A black background with white text

Description automatically generated with low confidence

MSc Data Science Project

7PAM2002-0509-2024

Department of Physics, Astronomy and Mathematics

**Data Science FINAL PROJECT REPORT**

**Project Title:**

Cardiovascular Disease Prediction Using Feature Selection and Ensemble Learning Models

**Student Name and SRN:**

HARI PRASAD GHANAPURAM and 22051910

**Supervisor:** Dr. Hyungrok Kim

**Date Submitted:** 24/08/2025

**Word Count:** 5452

**GitHub Link:** [**https://github.com/harighanapuram1329/Cardiovascular-Disease-Prediction-Using-Feature-Selection-and-Ensemble-Learning-Models/tree/main**](https://github.com/harighanapuram1329/Cardiovascular-Disease-Prediction-Using-Feature-Selection-and-Ensemble-Learning-Models/tree/main)

# DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science **in Data Science** at the University of Hertfordshire.

I have read the detailed guidance to students on academic integrity, misconduct and plagiarism information at [Assessment Offences and Academic Misconduct](https://ask.herts.ac.uk/assessment-offences-and-academic-misconduct) and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: HARI PRASAD GHANAPURAM

Student Name signature: HARI PRASAD GHANAPURM

Student SRN number: 22051910

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

# Acknowledgement

As I near the completion of my postgraduate studies, I reflect with deep gratitude on the incredible learning journey I have experienced. I am truly thankful to all those who have supported and encouraged me along the way.

First and foremost, I extend my heartfelt thanks to Almighty God for His boundless blessings, for continually inspiring me, and for instilling in me the confidence and courage to persevere with faith and determination.

I am deeply grateful to Dr. Hyungrok Kim, my supervisor, for their invaluable guidance, steadfast support, and willingness to nurture my curiosity with patience and understanding throughout this project.

I would also like to express my sincere appreciation to all my professors at the University of Hertfordshire for imparting their knowledge and expertise, which have greatly enriched my academic journey and helped me grow both personally and professionally.

Last but not least, my deepest gratitude goes to my parents and friends for their unwavering encouragement, constant motivation, and steadfast belief in me without which this achievement would not have been possible.

# Abstract

Coronary artery disease (CAD) continues to be one of the major causes of deaths in the world and therefore, there is urgent need to have proper early prediction tool that can help in the better outcomes of the patient and overall healthcare expenditure. The conventional methods of diagnosis usually require costly imaging and invasive tests which results into a challenge of the wide spread screening. Machine learning offers promising solutions, but existing models frequently suffer from overfitting and poor generalization due to high-dimensional clinical data and inadequate feature selection. The research will overcome these limitations by building CAD prediction models through structured feature selection methodology. Using the 303 patient records which included 54 clinical features, three approaches were applied which are Recursive Feature Elimination, Forward Selection, and Backward Elimination. These approaches by decreasing dimensionality by 44%. SMOTENC dealt with the problem of class imbalance. All the approaches were tested with the help of the Random Forest, XGBoost, and Gradient Boosting algorithms. Random Forest with Backward Feature Elimination achieved optimal performance with 85% accuracy, 87% precision, 85% recall, and 86% F1-score. Results shows that feature selection methods significantly improve model generalization, providing a reliable, cost-effective tool for clinical CAD risk assessment.

Table of Contents

[DECLARATION STATEMENT ii](#_Toc206153369)

[Acknowledgement iii](#_Toc206153370)

[Abstract iv](#_Toc206153371)

[1 Introduction 1](#_Toc206153372)

[1.1 Background 1](#_Toc206153373)

[1.2 Problem Statement 1](#_Toc206153374)

[1.3 Aim 2](#_Toc206153375)

[1.4 Research Question 2](#_Toc206153376)

[1.5 Project Objectives 2](#_Toc206153377)

[1.6 Novelty of the Research 2](#_Toc206153378)

[1.7 Report Structure 3](#_Toc206153379)

[2 Literature Comparison 4](#_Toc206153380)

[2.1 Identification of Gaps 5](#_Toc206153381)

[3 Methodology 7](#_Toc206153382)

[3.1 Research Design 7](#_Toc206153383)

[3.2 Dataset Description 8](#_Toc206153384)

[3.3 Class Balancing with SMOTENC 8](#_Toc206153385)

[3.4 Feature Selection Techniques 9](#_Toc206153386)

[3.5 Model Selection and Training 9](#_Toc206153387)

[3.6 Hyperparameter Tuning 9](#_Toc206153388)

[3.7 Evaluation Metrics 10](#_Toc206153389)

[3.8 Ethical Considerations 10](#_Toc206153390)

[3.9 Project Management 10](#_Toc206153391)

[3.10 Summary 11](#_Toc206153392)

[4 Experiment 12](#_Toc206153393)

[4.1 Data Preprocessing 12](#_Toc206153394)

[4.1.1 Dataset Overview 12](#_Toc206153395)

[4.1.2 Initial Data Inspection and Cleaning 12](#_Toc206153396)

[4.1.3 Exploratory Data Analysis (EDA) 12](#_Toc206153397)

[4.1.4 Encoding and Correlation Analysis 15](#_Toc206153398)

[4.1.5 Final Dataset Preparation 16](#_Toc206153399)

[4.1.6 Data Partitioning and Balancing with SMOTENC 16](#_Toc206153400)

[4.2 Feature Selection Techniques 17](#_Toc206153401)

[4.3 Model Implementation and Hyperparameter Tuning 18](#_Toc206153402)

[5 Result Analysis and Conclusion 21](#_Toc206153403)

[5.1 Overall Result Analysis 21](#_Toc206153404)

[5.2 Training Performance 21](#_Toc206153405)

[5.3 Testing Performance 22](#_Toc206153406)

[5.4 Conclusion 23](#_Toc206153407)

[References 24](#_Toc206153408)

[Appendices 26](#_Toc206153409)

[Learning Curves 26](#_Toc206153410)

[Confusion Matrices 31](#_Toc206153411)

[Code 35](#_Toc206153412)

# Introduction

## Background

The world health organization (WHO) facts state that cardiovascular disease (CVD) is the leading cause of death across the globe, claiming close to 17.9 million lives every year. These conditions, which include heart failure, stroke, and coronary artery disease, typically progress silently and are not identified until they are severe or fatal (Parsan et al, 2025). It is important to identify those at risk early in order to intervene and prevent it on time. Statistical methods have traditionally been utilized to detect risks (e.g., the Framingham Risk Score), and these models incorporate a relatively small number of clinical variables and linear relationships (Shishehbori and Awan, 2024). Nevertheless, these models can be incapable of representing the nonlinear complicated interactions among risk factors that contemporary healthcare data demonstrate. Such a weakness introduces the possibility of using machine learning (ML) methods, which may be trained on large, heterogeneous datasets and identify crisp patterns that might otherwise be challenging to identify using traditional methods.

Current improvements and developments in data science and ML allowed achieving better predictions in healthcare-related matters, especially when applying ensemble learning models like RF, Gradient Boosting, and XGBoost. They are predictively accurate and robust since they rely on the outputs of many base learners. But the efficiency of any ML model strongly relies on the quality and relevancy of features that are given as input. The majority of medical datasets, such as the well-known UCI Heart Disease dataset, contain duplicate or unnecessary features that can either reduce model accuracy or increase computing costs. (Anderson, 2024). RFE, Backward Elimination and Forward Selection are feature selection models that can be utilized to overcome this problem and determine the most important predictors, and eliminate the less valuable variables. The prospect of this ensemble models combined with strategic feature selection approach is high in terms of accuracy and interpretability of CVD prediction.

## Problem Statement

Although the ensemble learning models have been proved to be capable of achieving high results, their performance may be compromised when dealing with high-dimensional and noisy data that is often represented in the medical diagnostics problems. Though the UCI Heart Disease dataset is quite popular, it has many relevant and irrelevant features that can make machine learning models confused, unless properly preprocessed. Moreover, systematic comparison of feature selection techniques specific to ensemble classifiers on the heart disease prediction problem has not been done. Most studies either ignore the feature selection process or apply one technique unjustifiably, resulting in models which might lack generalization ability or provide explainability to be used in the clinical context (Cheng, 2024). The research should fill this gap by comprehensively exploring the impact of various feature selection methods on the results of such ensemble classifiers as Random Forest, Gradient Boosting, and XGBoost. The aim is to determine the optimum feature selection approach that would enhance the predictive performance and interpretability of the methods in the diagnosis of the cardiovascular disease.

## Aim

The aim of the research is to develop a robust predictive model for cardiovascular disease by investigating the impact of different feature selection techniques on the performance of ensemble learning classifiers using the UCI Heart Disease dataset.

## Research Question

How do different feature selection methods affect the accuracy and performance of ensemble classifiers in predicting cardiovascular disease?

## Project Objectives

* To review and critically evaluate existing literature on predicting cardiovascular disease using machine learning, with a focus on ensemble methods and feature selection techniques.
* To preprocess and explore the UCI Heart Disease dataset, ensuring data quality through handling of missing values, encoding categorical features, and normalizing continuous variables.
* To implement and compare various feature selection techniques such as Recursive Feature Elimination (RFE), Forward Selection and Backward Elimination to find the most appropriate predictors of heart disease.
* To build and evaluate ensemble models with the classifiers Random Forest, Gradient Boosting, XGBoost using selected features to assess their prediction performance.
* To analyse and interpret the results by comparing model metrics such as accuracy, precision, recall, and F1-score across various feature selection methods and ensemble models.
* To draw conclusions on which feature selection and ensemble method combination yields the most exact and interpretable model for cardiovascular disease prediction.

## Novelty of the Research

This research presents a new framework for comparison that assesses how different feature selection techniques affect the performance of the most advanced ensemble models in the particular field of cardiovascular disease prediction. Unlike many previous studies that either ignore feature selection or rely on default features, this study systematically investigates how selecting the most relevant features improves both model performance and interpretability. The integration of thorough data preprocessing, feature engineering, and evaluation metrics ensures that the results are practically relevant for deployment in clinical decision-support systems.

## Report Structure

There are several chapters in the report. Chapter 1 introduces the research background, aims, objectives, and significance. Chapter 2 presents a literature review and critical analysis of existing work in CVD prediction using ML. Chapter 3 explains the methodology, including data collection, preprocessing, feature selection, and model implementation. Chapter 4 presents the results and performance analysis. Chapter 5 provides a discussion of the results, and summaries the report. The main sections are followed by references and appendices with the code and additional plots.

# Literature Comparison

Determination of CVD is a vital medical diagnostic area that has already been largely explored. Although medicine has seen extensive use of ML and ensemble methods, only a few of them involve feature selection to augment the predictive performance. The main aim of this research is to create a reliable CVD detection model by use of ensemble classifiers along with feature selection techniques to the UCI Heart Disease dataset. This objective is compared to prior studies below.

Most importantly, Singh et al, (2022) investigated the significance of feature selection to boost the classification accuracy and reduce the computational time in predicting CVD. ML algorithms, Random Forest, Naïve Bayes, SVM along with filter-based feature selection (Pearson Correlation and Chi-Square) had up to 84% accuracy with Logistic Regression. The authors did not explore wrapper or embedded methods, however. This research extends this using wrapper techniques such as Recursive Feature Elimination (RFE), as well as stepwise methods (Forward and Backward Selection) to enable further comparison of the relative effects of feature selection on ensemble models.

The work of Tiwari et al, (2022) is a stacked ensemble framework merged datasets and using classifiers like extra trees, RF and XGBoost, results in 92.34% accuracy. Although their result demonstrates the efficacy of stacked ensembles, feature selection is not considered for their performance. In doing so, this research analyzes which features impact base ensemble models individually, in turn resulting in both improved model performance and interpretability.

In this work, Alqahtani et al, (2022) utilized a large dataset (70K records) and developed a hybrid ML and deep learning model ensemble which present 88.7% accuracy using random forest based feature extraction. Instead of implementing systematic feature selection, the authors were interested in scale and model complexity. This is in contrast to this approach that focuses on comparing the efficacy of feature selection methods prior to designing ensemble classifiers to drive more efficiency and standardize for a smaller dataset.

Using the Chi Squared, Mutual Information and Imbalance features, 3458 unique features were selected and Relief and LASSO feature selection were applied to these 3458 features which were used to select 10 features. Random Forest with Relief and LASSO resulted in high accuracy (99.05%) (Ghosh et al, 2021). The goal is to perform feature selection and ensemble learning together and their hybrid method fits this specification. However, there is a risk that reliance on a small set of techniques will affect generalizability. RFE and stepwise selection are used in this research to select the most robust approach across multiple classifiers.

With that in mind, Noroozi et al, (2023) examined 16 feature selection methods (filter, wrapper and evolutionary) on their effectiveness with seven ML models. Filter methods, for instance, Correlation Based Feature Selection (CFS) and information gain were used, that worked well, while wrappers improved sensitivity. By performing an extensive comparison, the inclusion of several feature selection methods which combine well with ensemble classifiers like XGBoost, Gradient Boosting and Random Forest.

In their study, Ganie et al, (2023) concentrated on boosting algorithms (Gradient Boost, XGBoost and AdaBoost) the best results, 92.2% accuracy, belong to Gradient Boost. Additionally, the authors used data preprocessing (imputation, outlier removal) without in depth feature selection, however. This research extends this idea and combines formal feature selection processes with ensemble models to form a framework that is both more interpretable and efficient as a predictor.

