
Smoking	It is of type binary that is 0 or 1, stating whether an individual smokes or not.
Alcohol intake	Binary type means 1 for the person who consumes alcohol and 0 if not.
Physical activity	Its type is binary, denoting whether an individual is physical exercise (1) or not (0).

4. System Architecture

The Fig. 3 represents the complete pipeline followed for cardiovascular disease prediction using machine learning techniques. The process begins from visualisation of the dataset, followed by pre-processing, model training and finally the comparison of output.

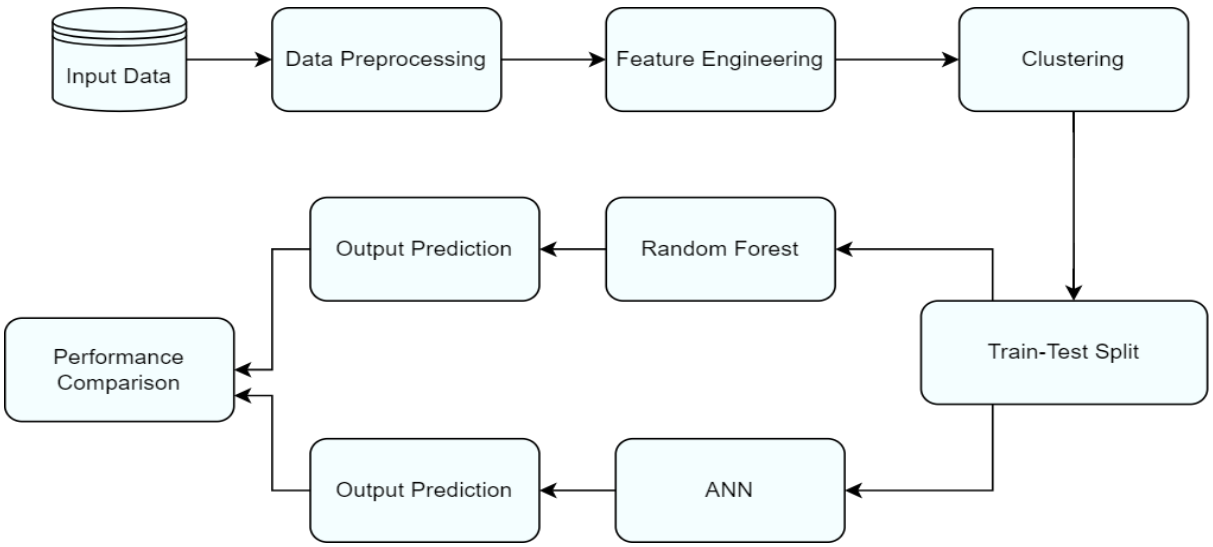


Fig 3 : Methodology diagram

5. Loading the Dataset and Initial Exploration

5.1 Library Imports

In this project we have used variety of machine learning libraries, i.e. from data pre-processing to visualisation and model evaluation. Below is the detailed explanation of the libraries used:

NumPy

It's a python library used for performing numerical operations. It allows support to multi-dimensional arrays and matrices, alongwith high level mathematical functions to operate on these arrays. In our project we have used Numpy for efficient array operations and numerical transformations.

Pandas

It is another library that is used for the analysis and manipulation of data. It consists of Series and DataFrame as its two primary data structures which are used for efficient data pre-processing, cleaning and exploration. In our project we have used it for reading and exploring the .csv file, performing feature engineering tasks.

Matplotlib/Seaborn

These libraries are used for data visualisation. Matplotlib works as a foundational plotting library, that provide precise control over visual elements such as grids, labels, colour schemas, which are particularly useful in plotting confusion matrix and ROC curves. On the other hand seaborn is built on top of Matplotlib, which simplifies the creation of aesthetically

pleasing plots. In our project it is used in visualization of data distribution, outlier detection, examination of feature relationships, through boxplots, pairplot, and heatmaps.

Scikit-learn

Scikit-learn is a popular machine learning library that offers a wide range of tools for data-preprocessing, model selection, training and evaluation. In our project it was used in splitting, scaling, encoding of dataset, Implementation of random forest and its evaluation through metrics likes accuracy, recall, precision and AUC.

TensorFlow & Keras

TensorFlow is a powerful open-source library and Keras is a high-level API build on its top are used for deep learning and large scale machine learning tasks. In this project, we used TensorFlow and Keras to construct and train an artificial neural network (ANN) with multiple layers, using early stopping to prevent overfitting.

KModes

KMode is a clustering algorithm which is used for handling categorical data. Unlike KMeans, which works on minimising the Euclidean distance, KMode uses a simple matching dissimilarity measure to group data points into clusters [18].

```
[ ] import numpy as np
    import pandas as pd
    from sklearn import metrics, preprocessing
    from sklearn.metrics import precision_score, recall_score, f1_score, roc_curve, roc_auc_score
    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.svm import SVC
    from kmodes.kmodes import KModes
    from sklearn.metrics import classification_report
    from sklearn.preprocessing import LabelEncoder
    import tensorflow as tf
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense, Dropout
    from tensorflow.keras.callbacks import EarlyStopping
    from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
    import matplotlib.pyplot as plt
    import seaborn as sns
```

5.2 Dataset Loading

This is the first step of our code implementation. In this step the dataset “cardio_train.csv” is loaded and is read using the Pandas DataFrame and is visualised by using the following code:

```
[ ] # read the data and show first 5 rows
    df = pd.read_csv("/content/drive/MyDrive/Datasets ML/cardio_train.csv",sep=";")
    df.head(5)
```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

In the dataset we can see 13 features, including the target feature cardio, where 1 represents whether patient has cardiovascular disease and 0 represents not.

5.3 Initial Data Inspection

To understand the structure of the dataset, we examined the data types and null values using .info( ).

