International Journal of Computer Vision and Signal Processing, 15(1), 90-100(2025) ORIGINAL ARTICLE

Data-Driven Diagnosis: Feature Engineering and Hyperparameter Tuning for Imbalanced Cardiovascular Disease Classification

Ankon Karmokar*

Department of Computer Science & Engineering
Jagannath University, Dhaka-1100, Bangladesh

Md. Manowarul Islam*, Arnisha Akhter, Uzzal Kumar Acharjee

Department of Computer Science & Engineering
Jagannath University, Dhaka-1100, Bangladesh

Abstract

Cardiovascular diseases (CVDs) have become the leading cause of death all over the world, emphasizing the need for prediction models to ensure accurate and timely medical interventions. In this paper, we propose a powerful end-to-end machine learning system that incorporates feature engineering, hyperparameter tuning, and ensemble model analysis to enhance the predictive performance of CVD. Preprocessing encompasses feature engineering methods designed to improve data quality, remove outliers, cap values, and normalize data. Five additional complex models—Random Forest (87.43%), Gradient Boosting (88.07%), AdaBoost (87.01%), XGBoost (88.11%), and LightGBM (88.02%)—are fine-tuned using the RandomizedSearchCV innervation library. XGBoost achieves the highest validation accuracy (88.11%) and is the most effective classifier. The proposed method offers a significant advantage over the conventional one in terms of precision, sensitivity, and F1-score, which can be applied to the screening, prevention, and clinical decision-making in cardiovascular healthcare.

Contribution of the Paper: Key contributions are that data quality can be improved through sophisticated feature engineering, model performance through hyperparameter optimization, and model interpretability via SHAP analysis for better decision-making, especially in sensitive areas such as healthcare.

Keywords: Cardiovascular Disease Classification, Feature Engineering, Machine Learning, XGBoost, Hyperparameter Tuning

IJCVSP

ISSN: 2186-1390 (Online)
http://cennser.org/IJCVSP

Article History:
Received: 10/4/2025
Revised: 31/7/2025
Accepted: 1/11/2025
Published Online: 23/11/2025

1. Introduction

Cardiovascular diseases cause an estimated 17.9 million deaths annually, are the leading cause of death throughout the world, and are responsible for more than 32% of all deaths all over the world (World Health Organization, 2021) [1]. So, it is necessary to expect and avoid CVD to prevent such catastrophic consequences. Recently, machine learning (ML) methods have emerged as one of the most effective means of medical diagnosis, as they can extract complex patterns from large datasets and facilitate data-driven clinical decisions [2]. Several studies have used a comprehensive pipeline that incorporates machine learning and deep learning techniques to predict cardiovascular disease. They typically involve preprocessing, extracting characteristics, and using various classifiers to enhance diagnosis determination. These ensemble methods, such as Random Forest, Decision Trees, K-Nearest Neighbors, XGBoost, and deep neural networks, have shown great promise for learning from clinical datasets in cardiovas-

*Corresponding author
Email addresses: b190305039@cse.jnu.ac.bd (Ankon Karmokar), manowar@cse.jnu.ac.bd (Md. Manowarul Islam), arnisha@cse.jnu.ac.bd (Arnisha Akhter), uzzal@cse.jnu.ac.bd (Uzzal Kumar Acharjee)

cular disease detection [3]. Tuning the hyperparameter is crucial to improve the performance of ML models, particularly ensemble-based methods, in predicting CVD. Of the methods mentioned, GridSearchCV and RandomizedSearchCV are the most commonly used to fine-tune model parameters using cross-validation approaches. They also note that in the grid search, a full search of all settings for fine-tuning parameters is conducted to identify the best one. In a random search, a random sample is explored to avoid computationally expensive constraints. An important one is cross-validation, such as RepeatedStratifiedKFold. When combined with a grid and randomized search, the ensembles consider it significant to learn more relevant predictive models, particularly for the intradataset application with ensemble classifiers such as RF and AdaBoost in cardiovascular condition detection scenarios [4].

This paper utilizes an ensemble learning model to predict CVD, incorporating hyperparameter optimization and feature engineering. In other words, the minority class in the imbalanced dataset is promoted to make accurate predictions. This technique, when combined with hyperparameter tuning, significantly contributes to the early diagnosis and intervention of cardiovascular disease by enhancing the collection of clinical data. The significant contributions of this work are as follows:

- **Comprehensive Feature Engineering:** Encoded categorical features, conducted chi-square testing, detected outliers using Z-score and IQR methods, and removed and capped the outliers for better data quality and model performance.
- **Hyperparameter Tuning:** Performed hyperparameter tuning using cross-validated RandomizedSearchCV of different classifiers to make models more efficient, accurate, and generalized.
- **Model Interpretability with SHAP:** Applying SHAP analysis to enhance model interpretability has provided valid interpretations of feature importance, making the decision process more understandable, especially in sensitive use cases such as healthcare.

2. Literature Review

The high mortality from cardiovascular disease and the concern with it make these diseases good candidates for prediction schemes in health care. Machine learning has been successfully applied, even in the early days, to predict heart disease in this specific domain, enabling quick but vitally necessary care.

Mesquita and Marques (2024) proposed a machine learning model that is explainable to predict heart disease. This approach serves to delay the model's interpretability for medical decision-making in health care. They also realized the value of transparent models in medical diagnostics, allowing doctors to trust the outcome [5].

Waigi et al. (2020) employed various machine learning techniques, including decision trees, to predict heart disease based on the Kaggle dataset. The results yielded an accuracy of 72.77%, thus shedding light on the issues surrounding the application of machine learning to the prediction of the parameters of heart disease [6].

Rani et al. (2021) developed a decision support system to predict heart disease based on machine learning approaches, including Naive Bayes, Support Vector Machines (SVM), Logistic Regression, and Random Forest. The use of specific classifiers for the diagnosis of heart disease was confirmed by this study, with Random Forest achieving an accuracy of 86.60% [7].