A deep stacking model with CNN-LSTM and CNN-GRU, improved through RFE is proposed by Almulihi et al, (2022) with 97.17% accuracy. They have high accuracy, but the deep learning model is not transparent nor clinically interpretable. However, the research modifies RFE to traditional ensemble classifiers, taking interpretability into account consistent with the practice of healthcare decision making.

In this work, Asif et al, (2023) achieved 98.15 percent accuracy with Extra Trees and performed hyperparameter tuning on the grid search. While their approach tuned model parameters well they did not consider feature selection. In so doing, this research evaluates if different selection strategies affect the outcomes of the respective ensemble model, doing a more holistic performance analysis.

Finally, it is concluded that there is prior research that demonstrates the effects both of ensemble learning and feature selection on CVD prediction. Most of the studies, however, consider only one of these aspects or, in case of considering all together, overlook one aspect, either model complexity or a narrow set of feature selection methods. There is a gap in current method comparison studies that this research fills by systematically comparing and contrasting RFE, Forward Selection and Backward Elimination with ensemble classifiers (Random Forest, Gradient Boosting, XGBoost). The goal of this research is to find a structured approach to this modeling problem that helps strike a balance between predictive accuracy and model interpretability and therefore, for their potential practical deployment in clinical settings.

## Identification of Gaps

Extensive research exists in cardiovascular disease detection based on ML and ensemble techniques, however, there are still some gaps. There are most existing studies focusing on model performance, but these neglect the systematic comparisons of feature selection methods and their impact on ensemble classifiers. The combination of wrapper-based methods like RFE or stepwise selection techniques, with ensemble methods is rarely studied. However, there is also little focus on combining predictions that are accurate with predictions that are interpretable, needed in clinical settings. Work in some fields, too, really depend on large merged datasets as well as reducing reproducibility on standardized datasets like UCI Heart Disease. Finally, this research fills these gaps by assessing the effect of different feature selection techniques on ensemble classifiers to determine the highest quality combination for CVD prediction that is accurate, efficient and interpretable.

# Methodology

The chapter narrates the procedure of creating a model in order to predicting the coronary artery disease (CAD) occurrence using the ensemble learning and feature selection approaches. It involves data preparation, class balancing, feature selection method, model training, practice of tuning hyperparameters and evaluation of its accuracy. All the techniques were applied and implemented effectively on the Z-Alizadeh Sani dataset which is compatible with the execution of the project.

## Research Design

The research is systematic and aimed at realizing an accurate model of prediction of cardiovascular disease. It begins with downloading the UCI Heart Disease dataset and subsequently carrying out an intensive process of preprocessing the dataset to ensure the quality of data. This includes normalizing continuous variables, encoding categorical features, and imputation of missing values. Then, three feature selection methods, i.e., RFE, Backward Elimination, and Forward Selection are utilized separately to determine the most informative predictors. Ensemble learning models, like RF, Gradient Boosting, and XGBoost are trained and evaluated using the chosen features of each approach. This training process involves cross-validation hyperparameter tuning that aims at optimization of performance. The performance measures utilized to evaluate the methods include accuracy, precision, recall, and F1-score. The metrics are useful in finding the best feature selection and model combination. The last step will be the comparison of results between models and interpretation of the features that significantly impact the prediction of the disease to assure statistical and clinical significance. Such systematic design will provide consistency in the methodology and meaningfulness.

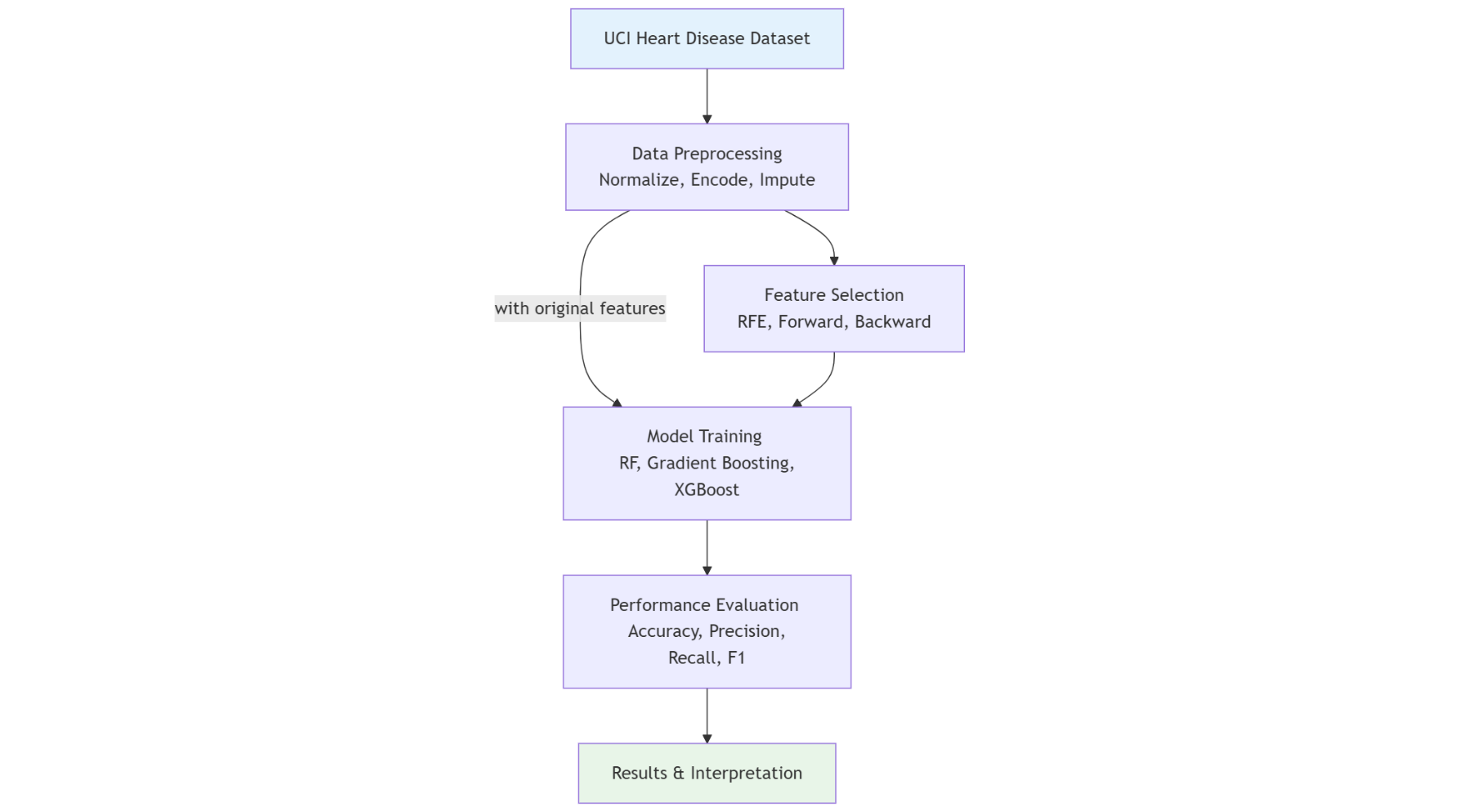


Figure 1 Project Architecture

## Dataset Description

The data employed in the research is the UCI Heart Disease dataset that is openly available and frequently utilized in academic activities. It is a compilation of heart disease data with various other sources such as Cleveland, Hungary, Switzerland, and Long Beach VA. The dataset of Cleveland is the most utilized because the records are relatively complete. It contains 303 patient cases and 14 variables, including goal (heart disease presence or absence), age, sex, kind of chest discomfort, resting blood pressure, and cholesterol. This is a dataset that was gathered to be utilized in the assessment of coronary artery disease and it was initially provided by the Cleveland Clinic Foundation.

This dataset could be not used as it is to train machine learning models but required proper preparation. Preprocessing was done as a first step, and it was an act of cleaning the data and dropping the features that were not informative. Categorical data were then converted to numerical format so that they could be used in the modeling algorithms.

## Class Balancing with SMOTENC

There was class imbalance in the original dataset, which may cause a skewed prediction of the models based on the majority class. To overcome this, SMOTENC algorithm was used following the data split of 80-20 train-test split. SMOTENC is more appropriate to be used on datasets with mixed data types. Features that had four or fewer unique values were considered categorical features and their indexes were provided to the algorithm to be represented meaningfully in synthetic samples. This step produced balanced training data of 340 samples, consisting of 170 samples per class.

## ****Feature Selection Techniques****

Feature selection helped to improve model accuracy, reduce overfitting, and speed up training. In this research, three techniques including RFE, Forward Selection, and Backward Elimination were applied to select the top 30 features from the dataset.

RFE was performed using a RF classifier as the base estimator. Starting with all features, the model ranked them by importance. The least important feature was removed in each iteration until only 30 remained. The resulting subset (X\_ rfe\_ chosen) was used for classification.

Forward Selection was implemented using the SequentialFeatureSelector from the **mlxtend** library. B eginning with no features, it added one at a time, selecting the feature that most improved accuracy based on a Random Forest classifier. The process continued until 30 features (X\_forward) were selected.

Backward Elimination also used SequentialFeatureSelector but started with all features. It removed one feature per iteration the one whose exclusion caused the smallest performance drop until 30 features (X\_backward) remained. This method preserved interactions between features critical to CAD prediction.

## ****Model Selection and Training****

Three ensemble models including RF, XGBoost, and GB were chosen for classification due to their ability to model complex, high-dimensional, and imbalanced datasets effectively.

RF is a bagging method that builds multiple DTs and employs majority voting for predictions (Mosavi et al. 2021). It is robust and provides feature importance scores. In this research, RF was used both as a classifier and for feature selection.

XGBoost is a regularized gradient boosting algorithm that constructs trees sequentially, correcting previous errors. It supports parallel processing and prevents overfitting. It was trained on the balanced dataset and tested using each selected feature subset.

Gradient Boosting, while similar to XGBoost, lacks some optimization features. It trains models on a gradient descent in stages and was used as a baseline in terms of the performance. The training and testing of all the models were done in the same way with the same data that was balanced and the set of features that had been chosen.

## Hyperparameter Tuning

RandomizedSearchCV is used to optimize the hyperparameters of each classifier. Here, it takes random samples of combinations of parameter-values within given ranges and assesses these through cross-validation. Each classifier had its own set of tunable parameters. For example, the Random Forest parameters were number of trees (n\_estimators), maximum tree depth (max\_depth), and minimum samples per leaf (min\_samples\_leaf).In XGBoost and Gradient Boosting, parameters including learning rate (learning\_rate) and the depth of the tree as well as the ratio of subsample were some of the important parameters. Based on this process, the best combination of parameters was obtained and the model was retrained on the balanced dataset and subsequently evaluated.

## Evaluation Metrics

Model performance was measured utilizing standard categorization metrics including accuracy, precision, recall, and F1-score. They were calculated from confusion matrix, which contains truth positives (TP), truth negatives (TN), false positives (TP) and false negatives (FN). The formulas are:

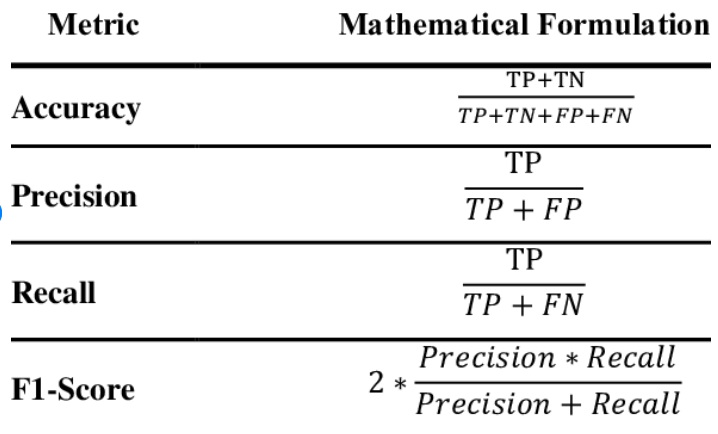


Figure 2 Evaluation Metrics (Azam et al, 2024)

The training and the test performance were measured. Confusion matrices were drawn to confirm how well each model differentiates between CAD and the non-CAD cases based on different combinations of modelled features.

## Ethical Considerations

This research ensured that ethical considerations were observed when dealing with medical information. The data used is publicly accessed and absolutely anonymized and does not include any personally identifiable information (PII), which makes the data privacy principles complied with. There was a secure processing and storing of data throughout the process of the project to ensure it is not misused or accessed by the wrong people.

## Project Management

Project management approach was implemented in a structured and iterative manner to support efficiently and timely completion of the research. The methodology was segmented into successive stages, including the acquisition of the dataset, its preprocessing, the selection of features, training of the model, hyperparameter tuning, and evaluation. Milestones and deliverables were defined in each phase, with the version-controlled scripts that could be used to reproduce them. Google Colab and GitHub are tools that have been used to handle code and documentation. Backups were carried out regularly and a systematic testing activity allowed to keep track of the progress and correct problems at their initial stages. This well-structured and modularized method allowed management of time, and end result being a fit and reliable system that was also scalable.

## Summary

This section discussed data preparation, class balancing, feature selection, and ensemble learning in order to come up with a CAD predication system that would be reliable. The models have been fine-tuned to give the best results and will form a good basis of the results and experimental analysis in the following chapter.