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           70000 non-null  int64
 1   age          70000 non-null  int64
 2   gender       70000 non-null  int64
 3   height       70000 non-null  int64
 4   weight       70000 non-null  float64
 5   ap_hi        70000 non-null  int64
 6   ap_lo        70000 non-null  int64
 7   cholesterol  70000 non-null  int64
 8   gluc         70000 non-null  int64
 9   smoke        70000 non-null  int64
 10  alco         70000 non-null  int64
 11  active       70000 non-null  int64
 12  cardio       70000 non-null  int64
dtypes: float64(1), int64(12)
memory usage: 6.9 MB
```

This revealed that our dataset is clean and consists of no missing values. Most of the features ae numerical and categorical encoded as integers.

5.4 Exploratory Data Analysis (EDA)

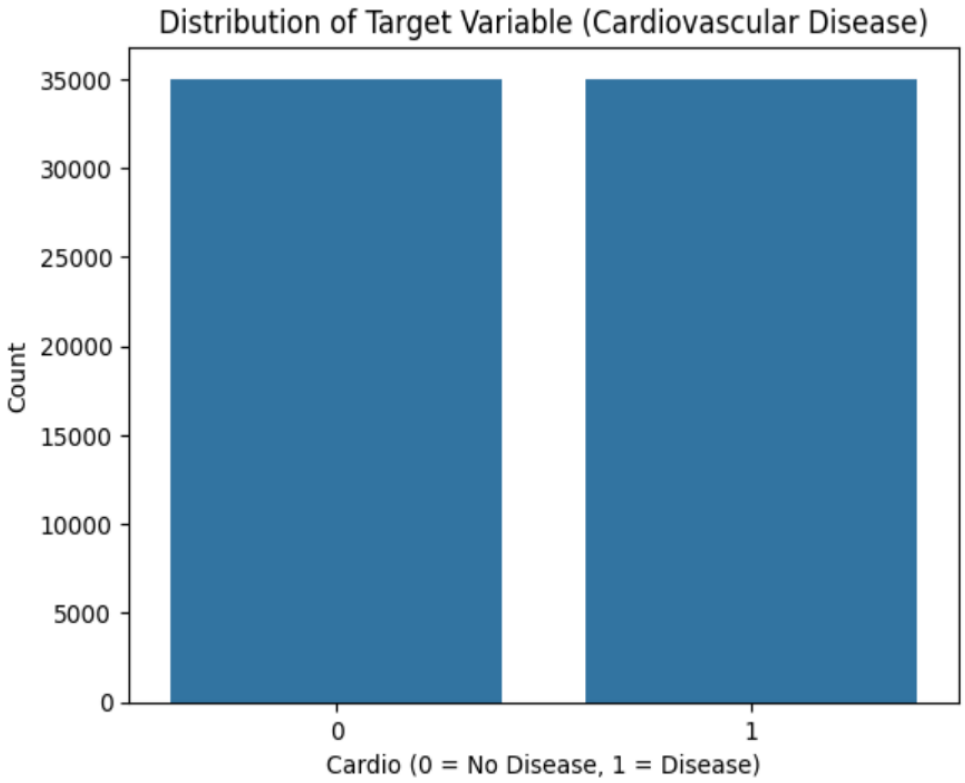
The objective of Exploratory Data Analysis (EDA) is to uncover the underlying patterns, identify outliers present in the dataset, and form hypothesis that may influence the model selection. In this step data is visualised using different types of plots to visualise the relation between features and target “cardio”.

5.4.1 Target Variable Distribution

This done to check whether there is a balance between the two classes of target feature “cardio”, where 0 represents no cardio disease and 1 represents cardio disease. In the below countplot we can see target variable has almost equal no of both classes, which is advantageous during model training as it reduces bias towards the majority class. Hence due balance clases there is also no requirement of using class imbalance technique such as SMOTE.

```
[ ] # Count the number of samples per class
    cardio_counts = df['cardio'].value_counts()
    print("Cardio Class Counts:\n", cardio_counts)
    # Plot the class distribution
    sns.countplot(x='cardio', data=df)
    plt.title('Distribution of Target Variable (Cardiovascular Disease)')
    plt.xlabel('Cardio (0 = No Disease, 1 = Disease)')
    plt.ylabel('Count')
    plt.show()
```

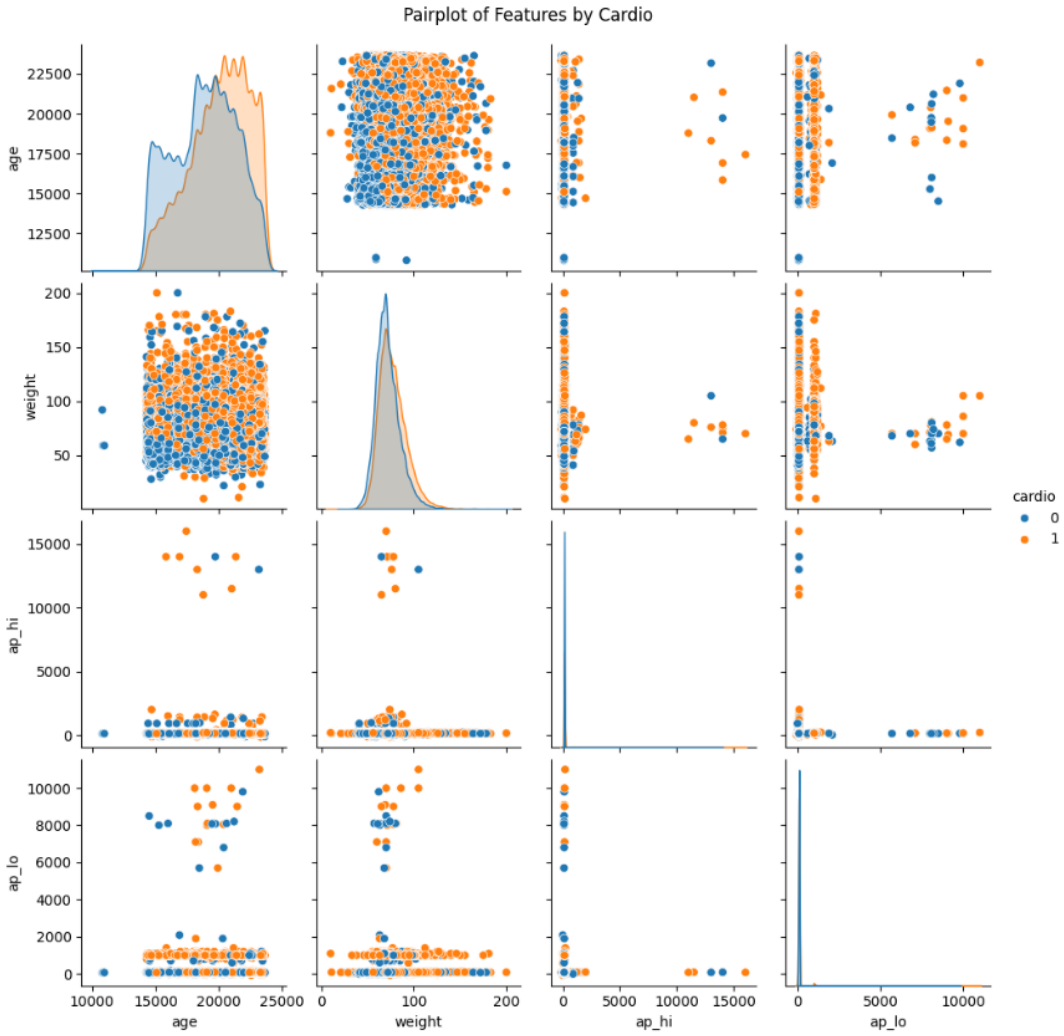
```
Cardio Class Counts:
 cardio
0    35021
1    34979
Name: count, dtype: int64
```



5.4.2 Pairplot Analysis

Below Pairplot is used to visualise the pairwise relationships between the key features such as age, weight, ap_hi and ap_lo across cardiovascular disease target classes. In its diagonal it reveals individual diagnosed with disease is slightly higher in age and weight distributions compared to those without disease. In the other scatter plots we can see higher values of systolic and diastolic blood pressure appear more frequently in the diseased group. These plots suggests us that age, weight, ap_hi and ap_lo can play a significant role cardiovascular disease prediction. Moreover it also helps in identifying potential outliers and overlapping region between the classes, which can be used for feature selection and improving model performance.