Miranda et al. (2021) report the use of logistic regression and SGD for the early detection of heart disease. They observed that, although the test results for such a simple model were preliminary, they were promising, implying that simple models could be practical when resources are scarce [8].

Shorewala (2021) implemented ensemble techniques in the early detection of coronary heart disease. The results also highlighted that ensemble methods, such as stacking, should improve the accuracy and stability of coronary heart disease prediction compared to single models [9].

Chen (2024) used the Kaggle dataset [10] using several ML algorithms, e.g., Decision Trees, LightGBM, Random Forest, and Logistic Regression. The model exhibited that LightGBM had the highest prediction accuracy of 76.9%. The authors emphasized the importance of data preprocessing and feature selection to enhance model performance [11].

3. Methodology

Figure 1 shows a proposed block diagram of the general framework, the classification method, and the prediction strategy for cardiovascular disease.

3.1. Dataset

In this paper, we employed the "heart_2020_cleaned.csv" dataset taken from the 'Personal Key Indicators of Heart Disease' dataset on Kaggle [10]. It involves 3,19,795 records with health characteristics, including BMI, drinking behavior, smoking status, physical activity, and diseases. Additionally, the data are highly imbalanced, with 91.4% of the instances classified as 'No' and only 8.6% as 'Yes', which complicates the prediction modeling. The data were preprocessed as follows: missing values were checked, outliers were removed and capped, and attribute selection was used to reduce the quality of the data. We believe that because the dataset's medical condition labels are self-reports, they provide valuable insights for creating this high-performance CVD prediction model. The dataset's limitations may affect the generalizability of the models learned from it, given its self-reported nature, the lack of

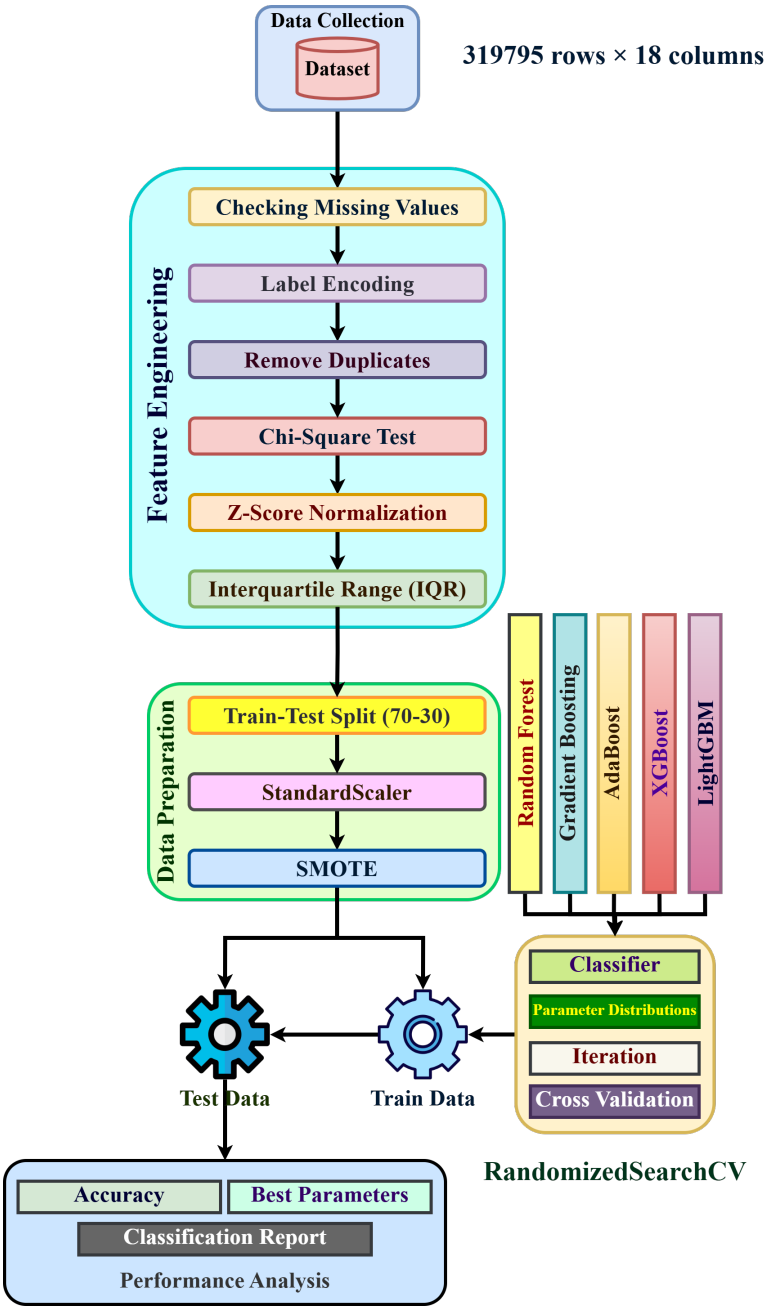

Figure 1: Block Diagram for Cardiovascular Disease Prediction

features (e.g., cholesterol, blood pressure), and the imbalance of class distribution. Outliers and nulls can also affect model performance. Additionally, the lack of data on critical clinical variables and potential confounders limits the model's generalizability across different populations. Lastly, the dataset's temporal scope is limited, which may hinder its generalizability to future predictions.

3.2. Feature Engineering

Feature engineering converts raw data into a form that a machine learning model can utilize. This includes handling missing values, encoding categorical variables, removing duplicates, and normalizing the information. These steps work toward a cleaner, more relevant, and analysis-ready dataset. Preprocessing helps enhance the quality of features for machine learning models.

3.2.1. Checking Missing Values

To ensure data integrity, missing values must be addressed and accounted for. The missing value can be imputed, such as with the mean, median, or mode, or rows or columns can be removed. This prevents the model from being biased or inaccurate. Unbiased imputation guarantees the consistency and trustworthiness of the data.