# Experiment

This chapter outlines experimental design and machine learning approaches on predicting coronary artery disease based on ensemble methodology, feature selection and class balancing on clinical data of patients.

## Data Preprocessing

### Dataset Overview

The experiment was conducted using the Z-Alizadeh Sani dataset, which includes detailed clinical, demographic, and laboratory data of 303 patients, each with 56 features. The target variable, *Cath*, indicates whether a patient is diagnosed with coronary artery disease (CAD) or is classified as normal. Specifically, 216 patients were labeled as having CAD, while 87 were identified as normal, highlighting a class imbalance that could potentially influence model outcomes.

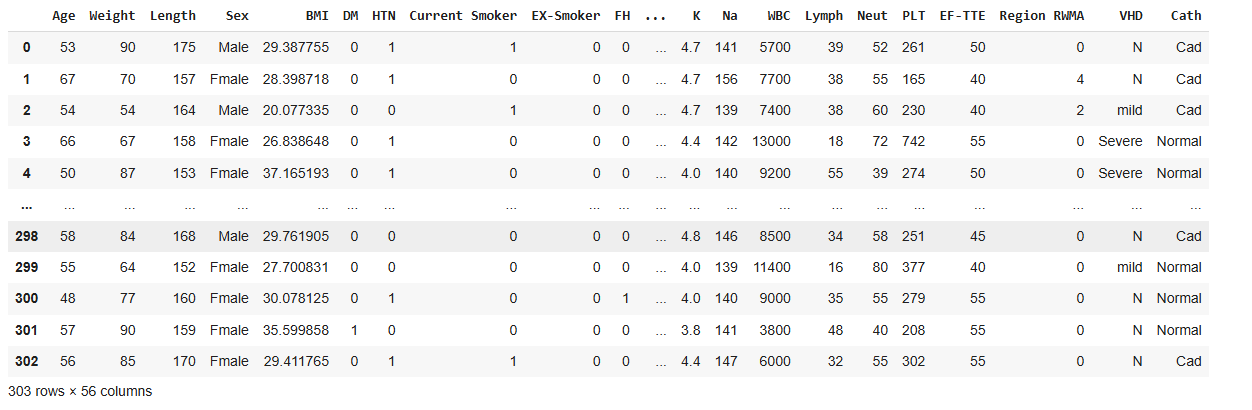


Figure 3 Initial Dataset View

### Initial Data Inspection and Cleaning

The dataset was initially loaded and inspected using various pandas functions. It comprised of 5 float-type, 29 integer-type, and 22 object-type columns. Importantly, the dataset did not contain any missing or duplicate records, ensuring data completeness. However, the feature *Exertional CP* had only a single unique value ('N') across all records. Since it did not provide any meaningful variance or information for classification, this column was dropped from the dataset, reducing the total number of features to 55.

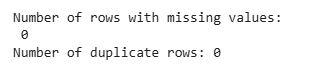


Figure 4 Number of Null and Duplicate Rows

### Exploratory Data Analysis (EDA)

To gain insights into the data distribution and feature relevance, several visualizations were created using Seaborn and Matplotlib.

Table 1 EDA of Various Attributes

|  |  |
| --- | --- |
| https://lh7-rt.googleusercontent.com/docsz/AD_4nXecLM81qv-dsNcvOtsurgf8CfCOIyz8QMl_PmO5tvojw2ITQoQGjKa7-fBDMevuxLlj2N1Dl2kkd6wmprU6jXn31GV9Rz4WeMs3G4X9CMqxpVRFIRApV4zc2YoDXc52i0nKDir5s_6mb6s7jAAb2vw?key=iTF2ZH4E9d1_DMThiypPrw | The distribution of the target variable revealed a higher prevalence of CAD in the dataset, with significantly more patients diagnosed with CAD than normal. |
| https://lh7-rt.googleusercontent.com/docsz/AD_4nXcI2FTbZKSa-FX1yyLZZGJvTH9xGwnlSjm18KH3YOql0SSqcDTNJaM6fTpJ5okSufV3OEZMqc_9N0qG9yaolwZwFx5MYA56lVxUL7PkoBs7FVlN16SYa9ZB-38hK-fIUGdX_NKzuOtbZgBEPQrUBg?key=iTF2ZH4E9d1_DMThiypPrw | A histogram of the *Age* feature showed that most patients fell within the 45 to 70-year age range, with a peak between 50 and 55 years. |
| https://lh7-rt.googleusercontent.com/docsz/AD_4nXe5-jskJ0qT6lzr6rfafKP9uSKQCebc8-jPd6XJpWEuUkh8MIrLtCewpXV2oT7yfziFzioLUc6w79-5z7Shd_Z-wkTm7_axULwY-feun0LU-TK_4luVClTdeLoKUpYrKWBBKpsZ7YpqpKc5Lg0tTg?key=iTF2ZH4E9d1_DMThiypPrw | Similarly, *Weight* values ranged from 48 to 120 kg, with most patients clustered between 65 and 85 kg, forming a slightly right-skewed distribution. |
| https://lh7-rt.googleusercontent.com/docsz/AD_4nXeWx_NCHvSoETHOxT5KNrta1Z-Jvp3P4T8N1i1lZYOWHnFkYip2UjaZQSRLwPxQitnfsxkLp9zd30dn68L3rz5uZU7weFjB9PrdJW8xf9rbPRxjDMDMQl4uAMDUURq7MDmphhSZGdHnPU95SRqGDg?key=iTF2ZH4E9d1_DMThiypPrw | A count plot for the *Sex* feature indicated a greater number of male patients compared to females. |
| https://lh7-rt.googleusercontent.com/docsz/AD_4nXdL2mGvCqhRRGBYB_0kp-mbfjlPObr0YWTC5Gr6B84dOULlwWG5Rbmhe-TixaudAYJBhVDeb5BXY-9qWuQisThRCjMZ6snO3PVd556rPXeDS8gCIkX7G5mqunBbe_cVFVmdoTIDYGomaX_BTqzFIPU?key=iTF2ZH4E9d1_DMThiypPrw | Analysis of *Valvular Heart Disease (VHD)* revealed that most patients had mild or no valvular issues, while fewer showed moderate or severe conditions. |
| https://lh7-rt.googleusercontent.com/docsz/AD_4nXf0EpyljXkO3HF1daPncxGB8LodjaTUedeW-lFOTKvX5cD2IRORg2IEoNLdXp20qaoOpjm8ikORtOuxlsVZ2K_720iOD6YVlKCCv2aeW7VipajyoPVyzEkYYYIKvv-AgWDIL7V98YIzuFELtaBHpGY?key=iTF2ZH4E9d1_DMThiypPrw | Regarding *Bundle Branch Block (BBB)*, the majority of patients had normal conduction, with only a small number exhibiting either Right or Left Bundle Branch Block. |
| https://lh7-rt.googleusercontent.com/docsz/AD_4nXdP2whA1ZljATARH6we4I4Lx-BGgQOxQvkbBSId3wPGd_cVpv7aLapWvswdWlTIemSck3TlS7EBigx4JWq1d5g0KMG5sE0BprhvQecCqtfsxXNG87kQ_8gc0ejYKjd9Ht0ZJWyzu9ZlqemdkPb3gfM?key=iTF2ZH4E9d1_DMThiypPrw | The blood pressure histogram showed that most patients had BP values around 130 mmHg. |
| https://lh7-rt.googleusercontent.com/docsz/AD_4nXdvE74BpSp-lMbmr_08CQznVF2pwmZYL4QhLRRzqb6gJ_OHGpOGRLXe8Yi7h7xf4AC2boVMzHxb_slAGfqq13ahF7T0UojI-sor_RLKH8mC-2l5dcTwxMoJOIj0yG8D66bKokl6th0OdsoIWl94Q0U?key=iTF2ZH4E9d1_DMThiypPrw | Additionally, more patients reported experiencing *Typical Chest Pain*, which could be a significant indicator of CAD presence. |

Following EDA, the next step involved transforming the dataset into a fully numerical format and analyzing inter-feature relationships to prepare it for machine learning models.

### Encoding and Correlation Analysis

To prepare the dataset for ML algorithms, all categorical features were encoded into numerical values using the LabelEncoder technique. This conversion made the dataset fully numeric, consisting of 5 float and 50 integer columns. A correlation heatmap was then generated to explore relationships between features. Strong positive correlations were observed between features such as *Weight* and *BMI* (0.73), *Length* and *Sex* (0.70), *Obesity* and *BMI* (0.71), *FBS* and *DM* (0.68), and *BP* and *HTN* (0.57). On the other hand, *Lymph* and *Neutrophils* showed a strong negative correlation of -0.92. Moreover, *Typical Chest Pain* had a notable negative correlation with *Atypical* (-0.72).

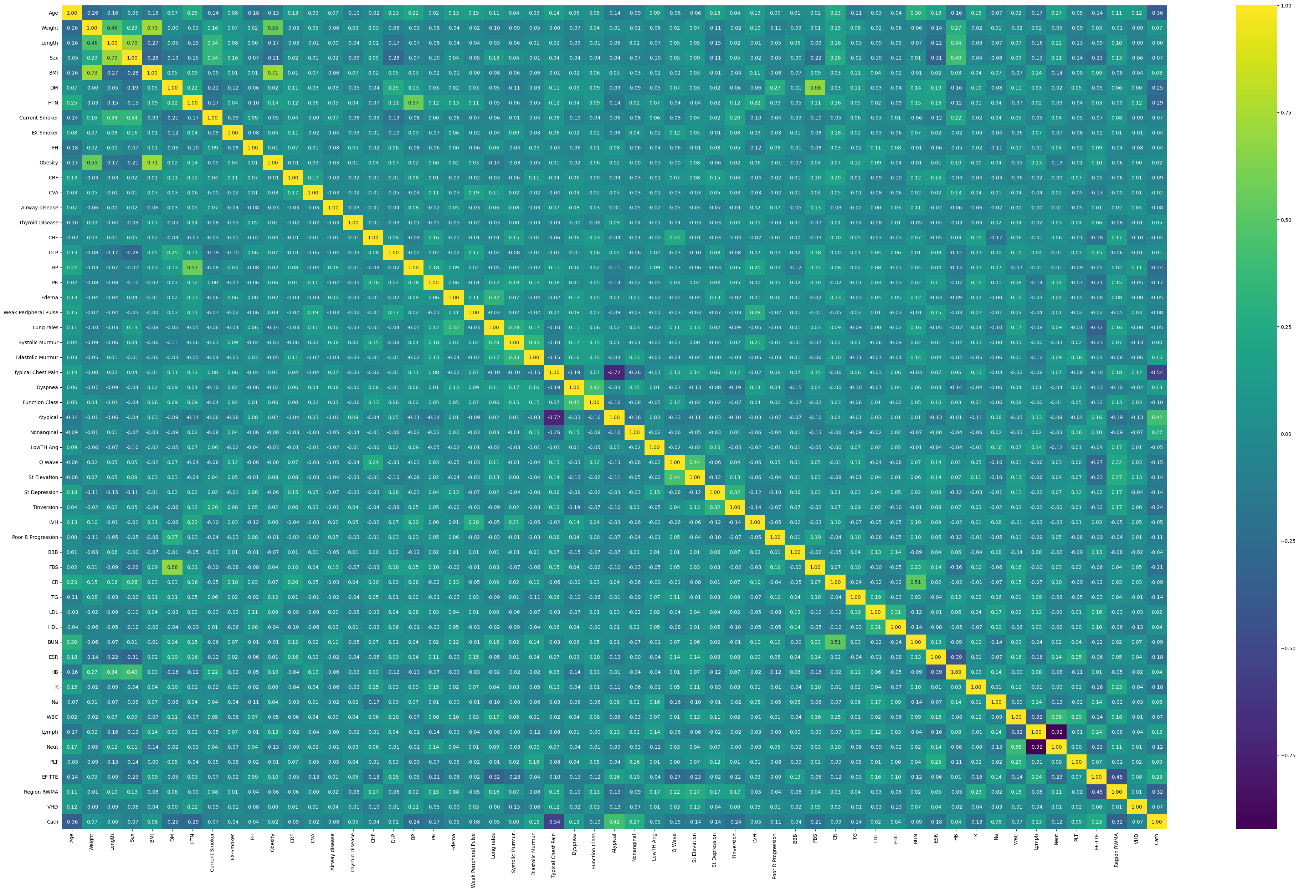


Figure 5 Encoding and Correlation Analysis

In relation to the target variable *Cath*, *Typical Chest Pain* demonstrated the strongest negative correlation (-0.52), while *Atypical Chest Pain* showed a positive correlation of 0.42. These findings suggest that the presence or absence of specific symptoms may significantly influence CAD diagnosis. After encoding and correlation analysis, the dataset was finalized for model development, ensuring all necessary preprocessing steps were complete.

### Final Dataset Preparation

After performing cleaning, transformation, and encoding, the final dataset comprised 55 features and 303 records, fully prepared for ML and DL model development. The processed dataset was saved as final\_data.csv for use in subsequent stages of the research.

### Data Partitioning and Balancing with SMOTENC

The experimental setup commenced with data partitioning and addressing class imbalance to ensure a fair and reproducible evaluation. This phase utilized the cleaned and fully preprocessed dataset. An 80–20 train-test split was applied, with random\_state=1 set to maintain consistency across runs. The target variable, **‘Cath’**, continued to represent the diagnosis outcome for coronary artery disease (CAD).