```
[ ]  # Pairplot of selected features against target
     sns.pairplot(df[['age', 'weight', 'ap_hi', 'ap_lo', 'cardio']], hue='cardio')
     plt.suptitle("Pairplot of Features by Cardio", y=1.02)
     plt.show()
```



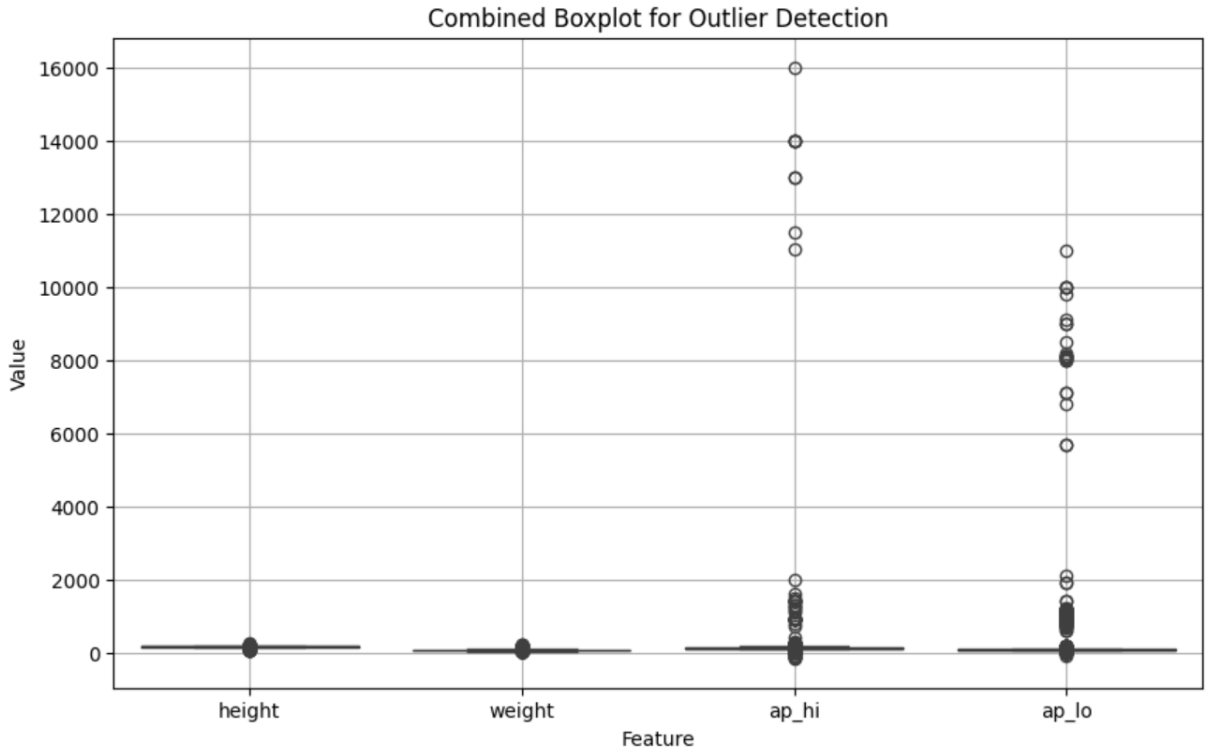
5.4.3 Outlier Detection via Boxplots

To identify the outliers in numerical features such as height, weight, ap_hi and ap_lo, we have drawn the below boxplot.

```
[ ]  # Select relevant columns
     outlier_cols = ['height', 'weight', 'ap_hi', 'ap_lo']

     # Melt the DataFrame to long format
     df_melted = df[outlier_cols].melt(var_name='Feature', value_name='Value')

     # Create a single boxplot
     plt.figure(figsize=(10, 6))
     sns.boxplot(x='Feature', y='Value', data=df_melted)
     plt.title('Combined Boxplot for Outlier Detection')
     plt.grid(True)
     plt.show()
```



In the above we can see height and weight has very slight spread, with some outliers due to incorrect entries. Other features such as ap_hi and ap_lo has visible outliers with biologically implausible values greater than 10,000.

6. Data Pre-processing

6.1 Removal of Outlier

After identification of outliers through box plot, the next step to remove these impractical values using Interquantile Range (IQR) approach [19]. In this method the data points lying outside the 2.5th to 97.5th percentile range is removed for selected continuous features. This techniques keeps the central 95% of the data in each variable by removing the 5%, i.e. 2.5% from both sides.

```
[ ]  df.drop(df[(df['height'] > df['height'].quantile(0.975)) | (df['height'] < df['height'].quantile(0.025))].index,inplace=True)
     df.drop(df[(df['weight'] > df['weight'].quantile(0.975)) | (df['weight'] < df['weight'].quantile(0.025))].index,inplace=True)
     df.drop(df[(df['ap_hi'] > df['ap_hi'].quantile(0.975)) | (df['ap_hi'] < df['ap_hi'].quantile(0.025))].index,inplace=True)
     df.drop(df[(df['ap_lo'] > df['ap_lo'].quantile(0.975)) | (df['ap_lo'] < df['ap_lo'].quantile(0.025))].index,inplace=True)
     len(df)

60142
```

Then we use the below code to check the cases that consists the value of ap_lo higher than ap_hi, as such type of records can lead to medical inconsistency.

```
[ ]  df[df['ap_lo']> df['ap_hi']].shape[0] # To check number of cases where diastolic pressure is higher than systolic?

0
```

6.2 Feature Engineering

In Feature engineering we transforms raw data into meaningful variables that better represent the underlying patterns and relationships within the data [20]. For this cardiovascular disease prediction project, we have derived several categorical features to enhance model performance and interpretability.

6.2.1 Age Transformation and Grouping

The original age feature is in days, which is not that intuitive for interpretation. So we have converted it into years using below formula.

```
[ ]  df['age'] = (df['age'] / 365).round().astype('int')  # Transformation: Converting age from days to years
     print(df.head())
```

```
    age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  \
0   50       2     168    62.0    110     80            1     1      0     0
1   55       1     156    85.0    140     90            3     1      0     0
2   52       1     165    64.0    130     70            3     1      0     0
3   48       2     169    82.0    150    100            1     1      0     0
4   48       1     156    56.0    100     60            1     1      0     0

   active  cardio
0       1       0
1       1       1
2       0       1
3       1       1
4       0       0
```

To better capture age-related trends and allow models to treat age as a categorical feature, we further grouped the age column into seven equal-width bins representing 5-year intervals:

```
[ ]  # Categorizing features
     # # Define the bin edges and labels
     age_edges = [30, 35, 40, 45, 50, 55, 60, 65]
     age_labels = [0, 1, 2, 3, 4, 5, 6]

     #  bin in  5 years span
     df['age_group'] = pd.cut(df['age'], bins=7, labels=range(7), include_lowest=True,right=True)
     df.head()
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	age_group
0	50	2	168	62.0	110	80	1	1	0	0	1	0	3
1	55	1	156	85.0	140	90	3	1	0	0	1	1	4
2	52	1	165	64.0	130	70	3	1	0	0	0	1	4
3	48	2	169	82.0	150	100	1	1	0	0	1	1	3
4	48	1	156	56.0	100	60	1	1	0	0	0	0	3

This transformation helps in discretizing continuous variables and reducing the complexity for algorithms that benefit from categorical inputs.