3.2.2. Label Encoding

Label encoding enables data transformation to a numeric form, which helps prepare the dataset for machine learning models. This method converts each category to a unique number, which the model ingests. It is widely used when the model does not directly support categorical variables. It helps to work with categorical features in regression and decision tree models.

3.2.3. Removing Duplicates

We remove duplicate records in the dataset to avoid duplication and model-training bias. Duplicates may bias analysis for some tasks, for instance, by providing a duplication-weighted overrepresentation of identical observations. Having distinct data points helps the model perform better, and the system operates more efficiently. This step enhances the accuracy of the dataset.

3.2.4. Chi-Square Test

This test measures the independence between a categorical feature and the class. It helps in discovering significant features that are statistically associated with the target. Features with low p-values were considered necessary for the model. This test is suggested for feature selection and dimension reduction [12].

Table 1 shows significant associations between the other characteristics and the target variable, with most of the p-values close to zero.

Figure 2 shows the distribution of BMI, PhysicalHealth, MentalHealth, and SleepTime. BMI is symmetrically distributed, with a peak around 30. PhysicalHealth and MentalHealth are off-balance towards lower values, suggesting fewer complaints. The clusters of SleepTime response have several peaks, representing typical sleep durations.

Table 1: Chi-Square Test Results

Feature	Chi-Square Statistic	p-value	Degrees of Freedom
HeartDisease	301704.833079	0.000000e+00	1.0
Smoking	3295.579216	0.000000e+00	1.0
AlcoholDrinking	396.828516	2.699813e-88	1.0
Stroke	11429.939506	0.000000e+00	1.0
DiffWalking	11638.546630	0.000000e+00	1.0
Sex	1671.147135	0.000000e+00	1.0
AgeCategory	18912.371040	0.000000e+00	12.0
Race	1030.058381	1.866553e-220	5.0
Diabetic	9862.118540	0.000000e+00	3.0
PhysicalActivity	2642.383094	0.000000e+00	1.0
GenHealth	19421.558082	0.000000e+00	4.0
Asthma	385.984033	6.196831e-86	1.0
KidneyDisease	6138.935909	0.000000e+00	1.0
SkinCancer	2477.966898	0.000000e+00	1.0

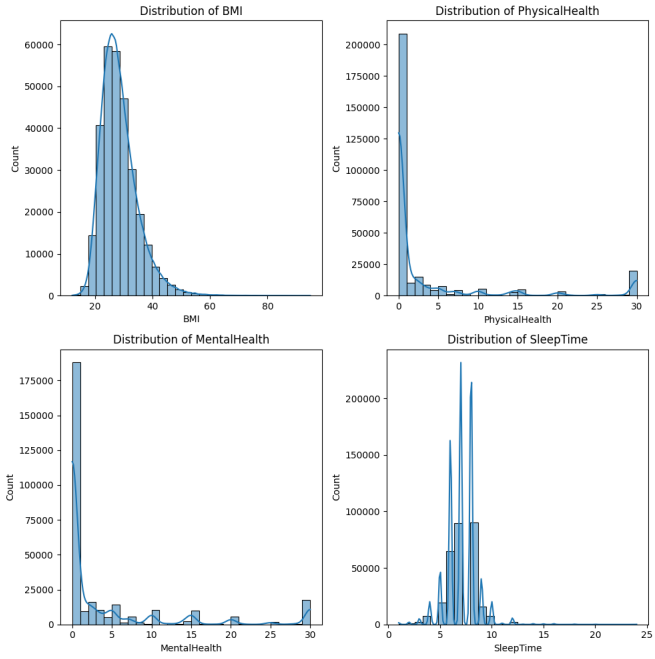

Figure 2: Distribution of Numerical Features

3.2.5. Z-Score Normalization

Z-score Normalization normalizes the data by scaling it to have a mean of 0 and a standard deviation of 1. This can be useful for scale-sensitive models. Outliers are identified and either removed or capped based on their Z-scores. It ensures that all features have the same effect on the model. The Z-score for each data point is calculated as [13]:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where x is the data point, μ is the mean, and σ is the standard deviation. Outliers are defined by $|z| > 3$, with

Class 1 outliers capped at $\mu \pm 3\sigma$, and Class 0 outliers removed. The threshold is chosen as $|z| > 3$ to capture abnormal patterns because 99.7% of the data in a normal distribution fall within three standard deviations from the mean. Outliers are data points beyond this range and are extreme, which could influence the results [14].

Algorithm 1 Outlier Handling Using Z-Score Normalization

Input: DataFrame df, Target Column 'HeartDisease', Threshold $z\_threshold = 3$
for Each Column in Numerical Columns **do**
  Calculate Z-Score: $z = \frac{x-\mu}{\sigma}$
  if Class == 1 **then**
    Cap Values Outside Bounds:
      If $z < -z\_threshold$, set value to $\mu - z\_threshold \times \sigma$
      If $z > z\_threshold$, set value to $\mu + z\_threshold \times \sigma$
  else if Class == 0 **then**
    Remove Rows with Outliers: If $|z| > z\_threshold$, Remove Row
  end if
end for
Output: DataFrame with Handled Outliers