Table 2 Dataset Configuration

|  |  |
| --- | --- |
| Parameter | Value |
| Total Records | 303 |
| Original Features | 54 |
| Training Records | 242 |
| Test Records | 61 |
| Selected Features | 30 |

The training set exhibited class imbalance i.e., **170 normal cases** and **72 CAD cases**. To address this, **SMOTENC** was applied to preserving categorical features while balancing the target distribution.

Table 3 Class Distribution

|  |  |  |
| --- | --- | --- |
| Condition | Original | After SMOTENC |
| No CAD (0) | 170 | 170 |
| CAD Present (1) | 72 | 170 |
| Total Training | 242 | 340 |

## Feature Selection Techniques

To develop model performance and reduce complexity, feature selection methods were applied to identify the most relevant predictors. Specifically, three distinct methods were employed: RFE, Forward Selection, and Backward Elimination. Each technique aimed to reduce dimensionality by retaining 30 out of the original 54 features, resulting in approximately 44% reduction. This process helped eliminate irrelevant or redundant features, thereby minimizing the risk of overfitting and improving both the training efficiency and interpretability of the models.

Table 4 Feature Selection Methods Comparison

|  |  |  |
| --- | --- | --- |
| Method | Approach | Key Features Selected |
| **RFE** | Recursive elimination by importance | Demographics (Age, Weight, Length, BMI), Comorbidities (DM, HTN), Vitals (BP, PR), Symptoms (Chest Pain types, Dyspnea), ECG (Tinversion), Extensive Labs (Lipids, CBC, Chemistry), Cardiac function (EF-TTE, RWMA, VHD) |
| **Forward Selection** | Greedy addition of best features | Demographics + Sex, Risk factors (Smoking, FH, CRF), Conditions (Airway disease, CHF, DLP), Signs (Edema, Weak pulse), Chest pain variants, ECG abnormalities (Q wave, ST changes, LVH), Selected labs (FBS, HDL, WBC) |
| **Backward Elimination** | Remove least informative features | Extended demographics, Smoking history (Current + Ex), Comorbidities (Obesity, CVA, Thyroid), Examination findings (Lung rales, Murmur), Comprehensive ECG (ST elevation/depression, T-inversion), Focused labs (CR, ESR, HB, K) |

After feature selection, models were trained using the balanced dataset.  The systematic approach maintained consistent train-test splits and SMOTENC balancing across all conditions.

Table 5 Dataset Dimensions Before and After Feature Selection Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Selection Method | Balanced Training | Test Records | Dimensionality Reduction |
| RFE | 340 (170 each class) | 61 | 44% (54→30) |
| Forward Selection | 340 (170 each class) | 61 | 44% (54→30) |
| Backward Elimination | 340 (170 each class) | 61 | 44% (54→30) |

All experimental conditions were conducted under identical processing parameters, ensuring a fair comparison while allowing for the identification of distinct predictor combinations. These complementary feature sets captured various clinical and pathological aspects of coronary artery disease (CAD), highlighting the strengths of each selection strategy. This organized approach became a good basis to further develop the model and test its performance.

## Model Implementation and Hyperparameter Tuning

Having specified the features and balanced dataset, machine learning models were trained and tested to determine how the variety of feature subsets influence the predictive performance. Three of the most widely used classifiers based on ensembles were applied to develop CAD prediction models, that is, RF, XGB, and GB. The training data were SMOTENC-balanced and all models were trained on the SMOTENC-balanced training data and tested on a fixed test set so that the results of an individual experiment were consistent.

GridSearchCV was used to find the best Hyperparameter with 2-fold cross-validation. This technique helped to find a good trade-off between the accuracy and generalization without taking a huge chance of overfitting. In general, the experimental structure allowed determining which ensemble model appeared to be the best to predict CAD reliably.

Table 6 Hyperparameter Tuning Summary

|  |  |
| --- | --- |
| **Model** | **Tuned Hyperparameters** |
| **Random Forest (RF)** | A close-up of a text  AI-generated content may be incorrect. |
| **XGBoost (XGB)** |  |
| **Gradient Boosting (GB)** | A close-up of words  AI-generated content may be incorrect. |

These parameters were chosen so as to regulate their model complexity, regularization, and learning behavior so as to balance the accuracy, precision, recall, and the F1-score. A confusion matrix was used to assess classification performance, highlighting true and false predictions for both positive and negative coronary artery disease cases. For example, confusion matrices of RF with all original variables for Training and testing sets are shown in below table. Other confusions matrices are attached in Appendix (Confusion matrices).

|  |  |
| --- | --- |
| **Training** | **Testing** |
|  |  |

Figure 6 Confusion Matrix - RF with Original Variables

Learning curves were also analyzed to track training and validation performance over time, revealing potential underfitting or overfitting. Learning curves of RF with all original features are shown below and others in Appendix (Learning Curves).

Figure 7 Learning Curve - RF with all Features

Performance analysis on training and test sets provided information of generalization capacity of any given model. Such a comprehensive study allowed determining the most effective model when predicting coronary artery disease, facilitating the correct and reliable diagnosis of a cardiovascular disease. To identify the most reliable method of model prediction, the next section provides details on the complete analysis of the entire results over various feature selection methods to identify the best technique to use.

A graph of a graph of a graph

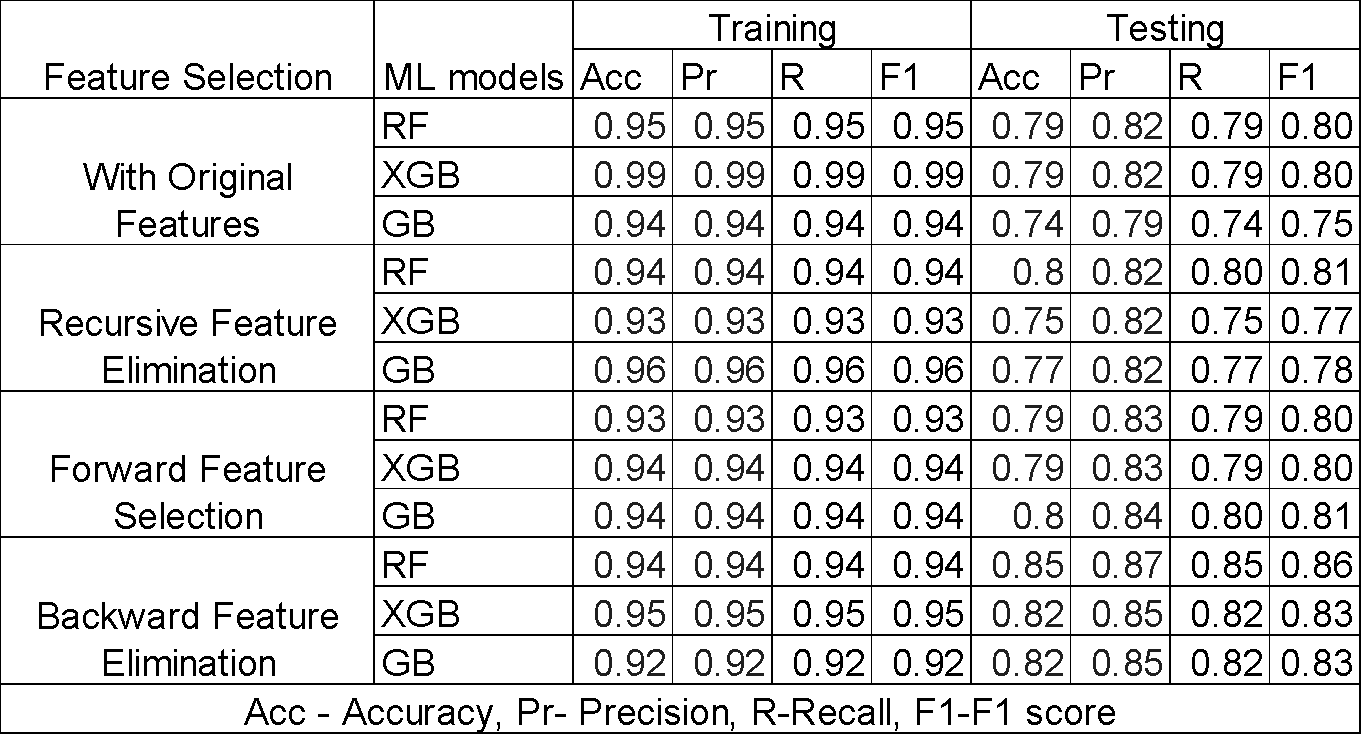
AI-generated content may be incorrect.

# Result Analysis and Conclusion

## Overall Result Analysis

This section evaluates the effectiveness of the ensemble models on different feature selection procedures. And, their training and testing results were used to determine the best feature selection approach to predict CAD. The overall results with the assessed accuracy (Acc), precision (Pr), recall (R) and F1-scores (F1) of all combinations between model and feature are presented in below Table.

Table 7 Overall Results



## Training Performance

The XGBoost (XGB) consistently reached greatest training accuracy, precision, recall and F1-scores of around 0.93 to 0.99 showing that it fitted the training data very well. Gradient Boosting (GB) and Random Forest (RF) also performed well in terms of training with the majority of the metrics exceeding 0.90.



Figure 8 Training Results

Differences in training scores across feature selection methods were quite low, shows that reducing features did not significantly affect model fitting.

## Testing Performance

Testing accuracy ranged from 0.74 to 0.85, showing varied generalization on unseen data. Random Forest combined with Backward Feature Elimination delivered the best testing results, achieving accuracy of 0.85, precision 0.87, recall 0.85, and F1-score 0.86. XGBoost and Gradient Boosting also performed well with backward elimination, with testing accuracies near 0.82 and solid precision and recall between 0.82 and 0.85.



Figure 9 Testing Results

Models using feature selection outperformed those trained on the full original feature set, indicating that removing irrelevant or noisy features improved generalization. Forward Selection and Recursive Feature Elimination yielded moderate testing performance (accuracy and F1 mostly between 0.79 and 0.81), where RF and GB were consistent. Models trained on all original features showed the lowest testing accuracy and F1-scores, likely due to overfitting or noise.

Overall, Feature selection enhanced model generalization, with Backward Feature Elimination producing the best test outcomes. Random Forest with backward elimination was the top performer across all testing metrics. Although XGBoost had the best training fit, its test results were slightly below RF’s. Gradient Boosting showed robust but generally lower test accuracy compared to the other two models.

## Conclusion

This research successfully developed ML models for coronary artery disease prediction, establishing the critical role of feature selection in medical AI applications. Through systematic evaluation of three feature selection methodologies and three algorithms, Random Forest with Backward Feature Elimination emerged as the optimal configuration, balancing accuracy and generalization for CAD prediction. Key findings demonstrate that feature selection substantially enhances performance on unseen data, with the optimal model achieving 85% testing accuracy. The 44% dimensionality reduction eliminated noise while preserving clinically meaningful predictors, improving computational efficiency and diagnostic reliability. SMOTENC successfully addressed class imbalance challenges inherent in medical datasets. This research contributes empirical evidence supporting feature selection superiority and provides a replicable methodological framework for medical machine learning. The findings offer healthcare practitioners a reliable, interpretable CAD risk assessment tool while reducing data collection burden. Future directions include prospective clinical validation, and multimodal data integration, to enhance clinical adoption and advance automated cardiac risk assessment.

# References

Alizadehsani, R., Roshanzamir, M., and Sani, Z. (2013). Z-Alizadeh Sani [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5Q31T>.

Almulihi, A., Saleh, H., Hussien, A.M., Mostafa, S., El-Sappagh, S., Alnowaiser, K., Ali, A.A. and Refaat Hassan, M., (2022). Ensemble learning based on hybrid deep learning model for heart disease early prediction. *Diagnostics*, *12*(12), p.3215.

Alqahtani, A., Alsubai, S., Sha, M., Vilcekova, L. and Javed, T., (2022). Cardiovascular disease detection using ensemble learning. *Computational Intelligence and Neuroscience*, *2022*(1), p.5267498.

Anderson, K., (2024), ‘The role of data preprocessing in machine learning accuracy for heart disease prediction: Hybrid models for heart disease prediction – combining neural networks with traditional’, Preprint, [Online]. (Available at: <https://www.researchgate.net/publication/386177388_The_Role_of_Data_Preprocessing_in_Machine_Learning_Accuracy_for_Heart_Disease_Prediction_Hybrid_Models_for_Heart_Disease_Prediction_Combining_Neural_Networks_with_Traditional>) [Accessed 15th June 2025].

Asif, D., Bibi, M., Arif, M.S. and Mukheimer, A., (2023). Enhancing heart disease prediction through ensemble learning techniques with hyperparameter optimization. *Algorithms*, *16*(6), p.308.

Azam, B., Kuttichira, D., Verma, B., Rahman, A. and Wang, L., 2024. Explainable Image Recognition With Graph-Based Feature Extraction. IEEE Access.

Cheng, X., 2024. A Comprehensive Study of Feature Selection Techniques in Machine Learning Models. *Insights in Computer, Signals and Systems*, *1*(1), pp.10-70088.

Ganie, S.M., Pramanik, P.K.D., Malik, M.B., Nayyar, A. and Kwak, K.S., (2023). An Improved Ensemble Learning Approach for Heart Disease Prediction Using Boosting Algorithms. *Comput. Syst. Sci. Eng.*, *46*(3), pp.3993-4006.