6.2.2 Body Mass Index (BMI)

Another well-established indicator of body fat and health status feature that we created is Body Mass Index [20]. It was calculated using the standard formula:

```
df['bmi'] = df['weight']/((df['height']/100)**2)
df.head()

bmiMin = int(df['bmi'].min())
bmiMax = int(df['bmi'].max())

print(bmiMin, bmiMax)

df['bmi'] = pd.cut(df['bmi'], bins=6, labels=range(6), right=True, include_lowest=True)

df.head()


df["bmi"].value_counts(normalize=True)
```

```
16 46
```

bmi	proportion
1	0.461325
2	0.330202
3	0.133068
0	0.038193
4	0.033554
5	0.003658

dtype: float64

Further computed BMI values were then binned in 6 categories, which categorically represents underweight, normal, overweight, and obese groups.

This binning makes it easier for the model to detect patterns present in different BMI ranges rather than treating it as a raw continuous variable.

6.2.3 Mean Arterial Pressure (MAP)

Mean Arterial Pressure is a composite feature that combines systolic (ap_hi)) and diastolic (ap_lo) blood pressure readings into a single clinically significant value [21].

Like BMI, the MAP values were also binned into 6 categories and simplifies the feature space.

```
[ ]  df['map'] = ((2* df['ap_lo']) + df['ap_hi']) / 3

     mapMin = int(df['map'].min())
     mapMax = int(df['map'].max())

     print(mapMin, mapMax)

     df['map'] = pd.cut(df['map'], bins=6, labels=range(6), right=True, include_lowest=True)

     df.head()
```

```
73 121
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	age_group	bmi	map
0	50	2	168	62.0	110	80	1	1	0	0	1	0	3	1	2
1	55	1	156	85.0	140	90	3	1	0	0	1	1	4	3	4
2	52	1	165	64.0	130	70	3	1	0	0	0	1	4	1	2
3	48	2	169	82.0	150	100	1	1	0	0	1	1	3	2	5
4	48	1	156	56.0	100	60	1	1	0	0	0	0	3	1	0

This transformation captures blood pressure trends along with reducing the effect of noise in raw systolic and diastolic readings.

6.3 Dropping Raw Numerical Columns

After the creation of categorised versions of age, BMI and MAP, we have dropped the original numerical columns, i.e. age , height, weight, ap_hi and ap_lo from the Data Frame. This step ensured the dataset was clean, categorical, and ready for encoding and modeling.

```
[ ]  # Drop Numerical Column
     df_og=df

     df=df.drop(['height','weight','ap_hi','ap_lo','age'],axis=1)

     df.head()
```

6.4 Categorical Encoding

As numerical format features are generally required by most machine learning algorithms. But our dataset contains categorical variables (both originally and as a result of feature engineering), so we applied label encoding to convert these categories into integer representations.

Label encoding assigns a unique numerical value to each category within a feature. This transformation is especially useful for algorithms such as Random Forest as it handle ordinal relationships and split based on categorical values [22]. Below is the result of Label encoding on original and feature engineered one that is age_group, BMI and MAP.

```
[ ]  le = preprocessing.LabelEncoder()
     df = df.apply(le.fit_transform)
     df.describe()
```

	gender	cholesterol	gluc	smoke	alco	active	cardio	age_group	bmi	map
count	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000
mean	0.347311	0.350953	0.220229	0.085631	0.051877	0.803648	0.488228	4.042233	1.673440	2.359449
std	0.476120	0.670076	0.567607	0.279820	0.221781	0.397241	0.499866	1.377070	0.898707	1.186906
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	3.000000	1.000000	2.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	4.000000	2.000000	2.000000
75%	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	5.000000	2.000000	3.000000
max	1.000000	2.000000	2.000000	1.000000	1.000000	1.000000	1.000000	6.000000	5.000000	5.000000

7. Clustering Analysis using KModes

In addition to supervised machine learning models, we have also applied unsupervised learning through clustering to explore natural groupings within the data. Since our dataset is fully categorical after pre-processing, we used the KModes algorithm, which is specifically designed for clustering categorical data [18].

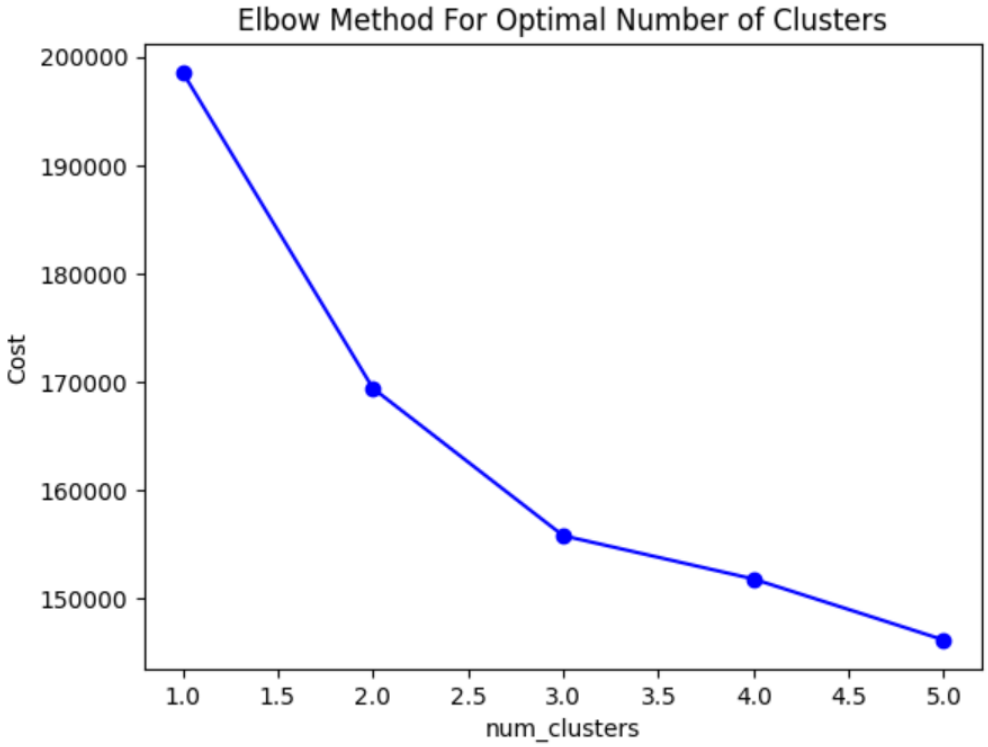
7.1 Elbow Method for Optimal Clusters

To determine the optimal number of clusters, we implemented Elbow Method, which involves plotting the cost (a measure of dissimilarity) against the number of clusters. The idea is to

identify the "elbow point" where the rate of cost reduction sharply decreases, indicating a good trade-off between complexity and clustering quality.