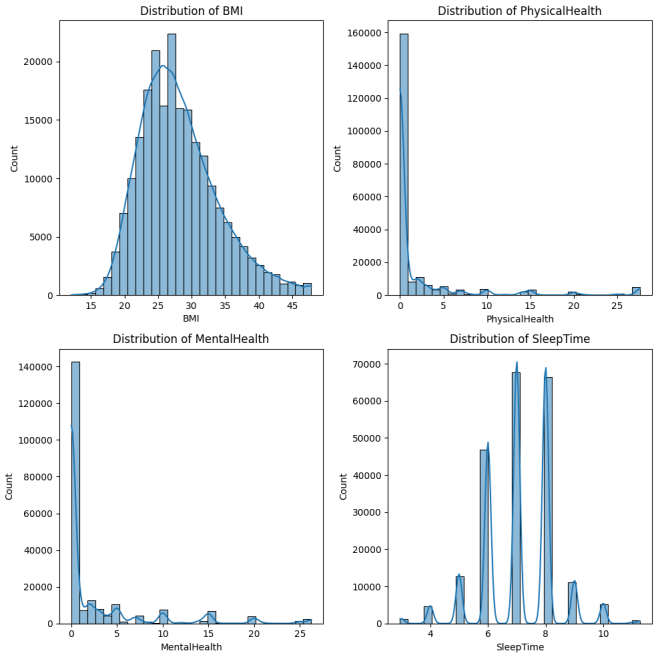

Figure 3: Distribution of Numerical Features (After Z-Score)

In Figure 3, after removing outliers using Z-scores and capping minority class outliers, the BMI distribution indicates a more balanced spread. PhysicalHealth and MentalHealth distributions also show reduced skewness as their extreme values have been capped, while SleepTime has now shown smoother peaks.

3.2.6. Interquartile Range (IQR)

A commonly used technique in statistics to identify and treat outliers in a dataset is the Interquartile Range (IQR) method. IQR is the difference between the third quartile (Q3) and the first quartile (Q1) [13]:

$$\text{IQR} = Q3 - Q1 \tag{2}$$

In general, outliers are observations outside the lower and upper bounds of the range.

$$\text{Lower Bound} = Q1 - 1.5 \times \text{IQR} \tag{3}$$

$$\text{Upper Bound} = Q3 + 1.5 \times \text{IQR} \tag{4}$$

In this work, the IQR method is used in two categories for outlier detection and removal. For one class (Class 1), outliers are capped at low and high values, and for the other (Class 0), they are completely discarded.

Algorithm 2 Outlier Handling Using Interquartile Range (IQR)

Input: DataFrame df, Target Column 'HeartDisease'
for Each Column in Numerical Columns **do**
  Calculate the First Quartile: $Q1 = \text{quantile}(0.25)$
  Calculate the Third Quartile: $Q3 = \text{quantile}(0.75)$
  Calculate IQR: $IQR = Q3 - Q1$
  Define Lower and Upper Bounds:
    $\text{lower\_bound} = Q1 - 1.5 \times IQR$
    $\text{upper\_bound} = Q3 + 1.5 \times IQR$
  if Class == 1 **then**
    Cap Values Outside Bounds for Class '1':
      If $df[col] <$ lower_bound, Set Value to lower_bound
      If $df[col] >$ upper_bound, Set Value to upper_bound
  else if Class == 0 **then**
    Remove Rows with Outliers for Class '0':
      If $df[col] <$ lower_bound or $df[col] >$ upper_bound, Remove Row
  end if
end for
Output: DataFrame with Handled Outliers

Figure 4 demonstrates a symmetric BMI distribution, with a central peak approximately at 30. PhysicalHealth and MentalHealth are very imbalanced, with concentrated values close to 0. SleepTime has several clear peaks that correspond to the most common sleep times.

3.3. Data Preparation

In the data preparation stage, we split the training and testing sets while preserving the class structure through stratification. The features are normalized using 'StandardScaler', and SMOTE addresses the class imbalance by generating artificial samples for the minority class. These processes make the dataset balanced and uniform for model training.

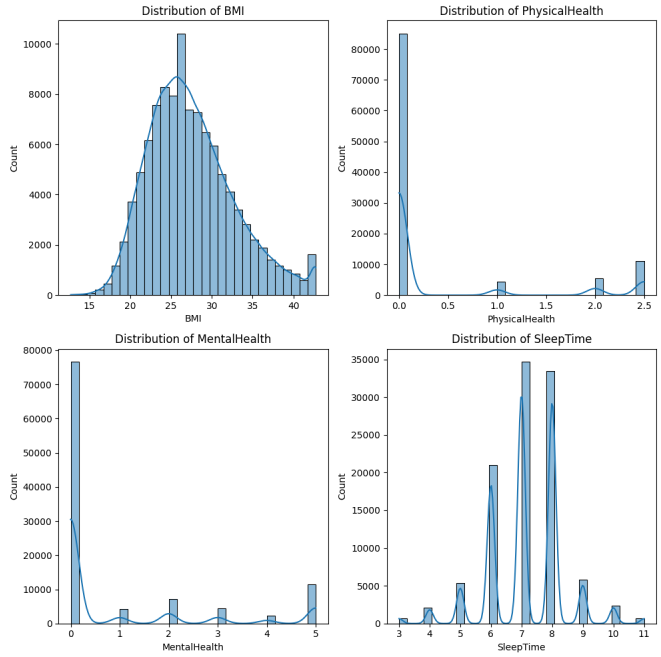

Figure 4: Distribution of Numerical Features (After IQR)

3.4. Machine Learning

Random Forest (RF): RF generates multiple decision trees trained on different random subsets of data. The final prediction is the sum of the forecasts by individual trees, reducing variance and increasing accuracy [7]:

$$\hat{y}_{RF} = \frac{1}{n}\sum_{i=1}^{n} T_i(x) \tag{5}$$

where $T_i(x)$ is the prediction of the i-th decision tree, and n is the number of trees.

Gradient Boosting (GB): Gradient Boosting constructs models in series, with each model correcting the errors (or residual) of its predecessor. The prediction is finally the sum of the weighted predictions of each model [15]:

$$\hat{y}_{GB} = \sum_{m=1}^{M} \alpha_m h_m(x) \tag{6}$$

where $h_m(x)$ is the prediction of the m-th model, and α_m is the weight assigned to the m-th model.