Ghosh, P., Azam, S., Jonkman, M., Karim, A., Shamrat, F.J.M., Ignatious, E., Shultana, S., Beeravolu, A.R. and De Boer, F., (2021). Efficient prediction of cardiovascular disease using machine learning algorithms with relief and LASSO feature selection techniques. *IEEE Access*, *9*, pp.19304-19326.

Mosavi, A., Sajedi Hosseini, F., Choubin, B., Goodarzi, M., Dineva, A.A. and Rafiei Sardooi, E., 2021. Ensemble boosting and bagging based machine learning models for groundwater potential prediction. *Water Resources Management*, *35*(1), pp.23-37.

Noroozi, Z., Orooji, A. and Erfannia, L., (2023). Analyzing the impact of feature selection methods on machine learning algorithms for heart disease prediction. *Scientific reports*, *13*(1), p.22588.

Parsan, S.J., Kim, M., Martinez Singh, M., Chahine, A. and Jafar, S., 2025. Advocating for Early Cardiovascular Disease Screening in Canada Authors. *Available at SSRN 5235658*.

Shishehbori, F. and Awan, Z., 2024. Enhancing cardiovascular disease risk prediction with machine learning models. *arXiv preprint arXiv:2401.17328*.

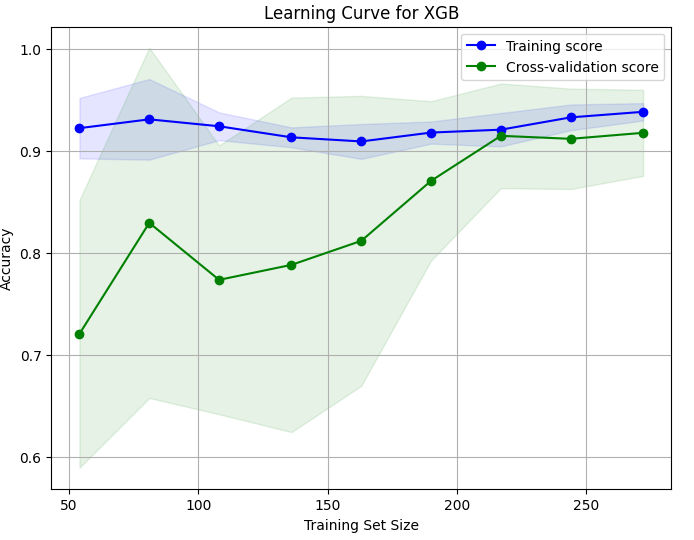
Singh, P., Pal, G.K. and Gangwar, S., (2022). Prediction of cardiovascular disease using feature selection techniques. *International Journal of Computer Theory and Engineering*, *14*(3), pp.97-103.

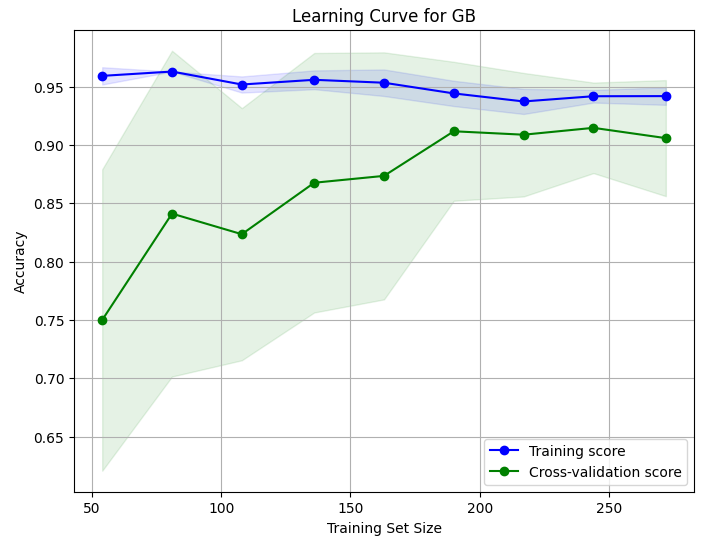
Tiwari, A., Chugh, A. and Sharma, A., (2022). Ensemble framework for cardiovascular disease prediction. *Computers in Biology and Medicine*, *146*, p.105624.