```
[ ]  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_)
```

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



From the elbow curve, we can see a noticeable drop in cost occurs between 1st and 2nd clusters, after which the curve begins to flatten. This indicates that 2 clusters offer a good balance between underfitting and overfitting.

7.2 KModes Clustering

Based on the Elbow Method, we have selected 2 clusters and built the KModes model accordingly:

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

```
array([0, 1, 0, ..., 0, 1, 0], dtype=uint16)
```

Each record in the dataset was assigned a cluster label of 0 or 1, representing the cluster to which it was most similar based on mode matching of categorical features.

7.3 Adding Cluster Information to Dataset

Now we have appended the predicted cluster labels to the original DataFrame for further analysis.

```
[ ]  # Adding clusters column in DF
     df.insert(0,"clusters",clusters,True)
     df.head()
```

	clusters	gender	cholesterol	gluc	smoke	alco	active	cardio	age_group	bmi	map
0	0	1	0	0	0	0	1	0	3	1	2
1	1	0	2	0	0	0	1	1	4	3	4
2	0	0	2	0	0	0	0	1	4	1	2
3	1	1	0	0	0	0	1	1	3	2	5
4	0	0	0	0	0	0	0	0	3	1	0

This allowed us to analyse the relationship between clusters and the target variable cardio, which helps us to understand whether the natural groupings in the data correspond to disease presence.

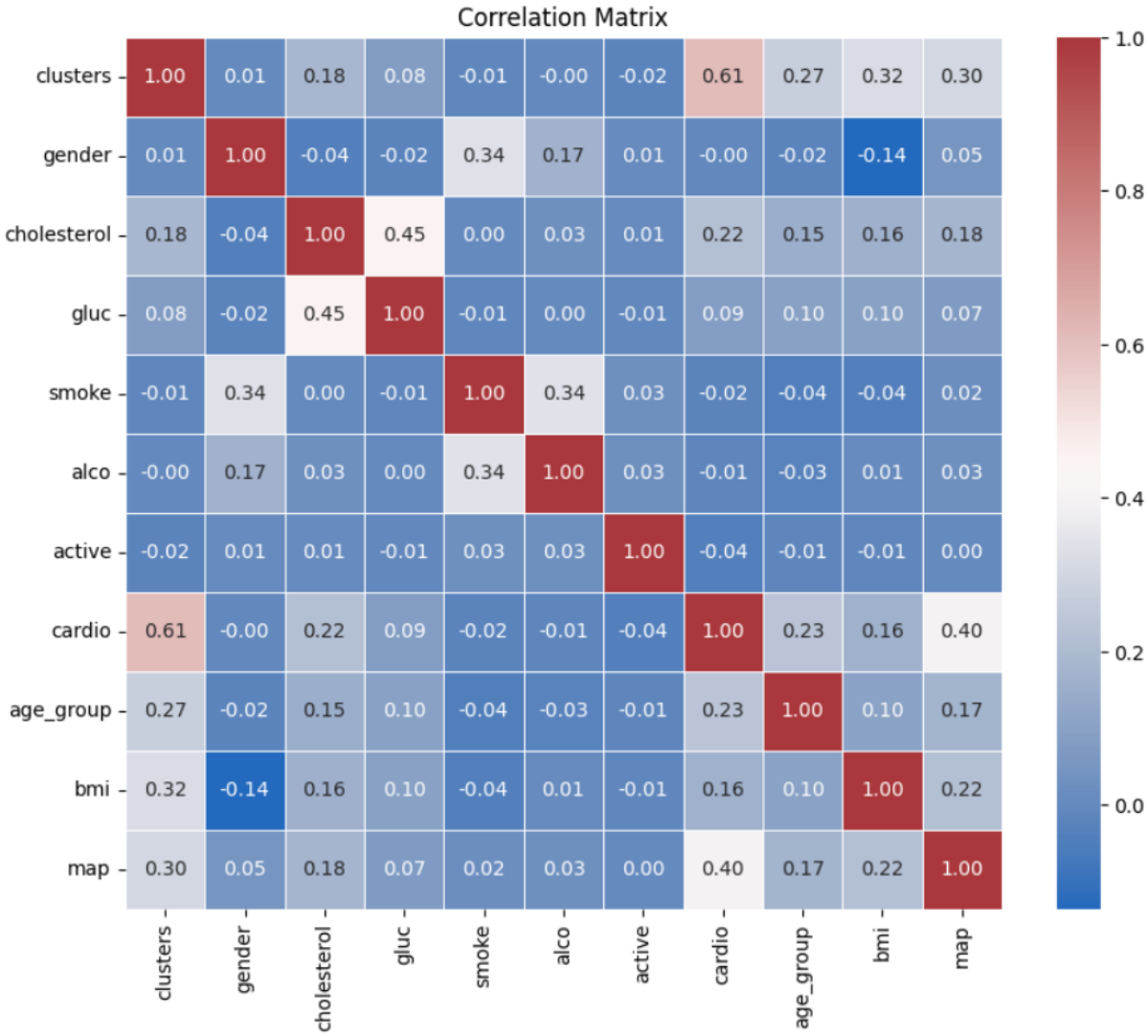
7.4 Cluster-Based Visualizations and Insights

To better understand the patterns discovered through KModes clustering, we performed additional analysis by visualizing how clusters relate to the target variable i.e. cardio and other key features.

7.4.1 Correlation Matrix Analysis

We began by plotting a correlation heatmap to examine the relationships between all categorical features, including the newly added clusters column.

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

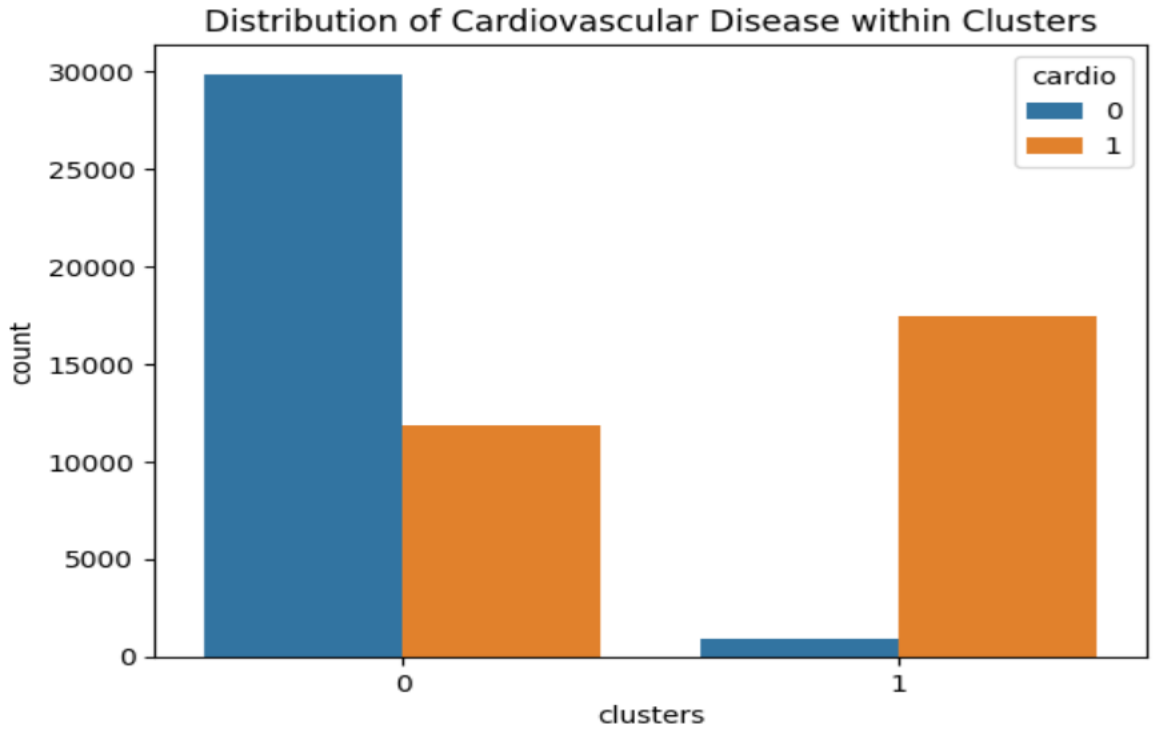


From the correlation matrix we can see that there is a moderate positive correlation (0.61) of clusters with cardio. This indicates that Kmodes clustering has effectively separated groups with distinct cardiovascular disease prevalence. Alongwith that other categorical features such as BMI, age_group and MAP also shows positive correlations with cardio, which reaffirms their clinical relevance. While features such as active, smoke, and alco shows very weak or negligible correlation with the target feature (cardio).

7.4.2 Distribution of Cardiovascular Disease Across Clusters

To further evaluate the utility of clustering, we have visualized the presence of cardiovascular disease (cardio) across the two identified clusters.