AdaBoost: AdaBoost concentrates on the wrong predictions in each training round by reweighting the samples, allowing more attention to complex samples for the next round. The ultimate prediction is a weighted vote of the majority [15]:

$$\hat{y}_{AdaBoost} = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right) \tag{7}$$

where $h_m(x)$ is the prediction of the m-th model, α_m is the weight, and the sign function is used for classification.

XGBoost (XGB): XGBoost is a tuned model of Gradient Boosting with the regularization term in the objective function, which enhances both performance and prevents overfitting. The loss of the model, together with the regularization term, is minimized [15]:

$$\hat{y}_{XGB} = \sum_{m=1}^{M} f_m(x) \tag{8}$$

Here, $f_m(x)$ is the prediction of the m-th model and M is the number of models.

The objective function for XGBoost is the following:

$$L(\theta) = \sum_{i=1}^{N} \ell(y_i, \hat{y}_i) + \sum_{m=1}^{M} \Omega(f_m) \tag{9}$$

where $\ell(y_i, \hat{y}_i)$ is the loss function and $\Omega(f_m)$ is the regularization term:

$$\Omega(f_m) = \gamma T + \frac{1}{2}\lambda \|w_m\|^2 \tag{10}$$

where γ and λ are regularization parameters, T is the number of leaves, and w_m represents the leaf weights.

LightGBM (LGBM): LightGBM is a gradient boosting framework that uses tree-based learning algorithms and histogram-based algorithms that discretize continuous values in histograms, which enables us to lower memory and train faster. In common with XGBoost, the final prediction is a superposition of the outputs of multiple models [11]:

$$\hat{y}_{LGBM} = \sum_{m=1}^{M} f_m(x) \tag{11}$$

Like XGBoost, LightGBM combines predictions from M models, where $f_m(x)$ is the prediction from the m-th model.

3.5. Hyperparameter Tuning (RandomizedSearchCV)

We tuned the hyperparameters using RandomizedSearchCV to optimize the critical parameters for each model. We explored the search across a wide range of parameters, varying by model. 10-fold cross-validation (repeated stratified 3 times) was applied to ensure a robust evaluation. The models were evaluated according to accuracy, precision, recall, and F1-score, and the model with the highest accuracy was selected [16]. In Table 2, several parameters have been amended as follows:

4. Results and Discussion

4.1. Environment Setup

The training-testing ratio of 70-30 was used to achieve generalizability and reliability. The implementation was carried out on Visual Studio Code, with system configuration: processor — Intel Core i5 10th-gen and 12 GB RAM & Windows OS version 11 (64-bit). Model performance is

Table 2: Hyperparameters Used in Our Models

Hyperparameter	Description
n_estimators	Random integer [100, 500] (number of trees)
max_depth	[None, 5, 10, 15, 20] (maximum tree depth)
min_samples_split	Random integer [2, 10] (samples required to split a node)
min_samples_leaf	Random integer [1, 5] (samples required at a leaf node)
learning_rate	Random uniform [0.01, 0.3] (learning rate)
subsample	Random uniform [0.5, 1.0] (sample fraction)
colsample_bytree	Random uniform [0.5, 1.0] (feature fraction)

evaluated through various metrics, including accuracy, precision, recall, F1-score, a confusion matrix, and an ROC curve, which provides a more detailed view of positive and negative outcomes (true and false). Performance measures calculated in this study are

Accuracy :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

Precision:

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

Sensitivity :

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

F1-score :

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{15}$$

Overall accuracy measures a model's ability to make correct predictions. Precision informs about the proportion of true positives among the positives known a priori. Sensitivity or Recall—This helps to provide an estimate of how many positives exist and have been accurately identified. The F1-score takes into account both precision and recall, considering the impact of false positives and false negatives.

4.2. Result Analysis

The results of the combination of ensemble algorithms are shown in Figure 5. The best testing accuracy for XGBoost is 88.11%, while the training accuracy is 89.94%. Gradient Boosting ranked second, achieving a training accuracy of 92.78% and a testing accuracy of 88.07%. Random Forest achieves a training accuracy of 92.46% and a testing accuracy of 87.43%. LightGBM achieves training and test accuracies of 91.16% and 88.02%, respectively. With a bit less, AdaBoost achieves a training accuracy of 87.17% and a testing accuracy of 87.01%. These findings highlight the high generalizability of the models, where XGBoost records the highest testing accuracy. The five candidate algorithms were fitted using 30-fold cross-validation, resulting in 150 models trained on various subsets of the data.

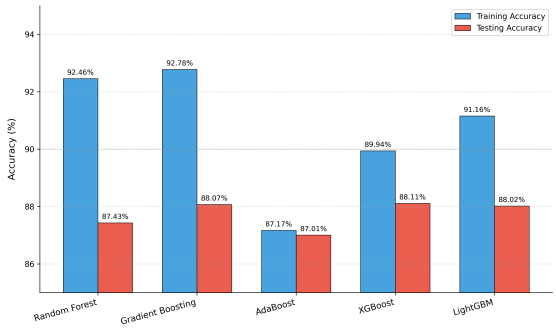

Figure 5: Train-Test Accuracy

The accuracy, precision, recall, and F1-score of five machine learning models (Random Forest, Gradient Boosting, AdaBoost, XGBoost, and LightGBM) are presented in Table 3 at the two-class level. The best accuracy and F1-score for Class 1 (88.11% and 77.41%, respectively) were achieved using XGBoost. Gradient Boosting almost matched its accuracy at 88.07% with a significantly higher Class 0 recall (92.26%). AdaBoost achieved high precision in Class 0 (93.56%) and recall in Class 1 (82.38%). For Class 0, Random Forest had the highest F1-score (91.40%). Overall, both XGBoost and Gradient Boosting performed remarkably well on most metrics, which indicates that they are better suited for different applications.