# Appendices

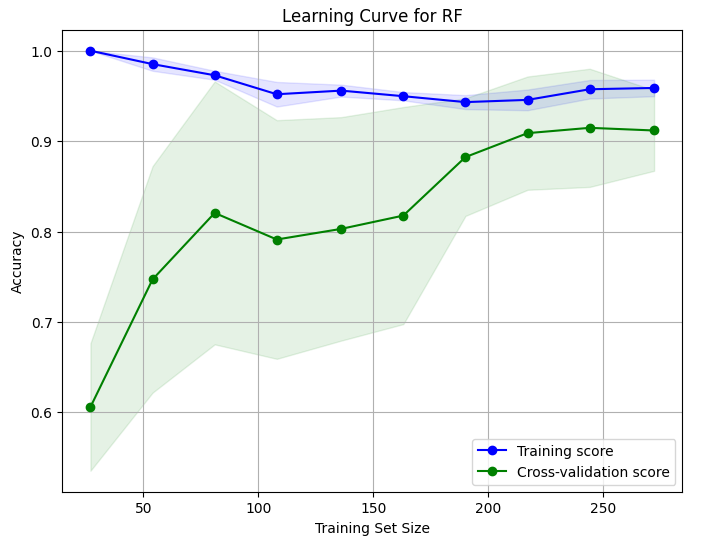
## Learning Curves

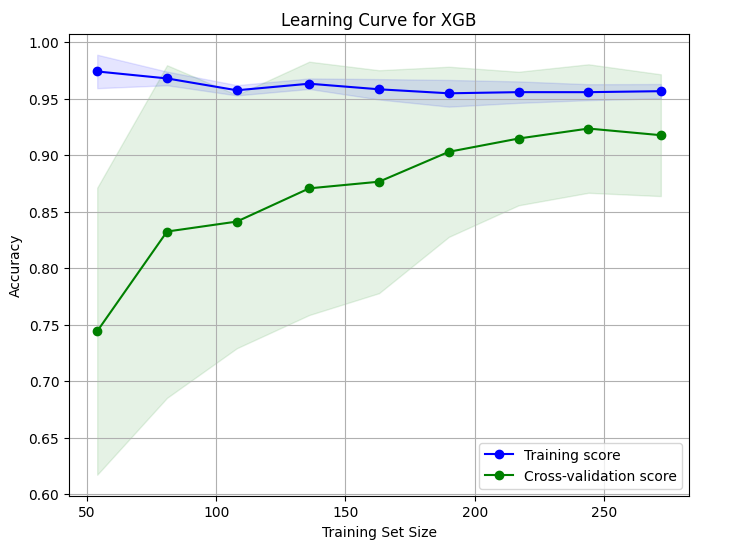
1. **With All Features**

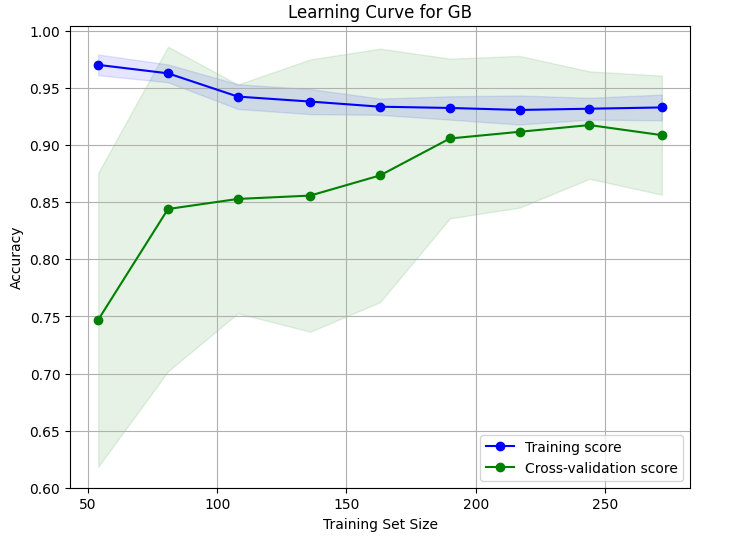
****



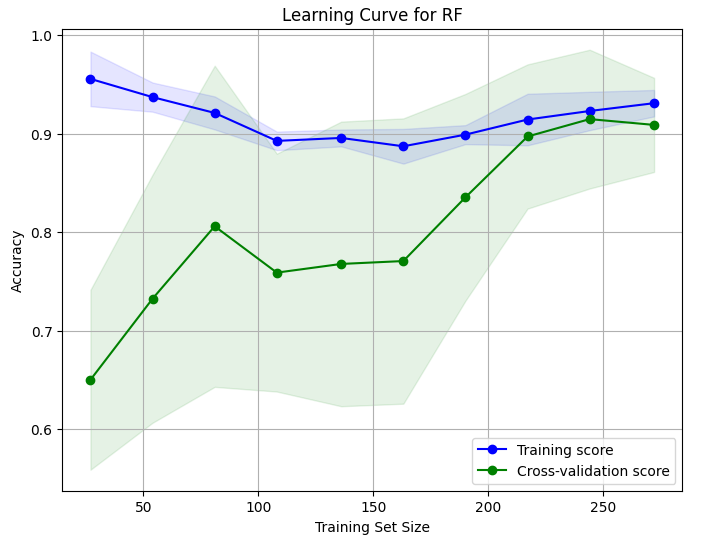
1. **With RFE Feature Selection**

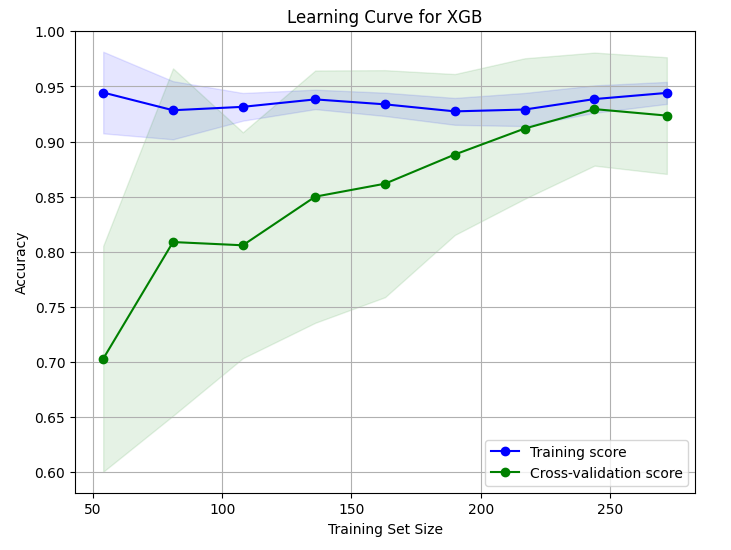
****

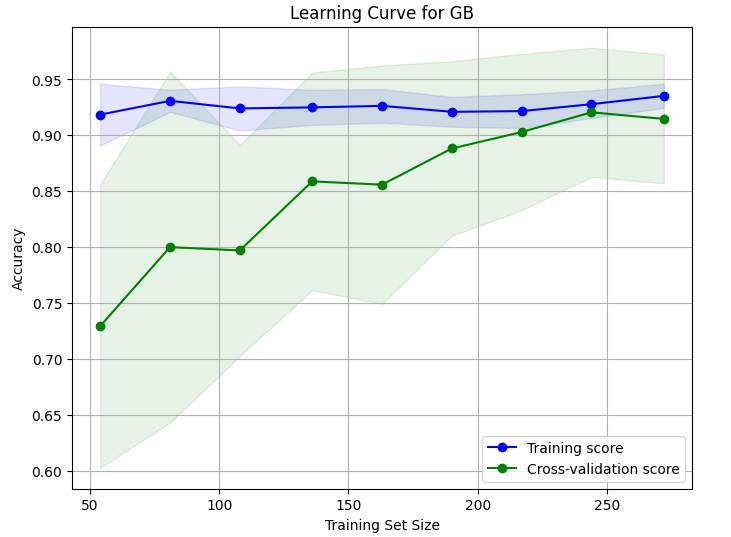
****

****

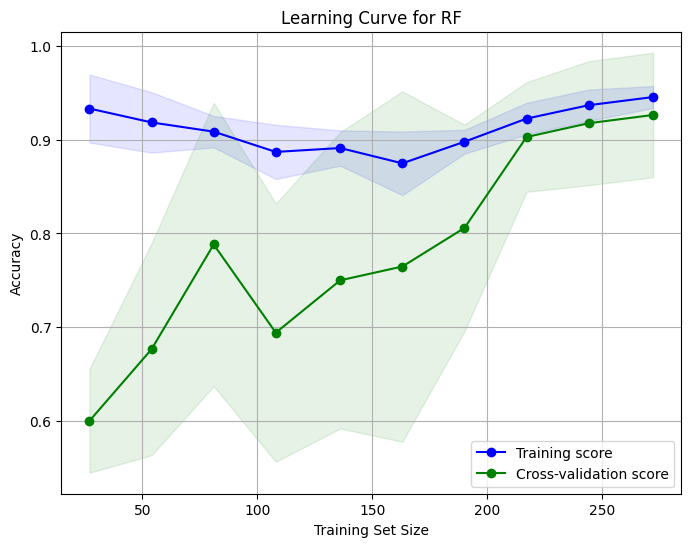
1. **With Forward Selection**

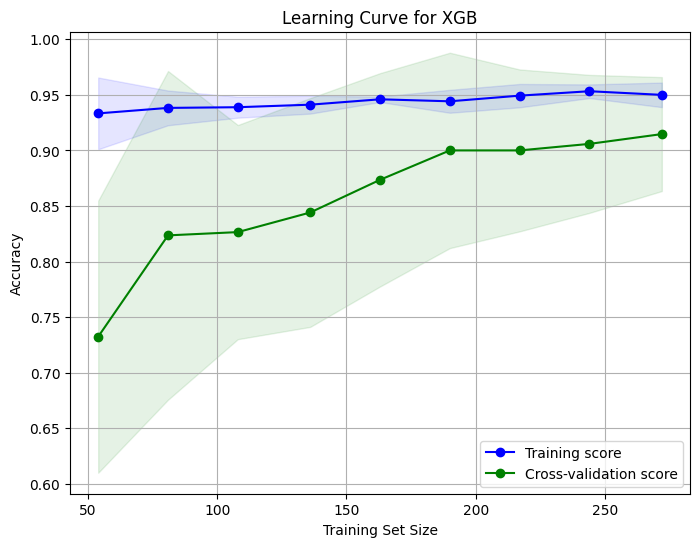
****

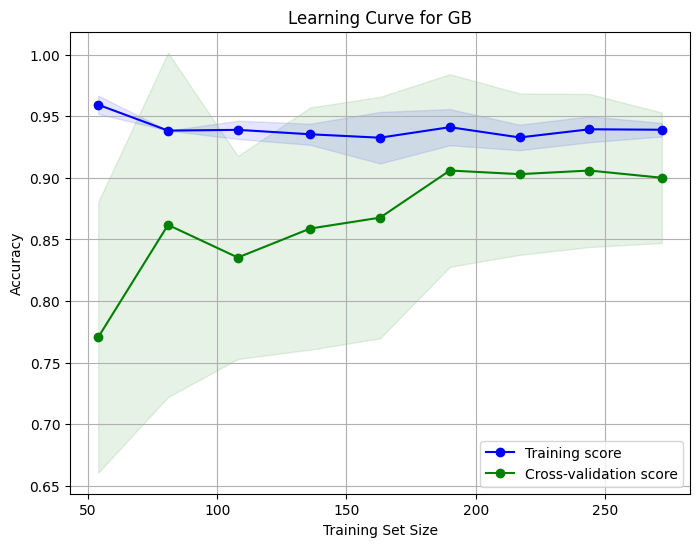
****

****

1. **With Backward Elimination**

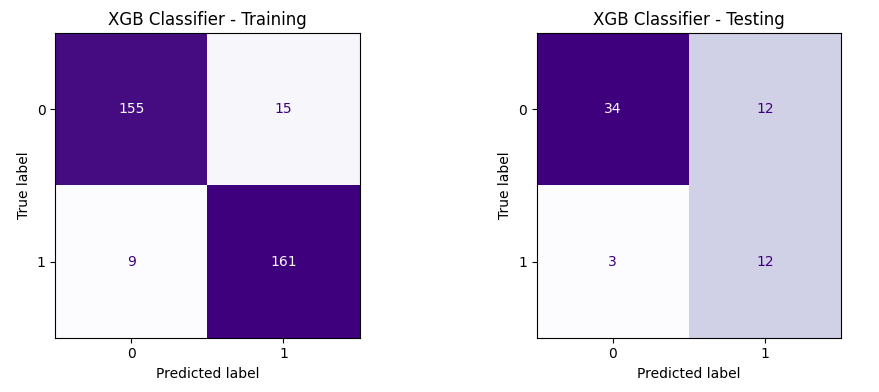
****

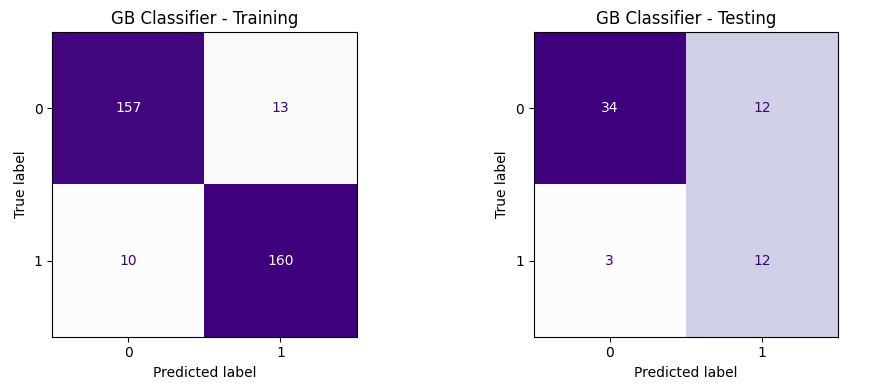
****

****

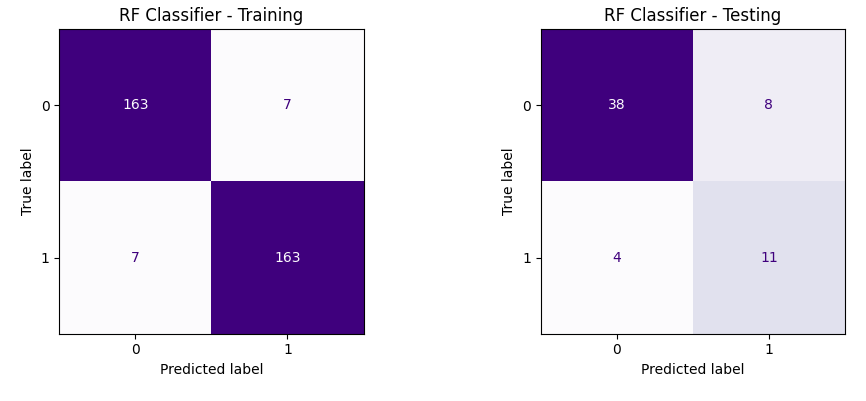
## Confusion Matrices

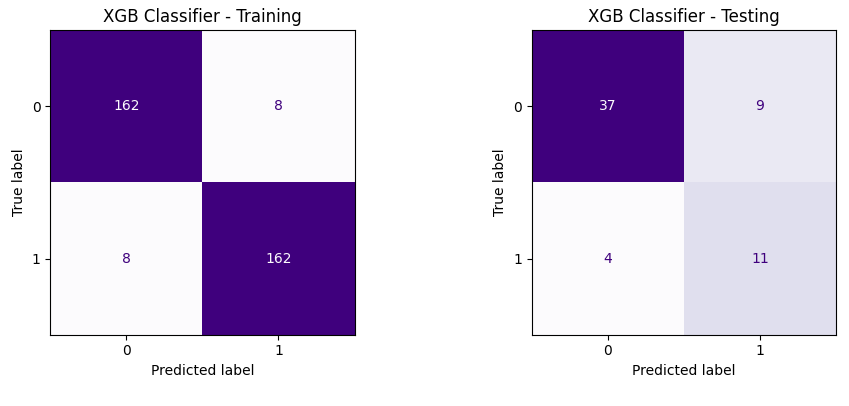
1. **With All Features**

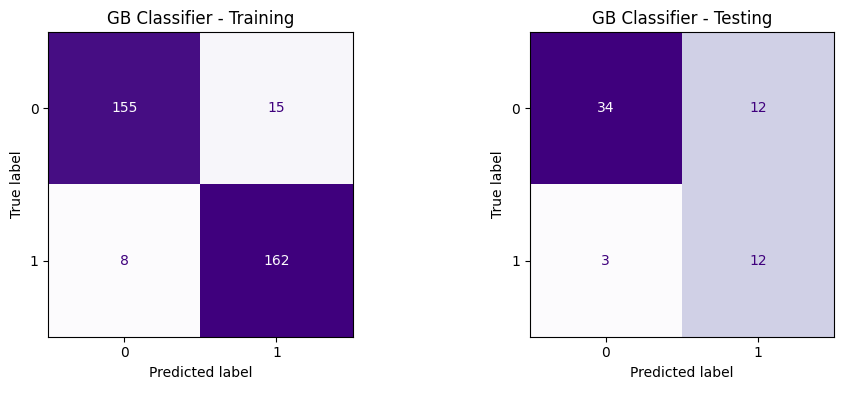




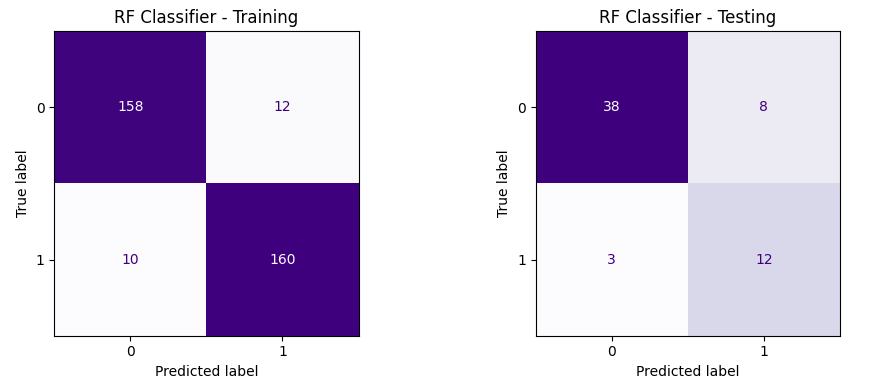
1. **With RFE Feature Selection**

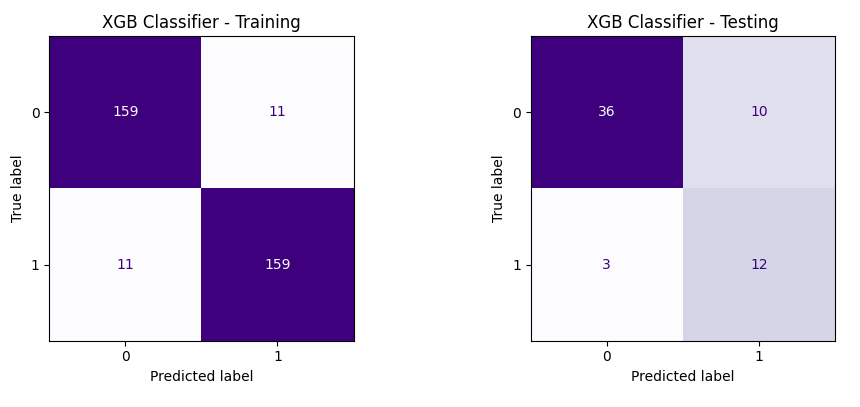
****

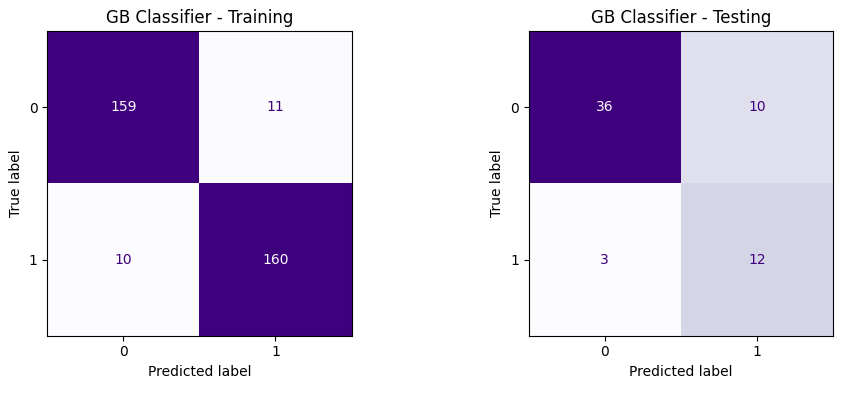
****

****

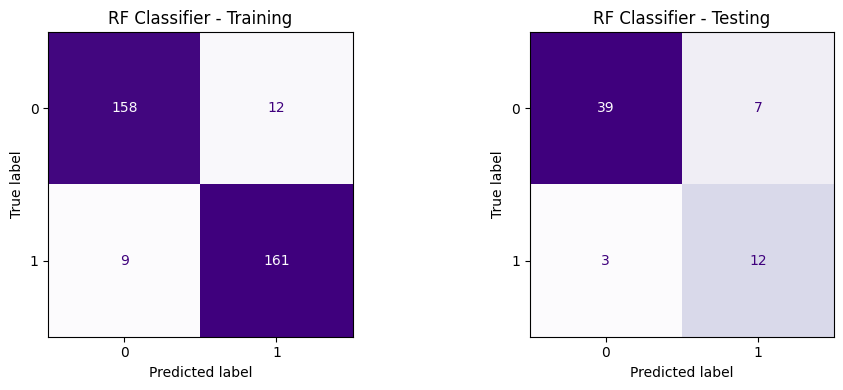
1. **With Forward Selection**

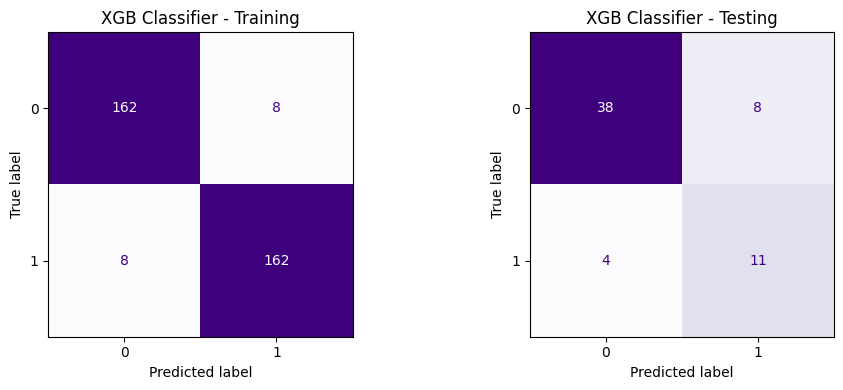
****

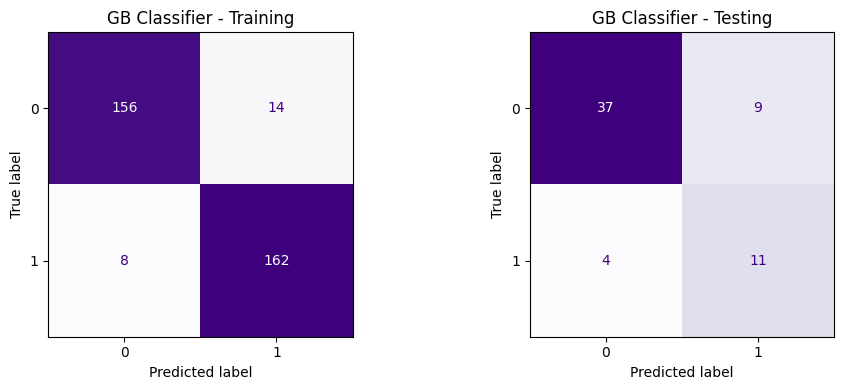
****

****

1. **With Backward Elimination**

****

****



## Code

**1.Data EDA and preprocessing**

# Importing libraries for data handling, visualization, and suppressing warnings

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder

cad\_df = pd.read\_csv('Z-Alizadeh sani dataset.csv')

cad\_df

cad\_df.info()

cad\_df.nunique()

Since the Exertional CP column has just one unique value ('N' for "No") for all 303 patient records, it can be safely removed.

print("\nTarget variable with categories:\n", cad\_df['Cath'].value\_counts())

Diameter narrowing of 50% or more classifies a patient with CAD; otherwise, the patient is considered normal. There are 216 patients with CAD and 87 normal cases.

# \*Data cleaning\*

# Checking for any NaN in the data

print("\nNumber of rows with missing values:\n", cad\_df.isnull().sum().sum())

# Checking for duplicate rows

duplicates = cad\_df.duplicated().sum()

print(f"Number of duplicate rows: {duplicates}")

There are no duplicate rows or missing values in the data.

cad\_df.drop(columns=['Exertional CP'], inplace=True)

cad\_df.shape

# \*Statistical Summary and Heat Map Analysis\*

cad\_df.describe() # Statistical summary

# \*Data Visualization\*

# Count Plot: Cath (Target variable)

sns.countplot(data=cad\_df, palette='viridis', x='Cath')

plt.title('Presence and Absence of CAD')

plt.ylabel('Num. of Patients')

plt.xlabel('0:No CAD, 1:CAD present')

plt.show()

# Histogram: Age distribution

sns.histplot(cad\_df['Age'], bins=50, color='dodgerblue', kde=True)

plt.title('Age Distribution of Patients')

plt.xlabel('Age in years)')

plt.ylabel('Num. of Patients')

plt.show()

# Histogram: Weight distribution

sns.histplot(cad\_df['Weight'], bins=50, color='mediumseagreen', kde=True)

plt.title('Weight Distribution of Patients')

plt.xlabel('Weight(48 to 120kg))')

plt.ylabel('Num. of Patients')

plt.show()

# Count plot: Sex

sns.countplot(data=cad\_df, x='Sex', palette='viridis')

plt.title('Sex')

plt.xlabel('Sex')

plt.ylabel('Num. of Patients')

plt.show()

# Count plot: VHD (valvular heart disease) Normal, mild, moderate, severe

sns.countplot(data=cad\_df, x='VHD', palette='viridis')

plt.title('valvular heart disease')

plt.xlabel('VHD')

plt.ylabel('Num. of Patients')

plt.show()

# Count plot: Bundle Branch Block (Normal, Right Bundle Branch Block (RBBB),Left Bundle Branch Block (LBBB))

sns.countplot(data=cad\_df, x='BBB', palette='viridis')

plt.title('Bundle Branch Block')

plt.xlabel('BBB')

plt.ylabel('Num. of Patients')

plt.show()

# Histogram:BP (blood pressure: mmHg) 90–190

sns.histplot(cad\_df['BP'], bins=50, color='mediumseagreen', kde=True)

plt.title('BP (blood pressure: mmHg) 90–190')

plt.xlabel('BP')

plt.ylabel('Num. of Patients')

plt.show()

The histogram shows that most patients have blood pressure around 130 mmHg

# Count plot: Typical Chest Pain

sns.countplot(data=cad\_df, x='Typical Chest Pain', palette='viridis')

plt.title('Typical Chest Pain ')

plt.xlabel('Typical Chest Pain ')

plt.ylabel('Num. of Patients')

plt.show()

le = LabelEncoder()

for columns in cad\_df.select\_dtypes(include='object'):

cad\_df[columns] = le.fit\_transform(cad\_df[columns])

cad\_df.info()

plt.figure(figsize=(50, 30))

sns.heatmap(cad\_df.corr(), annot=True, fmt=".2f", cmap='viridis')

plt.show()

cad\_df

cad\_df.to\_csv('final\_data.csv', index=False)

**2.With Original Features**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.ensemble import GradientBoostingClassifier

from imblearn.over\_sampling import SMOTENC

from collections import Counter

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.model\_selection import RandomizedSearchCV

import warnings

warnings.filterwarnings('ignore')

cad\_df = pd.read\_csv('final\_data.csv')

cad\_df['Cath'].value\_counts()

# \*\*\*Data splitting\*\*\*

#Separate features and target columns

X\_feature = cad\_df.drop('Cath', axis=1)

y\_target = cad\_df['Cath']

# 80% data for training and 20% data for testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_feature, y\_target, test\_size=0.2, random\_state=1)

print(f"The training set contains {X\_train.shape[0]} patient records and {X\_train.shape[1]} features.")