```
[ ]  # Distribution of cardio in clusters
     sns.countplot(x='clusters', hue='cardio', data=df)
     plt.title('Distribution of Cardiovascular Disease within Clusters')
     plt.show()
```



We can see that in cluster 0, majority of patients were without cardiovascular disease (label 0) while cluster 1 predominantly contains patients with cardiovascular disease (label 1).

This visual clearly shows that the clustering approach aligned well with the target class, indicating KModes successfully distinguished two meaningful health segments within the population. Further we will drop insignificant feature such as, gender and alco as they do not play a key role in determining the target.

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

     x.head()
```

	clusters	cholesterol	gluc	smoke	active	age_group	bmi	map
0	0	0	0	0	1	3	1	2
1	1	2	0	0	1	4	3	4
2	0	2	0	0	0	4	1	2
3	1	0	0	0	1	3	2	5
4	0	0	0	0	0	3	1	0

7.5 Train-Test Split

To train and evaluate our machine learning models effectively, we divided the dataset into training and testing subsets using an 80-20 split. This is a standard practice to ensure that models are trained on one portion of the data and evaluated on another, unseen portion to estimate generalization performance.

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

8. Model Training

8.1 Random Forest Classifier

The first method that we implemented is random forest. In this method we have build model using 200 decision trees with max_depth = 20, to avoid overfitting. Further the tree growth is controlled by min_samples_split and min_samples_leaf. For selecting the subset of features at each split we used sqrt method. The model is trained and tested on 80% and 20% of the dataset respectively.

```
[ ]  from sklearn.metrics import accuracy_score
     rfModel = RandomForestClassifier(
         n_estimators=200,
         max_depth=20,
         min_samples_split=5,
         min_samples_leaf=2,
         max_features='sqrt',
         random_state=1
     )

     # Train the model
     rfModel.fit(x_train, y_train)

     # Predict on test data
     rf_pred = rfModel.predict(x_test)
     # Accuracy
     rf_accuracy = accuracy_score(y_test, rf_pred) * 100
     print(f"Random Forest Accuracy: {rf_accuracy:.2f}%")
```

```
Random Forest Accuracy: 85.34%
```

8.2 Artificial Neural Network

The other method we have implemented Multi-layer feedforward neural network using Keras Sequential API and trained model for 100 epochs with batch size 32. It consists of three hidden layers with 120, 64, 32 neurons respectively with ReLu as an activation function. The model was compiled with the Adam optimizer to adaptively control the learning rate and binary_crossentropy as Loss Function is used to measure the error between predicted probabilities and true binary labels in classification tasks. Further early stopping is applied by monitoring the validation loss to prevent overfitting. Finally the output layer with sigmoid activation function is used to produce probability scores for binary prediction.

```
[ ]  # Define ANN architecture
     model = Sequential()
     model.add(Dense(128, input_dim=x_train.shape[1], activation='relu'))
     model.add(Dropout(0.3))  # prevent overfitting
     model.add(Dense(64, activation='relu'))
     model.add(Dropout(0.2))
     model.add(Dense(32, activation='relu'))
     model.add(Dense(1, activation='sigmoid'))  # Binary classification

     # Compile model
     model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass ar
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```python
# Use early stopping to avoid overfitting
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train
history = model.fit(
    x_train, y_train,
    validation_split=0.2,
    epochs=100,
    batch_size=32,
    callbacks=[early_stop],
    verbose=1
)
```

```
Epoch 1/100
1203/1203 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8479 - loss: 0.3043 - val_accuracy: 0.8542 - val_loss: 0.2938
Epoch 2/100
1203/1203 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.8535 - loss: 0.2974 - val_accuracy: 0.8553 - val_loss: 0.2894
Epoch 3/100
1203/1203 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.8541 - loss: 0.2970 - val_accuracy: 0.8552 - val_loss: 0.2883
Epoch 4/100
1203/1203 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8521 - loss: 0.2994 - val_accuracy: 0.8560 - val_loss: 0.2887
Epoch 5/100
1203/1203 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8494 - loss: 0.3036 - val_accuracy: 0.8561 - val_loss: 0.2875
Epoch 6/100
1203/1203 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.8517 - loss: 0.3008 - val_accuracy: 0.8562 - val_loss: 0.2866
Epoch 7/100
1203/1203 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.8554 - loss: 0.2955 - val_accuracy: 0.8561 - val_loss: 0.2895
Epoch 8/100
1203/1203 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.8503 - loss: 0.2978 - val_accuracy: 0.8551 - val_loss: 0.2898
```

```python
# Predict on test set
y_pred_prob = model.predict(x_test)
y_pred = (y_pred_prob > 0.5).astype(int)

# Accuracy
ann_accuracy = accuracy_score(y_test, y_pred) * 100
print(f"ANN Accuracy: {ann_accuracy:.2f}%")
```

```
376/376 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
ANN Accuracy: 85.75%
```

9. Results Evaluation

9.1 Evaluation of Random Forest

To evaluate the performance of the Random Forest classifier in distinguishing between patients with and without cardiovascular disease, we analysed both the classification report and confusion matrix.

A. Classification Report:

In below classification report we can that random forest has overall accuracy of 85% with precision, recall, F1-score of 0.87, 0.82 and 0.84 respectively for the prediction of disease. Overall random forest performs well across both classes.