Table 3: Performance Metrics of Different Algorithms (in %)

Algorithms	Accuracy	Precision		Recall		F1-score	
		Class 0	Class 1	Class 0	Class 1	Class 0	Class 1
Random Forest	87.43	92.90	73.41	89.95	80.13	91.40	76.62
Gradient Boosting	88.07	91.73	77.24	92.26	75.96	92.00	76.60
AdaBoost	87.01	93.56	71.47	88.62	82.38	91.02	76.54
XGBoost	88.11	92.70	75.65	91.17	79.26	91.93	77.41
LightGBM	88.02	92.38	75.93	91.42	78.21	91.90	77.05

The bar plot in Figure 6 shows the Gini scores for Random Forest, Gradient Boosting, AdaBoost, XGBoost, and LightGBM. The Gini value, used as a measure of model

performance, reflects the degree of accuracy in the model predictions (higher values imply better predictive performance). The Gini of the Random Forest model is 0.86528, closely followed by Gradient Boosting, with a Gini value of 0.86672. AdaBoost gives a slightly better score of 0.87278, and XGBoost, with a score of 0.87385, is the best among the five models. Considering its performance, LightGBM has a Gini score of 0.86704, which falls within the upper-middle-rank category. Each model also features a unique color for each node, which enhances the visual separation of the nodes in the plot. For ease of representation, each bar has been labeled with its Gini score. The bars are thin to produce a clean and streamlined visual representation that highlights the small differences in model performance.

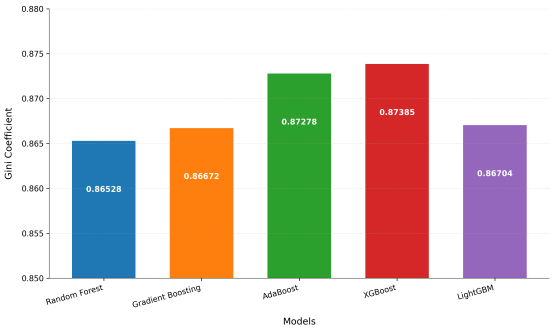

Figure 6: Comparison of Gini Coefficients across ML Models

Table 4 summarizes the performance of the XGBoost and Logistic Regression models. XGBoost outperforms Logistic Regression in several key metrics, notably its training accuracy of 89.94% compared to 81.67% for Logistic Regression. The ROC AUC score of XGBoost is 93.69%, much higher than 89.57% of Logistic Regression; the Gini score of XGBoost is 87.39%, while Logistic Regression is only 79.15%. The optimal XGBoost parameter values are colsample_bytree = 0.571, learning_rate = 0.205, and max_depth = 3; for Logistic Regression, a regularization parameter C = 3.755 with an 'l1' penalty. Regarding accuracy, XGBoost achieves 88.11%, while Logistic Regression achieves 81.02%. The XGBoost macro and weighted F1-scores are 84.67% and 88.20%, respectively, better than the likelihood regression scores of 77.48% and 81.82%.

Table 4: Performance Comparison (XGBoost vs. Logistic Regression)

Metric	XGBoost	Logistic Regression
Best Parameters	colsample_bytree: 0.571, learning_rate: 0.205, max_depth: 3	C: 3.755, penalty: 'l1'
Training Accuracy	89.94%	81.67%
Testing Accuracy	88.11%	81.02%
ROC AUC	93.69%	89.57%
Gini Score	87.39%	79.15%
Macro F1-score	84.67%	77.48%
Weighted F1-score	88.20%	81.82%

Figure 7 shows the confusion matrix of the XGBoost model. This means that if the actual value was Class 0 (the negative class), then the model correctly predicted 21,542 instances as Class 0 (true negatives), and 2,087 instances were incorrectly classified as Class 1 (false positives). From Class 1 (the positive class), 6,482 are Class 1 (true positives), whereas 1,696 instances are mistakenly considered as Class 0 (false negatives). The confusion matrix indicates that the model can better differentiate between the two classes in one than in the other (Class 0 vs. Class 1).

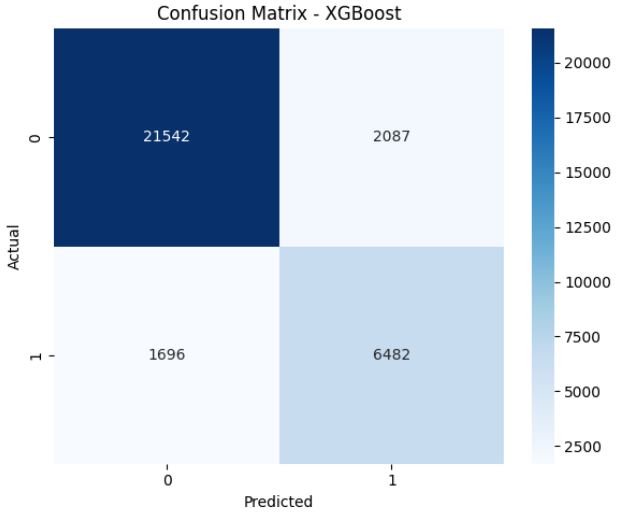

Figure 7: Confusion Matrix of XGBoost Model

The ROC curve of the XGBoost model is shown in Figure 8, and an excellent discriminative power is observed, with an AUC of 93.69%. This excellent AUC demonstrates the model's good performance in differentiating patients with cardiovascular disease from those without, suggesting that it is a valuable tool for clinical prediction.