print(f"The test set contains {X\_test.shape[0]} patient records, each with {X\_test.shape[1]} features.")

# \*\*\*SMOTENC\*\*\*

categorical\_cols = [col for col in X\_train.columns if X\_train[col].nunique() <= 4]

categorical\_indices = [X\_train.columns.get\_loc(col) for col in categorical\_cols]

smote\_nc = SMOTENC(categorical\_features=categorical\_indices, random\_state=1)

X\_balanced, y\_balanced = smote\_nc.fit\_resample(X\_train, y\_train)

print("Imbalanced distribution:", Counter(y\_train))

print("Balanced distribution:", Counter(y\_balanced))

Applied SMOTENC balancing only to the testing set

X\_balanced

X\_balanced.shape

# \*\*\*Random Forest Classifier (RF)\*\*\*

param\_random = {'n\_estimators': [50, 100, 150],

'max\_depth': [2, 3, 4],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [2, 4, 8],

'max\_features': ['sqrt', 'log2']}

RF\_mdl = RandomForestClassifier()

RF\_randomcv = RandomizedSearchCV(RF\_mdl, param\_random, n\_iter=20, cv=5)

RF\_randomcv.fit(X\_balanced, y\_balanced)

RF\_param = RF\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(RF\_randomcv.best\_score\_))

print("Optimal parameters of RF classifier: ", RF\_param)

#Training RF model with optimal parameters

RF\_best\_mdl = RandomForestClassifier(\*\*RF\_param)

RF\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = RF\_best\_mdl.predict(X\_balanced)

training\_acc = RF\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of RF classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = RF\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

# Plotting the training confusion matrix

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('RF Classifier - Training')

# Plotting the testing confusion matrix

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('RF Classifier - Testing')

plt.tight\_layout()

plt.show()

# \*\*\*XGBoost (XGB)\*\*\*

param\_random = {'n\_estimators': [50, 80, 100],

'max\_depth': [2, 3, 4],

'learning\_rate': [0.01, 0.02, 0.05],

'subsample': [0.6, 0.7, 0.8]}

XGB\_mdl = XGBClassifier()

XGB\_randomcv = RandomizedSearchCV(XGB\_mdl, param\_random, n\_iter=20, cv=5)

XGB\_randomcv.fit(X\_balanced, y\_balanced)

XGB\_param = XGB\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(XGB\_randomcv.best\_score\_))

print("Optimal parameters of XGB classifier: ", XGB\_param)

#Training XGB model with optimal parameters

XGB\_best\_mdl = XGBClassifier(\*\*XGB\_param)

XGB\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = XGB\_best\_mdl.predict(X\_balanced)

training\_acc = XGB\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of XGB classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = XGB\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('XGB Classifier - Training')

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('XGB Classifier - Testing')

plt.tight\_layout()

plt.show()

# \*\*\*Gradient Boosting (GB)\*\*\*

param\_random = {'n\_estimators': [50, 100, 150],

'max\_depth': [2, 3, 4],

'learning\_rate': [0.005, 0.001, 0.015],

'subsample': [0.6, 0.7, 0.8],

'min\_samples\_split': [10, 12],

'min\_samples\_leaf': [6, 8, 10]}

GB\_mdl = GradientBoostingClassifier()

GB\_randomcv = RandomizedSearchCV(GB\_mdl, param\_random, n\_iter=20, cv=5)

GB\_randomcv.fit(X\_balanced, y\_balanced)

GB\_param = GB\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(GB\_randomcv.best\_score\_))

print("Optimal parameters of GB classifier: ", GB\_param)

#Training GB model with optimal parameters

GB\_best\_mdl = GradientBoostingClassifier(\*\*GB\_param)

GB\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = GB\_best\_mdl.predict(X\_balanced)

training\_acc = GB\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of GB classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = GB\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('GB Classifier - Training')

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('GB Classifier - Testing')

plt.tight\_layout()

plt.show()

**3.RFE feature selection**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.feature\_selection import RFE

from imblearn.over\_sampling import SMOTENC

from collections import Counter

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.model\_selection import RandomizedSearchCV

import warnings

warnings.filterwarnings('ignore')

cad\_df = pd.read\_csv('final\_data.csv')

cad\_df['Cath'].value\_counts()

# \*\*\*Recursive Feature Elimination (RFE)\*\*\*

#Separate features and target columns

X\_feature = cad\_df.drop('Cath', axis=1)

y\_target = cad\_df['Cath']

RF\_mdl = RandomForestClassifier(random\_state=1)

select = RFE(RF\_mdl, n\_features\_to\_select=30) # RFE with 30 features selected

select.fit(X\_feature, y\_target)

selected\_features = X\_feature.columns[select.support\_]

X\_rfe\_chosen = X\_feature[selected\_features]

print("Selected features with RFE:", selected\_features.tolist())

print("\nDataframe with only the selected features:")

print(X\_rfe\_chosen.head(5))

# \*\*\*Data splitting\*\*\*

# 80% data for training and 20% data for testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_rfe\_chosen, y\_target, test\_size=0.2, random\_state=1)

print(f"The training set contains {X\_train.shape[0]} patient records and {X\_train.shape[1]} features.")

print(f"The test set contains {X\_test.shape[0]} patient records, each with {X\_test.shape[1]} features.")