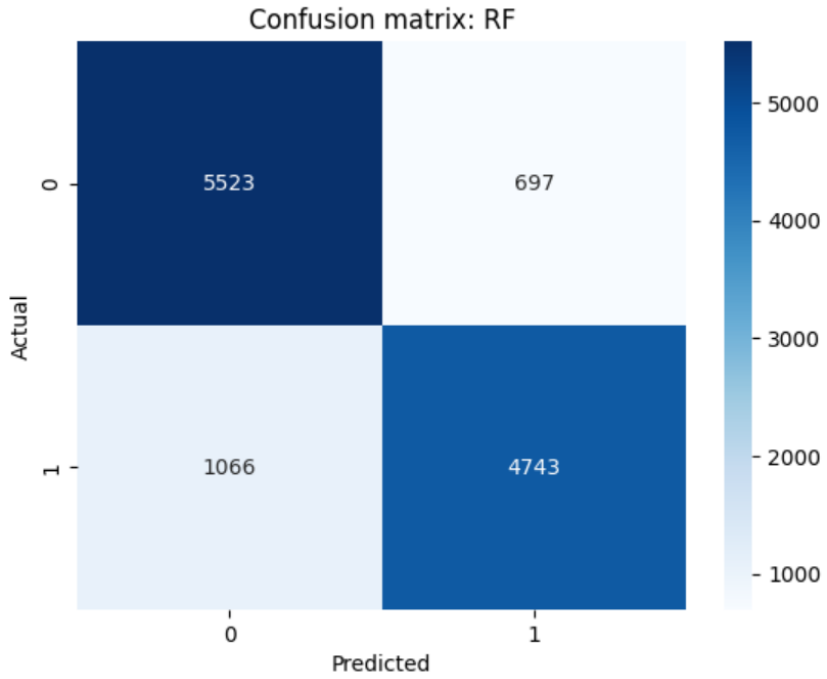
```
Classification Report for RF:
                precision    recall  f1-score   support

           0       0.84      0.89      0.86      6220
           1       0.87      0.82      0.84      5809

    accuracy                           0.85     12029
   macro avg       0.86      0.85      0.85     12029
weighted avg       0.85      0.85      0.85     12029
```

B. Confusion Matrix

The below confusion matrix highlights that the model is effective at capturing true positives while maintaining a relatively low false positive rate. However, the 1066 false negatives suggest that some actual disease cases are being missed, which is critical in medical diagnosis.



9.2 Evaluation of Artificial Neural Network (ANN)

The performance of the ANN model was assessed using standard evaluation metrics. These metrics provide insights into how effectively the model distinguishes between healthy individuals and those with cardiovascular disease.

A. Classification Report

In the below classification report we can see the model has an overall accuracy of 86%, with Precision, Recall and F1 sore be 0.88, 0.82 and 0.85 respectively. Overall ANN slightly outperforms the Random Forest model in terms of overall F1-score and accuracy.

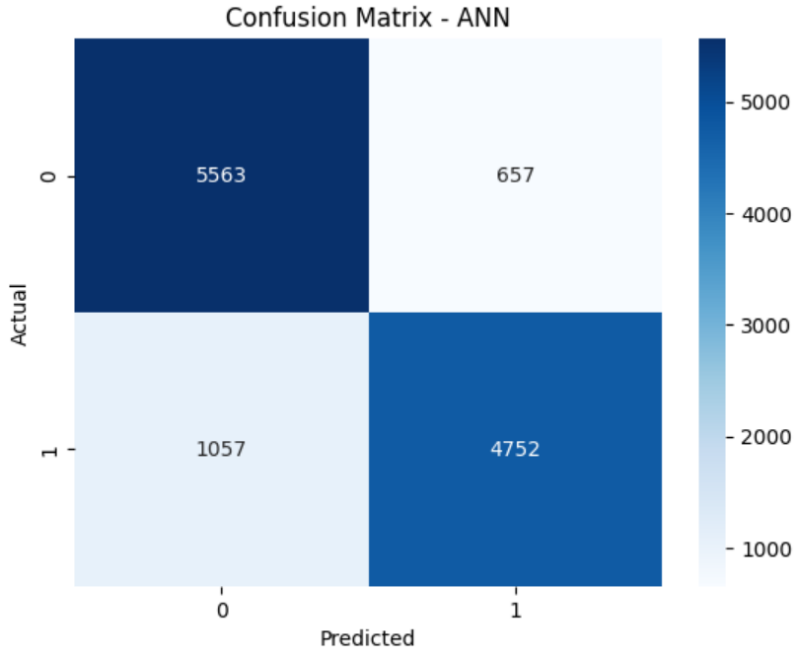
```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.89      0.87      6220
           1       0.88      0.82      0.85      5809

    accuracy                           0.86     12029
   macro avg       0.86      0.86      0.86     12029
weighted avg       0.86      0.86      0.86     12029
```

B. Confusion Matrix Interpretation

Like the Random Forest model, the ANN also maintains a low false positive rate. Importantly, it reduces false negatives slightly more effectively, making it a better candidate for medical prediction tasks where identifying disease cases is critical.

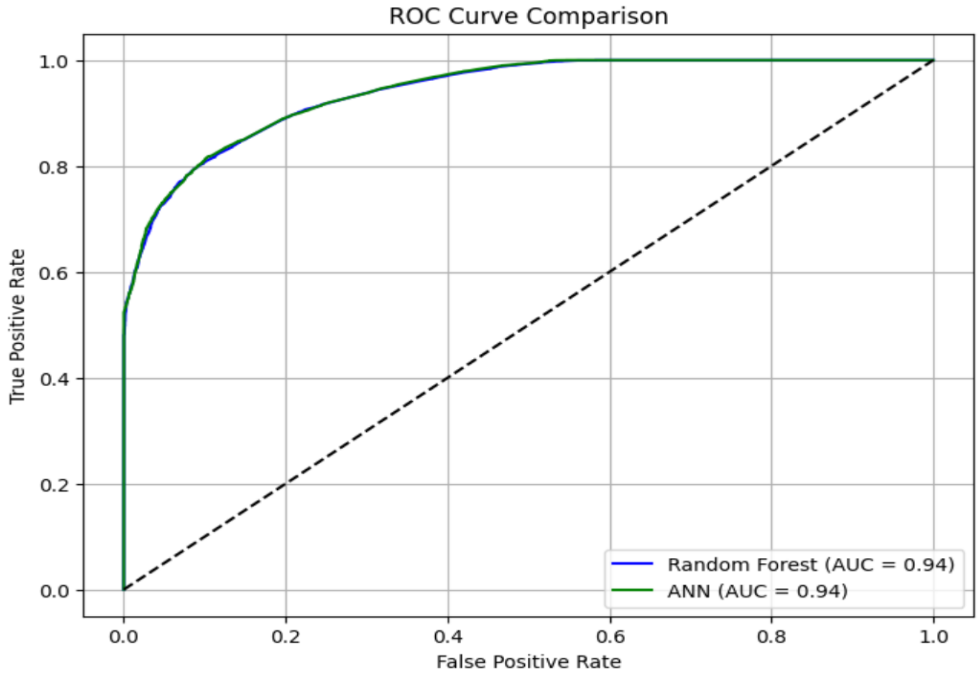


10. Model Performance Comparison

To compare the predictive performance of the Random Forest and Artificial Neural Network models, we have utilized classification metrics and ROC curve analysis [23]. The final ROC curve suggests that both models achieved an excellent Area Under the Curve (AUC) score of 0.94, indicating strong capability in distinguishing between patients with and without cardiovascular disease. While the curves are nearly identical, the ANN model slightly outperformed the Random Forest in terms of overall classification accuracy achieving 85.75% compared to 85.34% by RF. Further in ANN, we can see a high F1 score for both classes along with fewer false positives and false negatives in the confusion matrix. Although