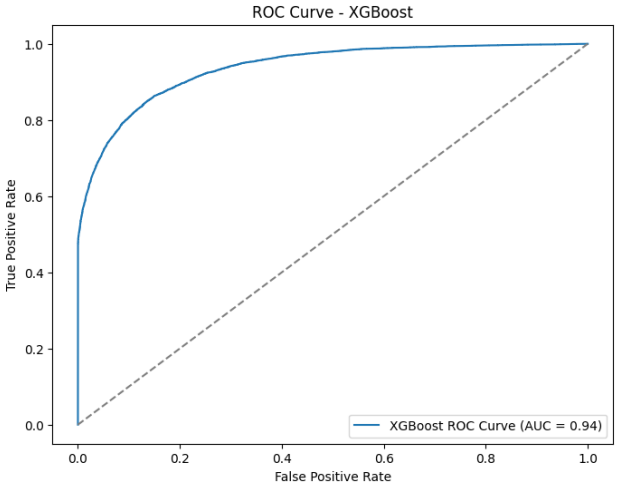

Figure 8: ROC Curve of XGBoost Model

Figure 9 presents the calibration curve for the XGBoost model, showing the plots of the predicted probabilities against the observed fraction of positives. The curve

is contrasted with a "Perfectly Calibrated" line, the line on which predictions would lie if calibration were perfect. The calibration of the XGBoost model suggests a moderate correlation between predicted probabilities and observed outcomes.

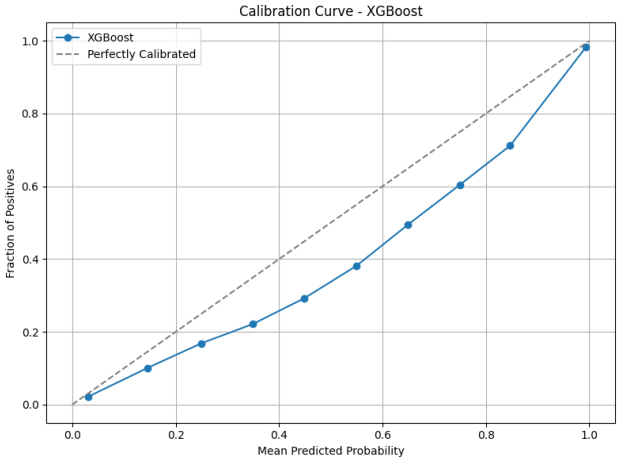

Figure 9: Calibration Curve of XGBoost Model

Figure 10 shows the decision curve analysis (DCA) for the XGBoost model compared to two reference strategies, "Treat None" and "Treat All". The blue curve representing the XGBoost model exhibits a net benefit of approximately zero at threshold probabilities near zero and becomes increasingly negative as the threshold probability approaches 1. It is clear that the net benefit of the model decreases at a threshold probability of about 0.7, and the model has reduced returns at higher threshold values. The Treat None strategy (black dashed line) maintains a net benefit of around -70, as the same baseline treatment is assumed in this setting. The Treat All strategy (red dashed line) starts with an estimated net benefit of approximately -70 and increases as more subjects are treated without a corresponding model prediction, suggesting that the model overlooks those who are treated.

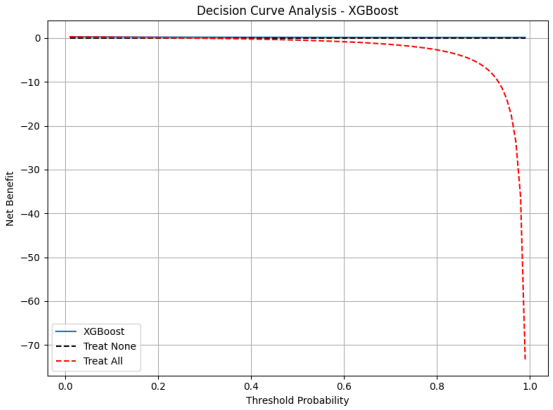

Figure 10: Decision Curve Analysis of XGBoost Model

Figure 11 illustrates the importance of the features of

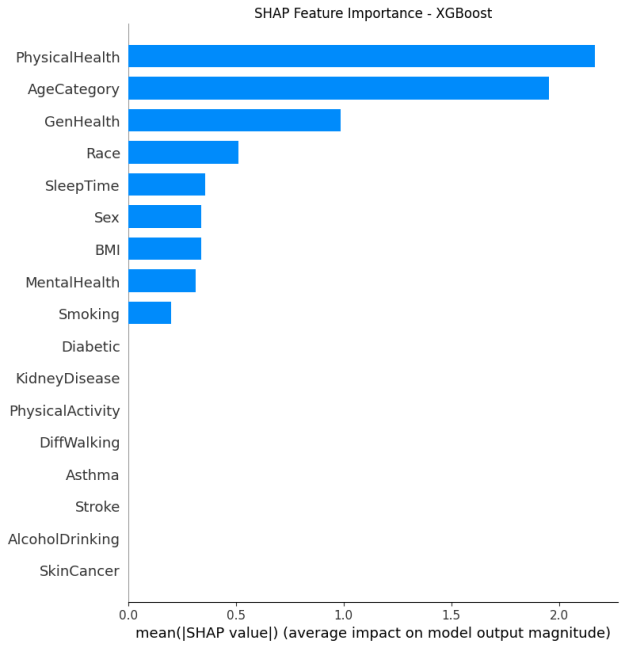

Figure 11: XGBoost Feature Importance via SHAP Values

an XGBoost model using SHAP values (SHapley Additive exPlanations). The bars represent the averaged absolute SHAP values, which indicate the average magnitude of the effects of each feature. The outcome shows that the most crucial feature is PhysicalHealth, followed by AgeCategory and GenHealth. These are the features with the most significant mean SHAP values, indicating that they provide the most explanation. Other essential variables include Race, SleepTime, and Sex. Meanwhile, SkinCancer, AlcoholDrinking, and Stroke have lower SHAP values, indicating that they have less influence on the predictions of the model.