# \*\*\*SMOTENC\*\*\*

categorical\_cols = [col for col in X\_train.columns if X\_train[col].nunique() <= 4]

categorical\_indices = [X\_train.columns.get\_loc(col) for col in categorical\_cols]

smote\_nc = SMOTENC(categorical\_features=categorical\_indices, random\_state=1)

X\_balanced, y\_balanced = smote\_nc.fit\_resample(X\_train, y\_train)

print("Imbalanced distribution:", Counter(y\_train))

print("Balanced distribution:", Counter(y\_balanced))

Applied SMOTENC balancing only to the testing set

X\_balanced

# \*\*\*Random Forest Classifier (RF)\*\*\*

param\_random = {'n\_estimators': [50, 100, 150],

'max\_depth': [2, 3, 4],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [2, 4, 8],

'max\_features': ['sqrt', 'log2']}

RF\_mdl = RandomForestClassifier()

RF\_randomcv = RandomizedSearchCV(RF\_mdl, param\_random, n\_iter=20, cv=5)

RF\_randomcv.fit(X\_balanced, y\_balanced)

RF\_param = RF\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(RF\_randomcv.best\_score\_))

print("Optimal parameters of RF classifier: ", RF\_param)

#Training RF model with optimal parameters

RF\_best\_mdl = RandomForestClassifier(\*\*RF\_param)

RF\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = RF\_best\_mdl.predict(X\_balanced)

training\_acc = RF\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of RF classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = RF\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

# Plotting the training confusion matrix

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('RF Classifier - Training')

# Plotting the testing confusion matrix

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('RF Classifier - Testing')

plt.tight\_layout()

plt.show()

# \*\*\*XGBoost (XGB)\*\*\*

param\_random = {'n\_estimators': [50, 80, 100],

'max\_depth': [2, 3, 4],

'learning\_rate': [0.01, 0.02, 0.05],

'subsample': [0.6, 0.7, 0.8]}

XGB\_mdl = XGBClassifier()

XGB\_randomcv = RandomizedSearchCV(XGB\_mdl, param\_random, n\_iter=20, cv=5)

XGB\_randomcv.fit(X\_balanced, y\_balanced)

XGB\_param = XGB\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(XGB\_randomcv.best\_score\_))

print("Optimal parameters of XGB classifier: ", XGB\_param)

#Training XGB model with optimal parameters

XGB\_best\_mdl = XGBClassifier(\*\*XGB\_param)

XGB\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = XGB\_best\_mdl.predict(X\_balanced)

training\_acc = XGB\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of XGB classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = XGB\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('XGB Classifier - Training')

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('XGB Classifier - Testing')

plt.tight\_layout()

plt.show()

# \*\*\*Gradient Boosting (GB)\*\*\*

param\_random = {'n\_estimators': [50, 100, 150],

'max\_depth': [2, 3, 4],

'learning\_rate': [0.005, 0.001, 0.015],

'subsample': [0.6, 0.7, 0.8],

'min\_samples\_split': [10, 12],

'min\_samples\_leaf': [6, 8, 10]}

GB\_mdl = GradientBoostingClassifier()

GB\_randomcv = RandomizedSearchCV(GB\_mdl, param\_random, n\_iter=20, cv=5)

GB\_randomcv.fit(X\_balanced, y\_balanced)

GB\_param = GB\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(GB\_randomcv.best\_score\_))

print("Optimal parameters of GB classifier: ", GB\_param)

#Training GB model with optimal parameters

GB\_best\_mdl = GradientBoostingClassifier(\*\*GB\_param)

GB\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = GB\_best\_mdl.predict(X\_balanced)

training\_acc = GB\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of GB classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = GB\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('GB Classifier - Training')

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('GB Classifier - Testing')

plt.tight\_layout()

plt.show()

**4.Forward Selection**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.ensemble import GradientBoostingClassifier

from mlxtend.feature\_selection import SequentialFeatureSelector

from imblearn.over\_sampling import SMOTENC

from collections import Counter

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.model\_selection import RandomizedSearchCV

import warnings

warnings.filterwarnings('ignore')

cad\_df = pd.read\_csv('final\_data.csv')

cad\_df['Cath'].value\_counts()

# \*\*\*Forward Selection\*\*\*

#Separate features and target columns

X\_feature = cad\_df.drop('Cath', axis=1)

y\_target = cad\_df['Cath']

rf\_model = RandomForestClassifier(random\_state=1)

forward\_selector = SequentialFeatureSelector(estimator=rf\_model, k\_features=30, forward=True, floating=False,scoring='accuracy', cv=2)

select\_fit = forward\_selector.fit(X\_feature, y\_target)

selected\_features = X\_feature.columns[list(select\_fit.k\_feature\_idx\_)]

X\_forward = X\_feature[selected\_features]

print('Selected features with Forward selection: ', selected\_features)

print("\nDataframe with only the selected features:")

print(X\_forward.head(5))

# \*\*\*Data splitting\*\*\*

# 80% data for training and 20% data for testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_forward, y\_target, test\_size=0.2, random\_state=1)

print(f"The training set contains {X\_train.shape[0]} patient records and {X\_train.shape[1]} features.")

print(f"The test set contains {X\_test.shape[0]} patient records, each with {X\_test.shape[1]} features.")

# \*\*\*SMOTENC\*\*\*

categorical\_cols = [col for col in X\_train.columns if X\_train[col].nunique() <= 4]

categorical\_indices = [X\_train.columns.get\_loc(col) for col in categorical\_cols]

smote\_nc = SMOTENC(categorical\_features=categorical\_indices, random\_state=1)

X\_balanced, y\_balanced = smote\_nc.fit\_resample(X\_train, y\_train)

print("Imbalanced distribution:", Counter(y\_train))

print("Balanced distribution:", Counter(y\_balanced))

Applied SMOTENC balancing only to the testing set

X\_balanced

# \*\*\*Random Forest Classifier (RF)\*\*\*

param\_random = {'n\_estimators': [50, 100, 150],

'max\_depth': [2, 3, 4],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [2, 4, 8],

'max\_features': ['sqrt', 'log2']}

RF\_mdl = RandomForestClassifier()

RF\_randomcv = RandomizedSearchCV(RF\_mdl, param\_random, n\_iter=20, cv=5)

RF\_randomcv.fit(X\_balanced, y\_balanced)

RF\_param = RF\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(RF\_randomcv.best\_score\_))

print("Optimal parameters of RF classifier: ", RF\_param)

#Training RF model with optimal parameters

RF\_best\_mdl = RandomForestClassifier(\*\*RF\_param)

RF\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = RF\_best\_mdl.predict(X\_balanced)

training\_acc = RF\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of RF classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = RF\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

# Plotting the training confusion matrix

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('RF Classifier - Training')

# Plotting the testing confusion matrix

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('RF Classifier - Testing')

plt.tight\_layout()

plt.show()

# \*\*\*XGBoost (XGB)\*\*\*

param\_random = {'n\_estimators': [50, 80, 100],

'max\_depth': [2, 3, 4],

'learning\_rate': [0.01, 0.02, 0.05],

'subsample': [0.6, 0.7, 0.8]}

XGB\_mdl = XGBClassifier()

XGB\_randomcv = RandomizedSearchCV(XGB\_mdl, param\_random, n\_iter=20, cv=5)

XGB\_randomcv.fit(X\_balanced, y\_balanced)

XGB\_param = XGB\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(XGB\_randomcv.best\_score\_))

print("Optimal parameters of XGB classifier: ", XGB\_param)

#Training XGB model with optimal parameters

XGB\_best\_mdl = XGBClassifier(\*\*XGB\_param)

XGB\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = XGB\_best\_mdl.predict(X\_balanced)

training\_acc = XGB\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of XGB classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = XGB\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('XGB Classifier - Training')

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('XGB Classifier - Testing')

plt.tight\_layout()

plt.show()

# \*\*\*Gradient Boosting (GB)\*\*\*

param\_random = {'n\_estimators': [50, 100, 150],

'max\_depth': [2, 3, 4],

'learning\_rate': [0.005, 0.001, 0.015],

'subsample': [0.6, 0.7, 0.8],

'min\_samples\_split': [10, 12],

'min\_samples\_leaf': [6, 8, 10]}

GB\_mdl = GradientBoostingClassifier()

GB\_randomcv = RandomizedSearchCV(GB\_mdl, param\_random, n\_iter=20, cv=5)

GB\_randomcv.fit(X\_balanced, y\_balanced)

GB\_param = GB\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(GB\_randomcv.best\_score\_))

print("Optimal parameters of GB classifier: ", GB\_param)

#Training GB model with optimal parameters

GB\_best\_mdl = GradientBoostingClassifier(\*\*GB\_param)

GB\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = GB\_best\_mdl.predict(X\_balanced)

training\_acc = GB\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of GB classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = GB\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('GB Classifier - Training')

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('GB Classifier - Testing')

plt.tight\_layout()

plt.show()

**5.Backward Elimination**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.ensemble import GradientBoostingClassifier

from mlxtend.feature\_selection import SequentialFeatureSelector

from imblearn.over\_sampling import SMOTENC

from collections import Counter

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.model\_selection import RandomizedSearchCV

import warnings

warnings.filterwarnings('ignore')

cad\_df = pd.read\_csv('final\_data.csv')

cad\_df['Cath'].value\_counts()

# \*\*\*Backward Elimination\*\*\*

#Separate features and target columns

X\_feature = cad\_df.drop('Cath', axis=1)

y\_target = cad\_df['Cath']

rf\_model = RandomForestClassifier(random\_state=1)

forward\_selector = SequentialFeatureSelector(estimator=rf\_model, k\_features=30, forward=False, floating=False, scoring='accuracy', cv=2)

select\_fit = forward\_selector.fit(X\_feature, y\_target)

selected\_features = X\_feature.columns[list(select\_fit.k\_feature\_idx\_)]

X\_backward = X\_feature[selected\_features]

print('Selected features with Backward Elimination: ', selected\_features)

print("\nDataframe with only the selected features:")

print(X\_backward.head(5))

# \*\*\*Data splitting\*\*\*

# 80% data for training and 20% data for testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_backward, y\_target, test\_size=0.2, random\_state=1)

print(f"The training set contains {X\_train.shape[0]} patient records and {X\_train.shape[1]} features.")

print(f"The test set contains {X\_test.shape[0]} patient records, each with {X\_test.shape[1]} features.")

# \*\*\*SMOTENC\*\*\*

categorical\_cols = [col for col in X\_train.columns if X\_train[col].nunique() <= 4]

categorical\_indices = [X\_train.columns.get\_loc(col) for col in categorical\_cols]

smote\_nc = SMOTENC(categorical\_features=categorical\_indices, random\_state=1)

X\_balanced, y\_balanced = smote\_nc.fit\_resample(X\_train, y\_train)

print("Imbalanced distribution:", Counter(y\_train))

print("Balanced distribution:", Counter(y\_balanced))

Applied SMOTENC balancing only to the testing set

X\_balanced

# \*\*\*Random Forest Classifier (RF)\*\*\*

param\_random = {'n\_estimators': [50, 100, 150],

'max\_depth': [2, 3, 4],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [2, 4, 8],

'max\_features': ['sqrt', 'log2']}

RF\_mdl = RandomForestClassifier()

RF\_randomcv = RandomizedSearchCV(RF\_mdl, param\_random, n\_iter=20, cv=5)

RF\_randomcv.fit(X\_balanced, y\_balanced)

RF\_param = RF\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(RF\_randomcv.best\_score\_))

print("Optimal parameters of RF classifier: ", RF\_param)

#Training RF model with optimal parameters

RF\_best\_mdl = RandomForestClassifier(\*\*RF\_param)

RF\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = RF\_best\_mdl.predict(X\_balanced)

training\_acc = RF\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of RF classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = RF\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

# Plotting the training confusion matrix

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('RF Classifier - Training')

# Plotting the testing confusion matrix

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('RF Classifier - Testing')

plt.tight\_layout()

plt.show()

# \*\*\*XGBoost (XGB)\*\*\*

param\_random = {'n\_estimators': [50, 80, 100],

'max\_depth': [2, 3, 4],

'learning\_rate': [0.01, 0.02, 0.05],

'subsample': [0.6, 0.7, 0.8]}

XGB\_mdl = XGBClassifier()

XGB\_randomcv = RandomizedSearchCV(XGB\_mdl, param\_random, n\_iter=20, cv=5)

XGB\_randomcv.fit(X\_balanced, y\_balanced)

XGB\_param = XGB\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(XGB\_randomcv.best\_score\_))

print("Optimal parameters of XGB classifier: ", XGB\_param)

#Training XGB model with optimal parameters

XGB\_best\_mdl = XGBClassifier(\*\*XGB\_param)

XGB\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = XGB\_best\_mdl.predict(X\_balanced)

training\_acc = XGB\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of XGB classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = XGB\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('XGB Classifier - Training')

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('XGB Classifier - Testing')

plt.tight\_layout()

plt.show()

# \*\*\*Gradient Boosting (GB)\*\*\*

param\_random = {'n\_estimators': [50, 100, 150],

'max\_depth': [2, 3, 4],

'learning\_rate': [0.005, 0.001, 0.015],

'subsample': [0.6, 0.7, 0.8],

'min\_samples\_split': [10, 12],

'min\_samples\_leaf': [6, 8, 10]}

GB\_mdl = GradientBoostingClassifier()

GB\_randomcv = RandomizedSearchCV(GB\_mdl, param\_random, n\_iter=20, cv=5)

GB\_randomcv.fit(X\_balanced, y\_balanced)

GB\_param = GB\_randomcv.best\_params\_

print("Best cross-validation score: {:.2f}".format(GB\_randomcv.best\_score\_))

print("Optimal parameters of GB classifier: ", GB\_param)

#Training GB model with optimal parameters

GB\_best\_mdl = GradientBoostingClassifier(\*\*GB\_param)

GB\_best\_mdl.fit(X\_balanced, y\_balanced)

# Training performance

train\_pred = GB\_best\_mdl.predict(X\_balanced)

training\_acc = GB\_best\_mdl.score(X\_balanced, y\_balanced)

print(f"Training phase accuracy of GB classifier: {training\_acc:.4f}")

print("=== Training Report ===")

print(classification\_report(y\_balanced, train\_pred))

# Testing performance

test\_pred = GB\_best\_mdl.predict(X\_test)

print("=== Testing Report ===")

print(classification\_report(y\_test, test\_pred))

matrix\_train = confusion\_matrix(y\_balanced, train\_pred)

matrix\_test = confusion\_matrix(y\_test, test\_pred)

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

disp\_val = ConfusionMatrixDisplay(confusion\_matrix=matrix\_train, display\_labels=[0, 1])

disp\_val.plot(ax=axs[0], cmap='Purples', colorbar=False)

axs[0].set\_title('GB Classifier - Training')

disp\_test = ConfusionMatrixDisplay(confusion\_matrix=matrix\_test, display\_labels=[0, 1])

disp\_test.plot(ax=axs[1], cmap='Purples', colorbar=False)

axs[1].set\_title('GB Classifier - Testing')

plt.tight\_layout()

plt.show()