both models are highly effective, the ANN demonstrates slightly superior predictive consistency and generalization, making it the preferred choice for this task.

```
[ ]  plt.figure(figsize=(8, 6))
     plt.plot(fpr_rf, tpr_rf, label=f"Random Forest (AUC = {auc_rf:.2f})", color='blue')
     plt.plot(fpr_ann, tpr_ann, label=f"ANN (AUC = {auc_ann:.2f})", color='green')
     plt.plot([0, 1], [0, 1], 'k--')
     plt.xlabel("False Positive Rate")
     plt.ylabel("True Positive Rate")
     plt.title("ROC Curve Comparison")
     plt.legend(loc="lower right")
     plt.grid(True)
     plt.show()
```



11. Conclusion

This research demonstrates the effectiveness of machine learning in cardiovascular disease prediction by drawing comparisons between the Random Forest model and the Artificial Neural Network (ANN) model. Both models exhibited strong predictive performance, but ANN slightly outperforms Random Forest. Hence ANN is best suited for real world applications. This project also signifies the importance of data processing and feature engineering techniques in boosting the model overall performance. Clustering analysis using KModes further supported the structure of dataset and revealed natural groupings aligned with disease labels.

In future, hybrid models can be implement which also uses other modalities in prediction of heart diseases. Explainable AI techniques such as SHAP and LIME can also play a key role in enhancing model Interpretability [24]. Future integration with wearable devices can also help in real time detection.

References

[1] “Cardiovascular diseases (CVDs),” World Health Organization, 11 Jun. 2021. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[2] H. V. Bhagat, M. Singh and others, “A Machine Learning Model for the Early Prediction of Cardiovascular Disease in Patients,” in *2023 Second International Conference on Advances in Computational Intelligence and Communication (ICACIC)*, IEEE, 2023, pp. 1--5.

[3] A. Nikam, S. Bhandari, A. Mhaske and S. Mantri, “Cardiovascular disease prediction using machine learning models,” in *2020 IEEE Pune section international conference (PuneCon)*, IEEE, 2020, pp. 22--27.

[4] T. Lin and M. Fan, “Cardiovascular Disease Risk Prediction Based on Deep Feature Extraction,” in *2024 9th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, vol. 9, IEEE, 2024, pp. 210--214.

[5] T. Soni, D. Gupta and M. Uppal, “Employing XGBoost Algorithm for Accurate Prediction of Cardiovascular Disease and its Risk Factors,” in *2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC)*, IEEE, 2024, pp. 353--358.

[6] S. Amal, L. Safarnejad, J. A. Omiye, I. Ghanzouri, J. H. Cabot and E. G. Ross, “Use of multi-modal data and machine learning to improve cardiovascular disease care,” *Frontiers in cardiovascular medicine,* vol. 9, p. 840262, 2022.

[7] M. Bhagawati and S. Paul, “Generative Adversarial Network-based Deep Learning Framework for Cardiovascular Disease Risk Prediction,” in *2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT)*, IEEE, 2024, pp. 1--4.

[8] L. L. Sowjanya, K. Kowshya, M. S. Roshini, B. Krishna, V. K. K. Kolli and others, “An Improved Inception V3 Deep learning model for Cardiovascular Disease Prediction,” in *2025 International Conference on Emerging Systems and Intelligent Computing (ESIC)*, IEEE, 2025, pp. 78--83.

[9] S. Sadasivuni, V. Damodaran, I. Banerjee and A. Sanyal, “Real-time prediction of cardiovascular diseases using reservoir-computing and fusion with electronic medical record,” in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, IEEE, 2022, pp. 58--61.

[10] IBM, “What is a Neural Network?,” 06 Oct. 2021. [Online]. Available: https://www.ibm.com/think/topics/neural-networks.

[11] M. M. Mijwel, “Artificial neural networks advantages and disadvantages,” *Mesopotamian Journal of Big Data,* vol. 2021, pp. 29--31, 2021.

[12] Z. C. Lipton, “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.,” *Queue,* vol. 16, no. 3, pp. 31--57, 2018.

[13] N. Donges, “4 Disadvantages of Neural Networks,” built In, 15 Aug. 2023. [Online]. Available: https://builtin.com/data-science/disadvantages-neural-networks.

[14] IBM, “Random Forest,” Ibm.com, 20 Oct. 2021. [Online]. Available: https://www.ibm.com/think/topics/random-forest.

[15] N. Donges, “Random Forest: a Complete Guide for Machine Learning,” Built in, 22 Jul. 2021. [Online]. Available: https://builtin.com/data-science/random-forest-algorithm.

[16] GeeksforGeeks, “What are the Advantages and Disadvantages of Random Forest?,” GeeksforGeeks, 15 Feb. 2024. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/what-are-the-advantages-and-disadvantages-of-random-forest/.

[17] S. Ulianova, “Cardiovascular Disease dataset,” www.kaggle.com, 2019. [Online]. Available: https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset.

[18] M. A. Carreira-Perpinn and W. Wang, “The K-modes algorithm for clustering,” *arXiv preprint arXiv:1304.6478,* 2013.

[19] N. Acharya, “Understanding Outlier Removal Using Interquartile Range (IQR),” Medium, 13 Feb. 2024. [Online]. Available: https://medium.com/%40nirajan.acharya777/understanding-outlier-removal-using-interquartile-range-iqr-b55b9726363e.

[20] J. Heaton, “An empirical analysis of feature engineering for predictive modeling,” in *SoutheastCon 2016*, IEEE, 2016, pp. 1--6.

[21] C. M. Bhatt , P. Patel, T. Ghetia and P. L. Mazzeo, “Effective heart disease prediction using machine learning techniques,” *Algorithms,* vol. 16, no. 2, p. 88, 2023.

[22] W. Zhu, R. Qiu and Y. Fu, “Comparative study on the performance of categorical variable encoders in classification and regression tasks,” *arXiv preprint arXiv:2401.09682,* 2024.

[23] A. P. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” *Pattern recognition,* vol. 30, no. 7, pp. 1145--1159, 1997.

[24] A. M. Salih, Z. Raisi-Estabragh, I. B. Galazzo, P. Radeva, S. E. Petersen, K. Lekadir and G. Menegaz, “A perspective on explainable artificial intelligence methods: SHAP and LIME,” *Advanced Intelligent Systems,* vol. 7, no. 1, p. 2400304, 2025.