4.3. Discussion

Table 5 compares the performance of the model with that of other existing studies, with an emphasis on the datasets used, the applied model, and the achieved accuracy. The experiments involve a variety of machine learning solutions applied to different datasets, although with a focus on the Kaggle one used here (and also in Chen, 2024 [11]). In the present investigation, multiple ensemble models are used, and the XGBoost model yields the best accuracy (88.11%), followed by Gradient Boosting (88.07%), LightGBM (88.02%), Random Forest (87.43%), and AdaBoost (87.01%). These results represent a significant improvement over those of Chen (2024) [11], whose highest reported accuracy was 76.9% using LightGBM. This demonstrates how the present study achieves improved prediction performance, particularly with ensemble modeling that outperforms simpler models. Compared to other studies, Waigi et al. (2020) [6] achieved a 72.77% accuracy in a Kaggle field using a Decision Tree model. Rani et al. (2021) [7] outperformed us on all datasets using different

models, achieving the highest accuracy of 86.60% with a Random Forest model in the UCI dataset. Miranda et al. (2021) [8], working with the UCI dataset, achieved an accuracy of 80% when training a Stochastic Gradient Descent (SGD) model. In Shorewala (2021) [9], an accuracy of 75.1% was achieved in a Kaggle dataset using the Stacking model. Chen (2024) [11] also works with the same Kaggle dataset as ours, although it yields inferior results, ranging from 76% for Decision Trees to 76.9% for LightGBM. This is in sharp contrast to the current study, as advanced ensemble methods produce a much higher accuracy than has ever been reported.

Table 5: Comparative Analysis of CVD Prediction (in %)

Study	**Dataset**	**Model**	**Accuracy**
Waigi et al. (2020) [6]	Kaggle	DT	72.77%
Rani et al. (2021) [7]	UCI	NB	83.55%
		SVM	84.46%
		LR	85.07%
		RF	86.60%
		AB	86.59%
Miranda et al. (2021) [8]	UCI	SGD	80%
Shorewala (2021) [9]	Kaggle	SM	75.1%
Chen (2024) [11]	Kaggle	DT	76%
		LGBM	76.9%
		RF	76.7%
		LR	76.8%
Current Study	**Kaggle** [10]	**RF**	**87.43%**
		GB	**88.07%**
		AB	**87.01%**
		XGB	**88.11%**
		LGBM	**88.02%**

5. Conclusions

This is a comparative analysis of cardiovascular disease prediction using ensembles of machine learning algorithms, including Random Forest, Gradient Boosting, AdaBoost, XGBoost, and LightGBM, with accuracies ranging from 87.01% to 88.11%. Advanced feature engineering combined with hyperparameter tuning reduced overfitting and boosted the score of RandomizedSearchCV. Of the models tested, the XGBoost model achieved the highest accuracy of 88.11%, with consistent precision, recall, and F1-score for the three measures. The effectiveness of ensemble models is well reflected in these results, with practical clinical applications in predicting cardiovascular disease.

Acknowledgments

Jagannath University Research Grant supported this research for the year 2024–2025 (UGC approved). It was carried out in the Emerging Data Science Lab (EDSL), Department of Computer Science and Engineering, Jagannath University, Dhaka-1100, Bangladesh.

References

[1] W. H. Organization, Cardiovascular diseases (cvds) (2021). URL https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[2] M. A. Naser, A. A. Majeed, M. Alsabah, T. R. Al-Shaikhli, K. M. Kaky, A review of machine learning's role in cardiovascular disease prediction: Recent advances and future challenges, Algorithms 17 (2). doi:10.3390/a17020078. URL https://www.mdpi.com/1999-4893/17/2/78

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

[4] B. Y. Kazangirler, E. Özkaynak, Conventional machine learning and ensemble learning techniques in cardiovascular disease prediction and analysis, Journal of Intelligent Systems: Theory and Applications 7 (2) (2024) 81–94.

[5] F. Mesquita, G. Marques, An explainable machine learning approach for automated medical decision support of heart disease, Data & Knowledge Engineering 153 (2024) 102339.

[6] D. Waigi, D. S. Choudhary, D. P. Fulzele, D. Mishra, et al., Predicting the risk of heart disease using advanced machine learning approach, Eur. J. Mol. Clin. Med 7 (7) (2020) 1638–1645.

[7] P. Rani, R. Kumar, N. M. S. Ahmed, A. Jain, A decision support system for heart disease prediction based upon machine learning, Journal of Reliable Intelligent Environments 7 (3) (2021) 263–275.

[8] E. Miranda, F. M. Bhatti, M. Aryuni, C. Bernando, Intelligent computational model for early heart disease prediction using logistic regression and stochastic gradient descent (a preliminary study), in: 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI), Vol. 1, IEEE, 2021, pp. 11–16.

[9] V. Shorewala, Early detection of coronary heart disease using ensemble techniques, Informatics in Medicine Unlocked 26 (2021) 100655.

[10] K. Pytlak, Personal key indicators of heart disease, https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease (2021).

[11] L. Chen, Heart disease prediction utilizing machine learning techniques, Transactions on Materials, Biotechnology and Life Sciences 3 (2024) 35–50. doi:10.62051/e054hq43. URL https://wepub.org/index.php/TMBLS/article/view/784

[12] E. Hokijuliandy, H. Napitupulu, Firdaniza, Application of svm and chi-square feature selection for sentiment analysis of indonesia's national health insurance mobile application, Mathematics 11 (17). doi:10.3390/math11173765. URL https://www.mdpi.com/2227-7390/11/17/3765

[13] B. P. C, How to detect outliers in machine learning (2022). URL https://www.freecodecamp.org/news/how-to-detect-outliers-in-machine-learning/

[14] fawwazmts, Z-score and modified z-score (2024).
URL https://medium.com/@fawwazmts/z-score-and-modified-z-score-f689296e4d3a

[15] GeeksforGeeks, Gradientboosting vs adaboost vs xgboost vs catboost vs lightgbm, accessed: 2025-05-17 (2025).
URL https://www.geeksforgeeks.org/gradientboosting-vs-adaboost-vs-xgboost-vs-catboost-vs-lightgbm/?ref=asr3

[16] T. scikit-learn developers, Randomizedsearchcv (2025).
URL https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html