Machine learning algorithms for predicting the risk of chronic kidney disease in type 1 diabetes patients: a retrospective longitudinal study

Abstract

This research introduces a novel framework designed to accurately predict heart disease while prioritizing stringent patient data privacy. A collaborative system involving multiple entities is established to facilitate efficient data collection, secure management, preprocessing, and analysis of medical data. To safeguard sensitive patient information, a robust encryption scheme is implemented to protect data throughout its lifecycle. A comprehensive ensemble of machine learning algorithms, including Naïve Bayes, Decision Trees, Random Forests, K-Nearest Neighbors, Support Vector Machines, and Long Short-Term Memory networks, is rigorously applied to the preprocessed data. To optimize predictive performance, dimensionality reduction techniques such as Principal Component Analysis and anomaly detection using Isolation Forest are incorporated. Rigorous hyperparameter tuning is performed to fine-tune the models and achieve optimal results. Empirical evaluation demonstrates the superior performance of the proposed framework, with the Random Forest algorithm achieving the highest mean F1-score of 91.02%. This robust performance is attributed to the effective combination of feature engineering, model selection, and hyperparameter optimization. By prioritizing data privacy and leveraging advanced machine learning techniques, this research offers a significant contribution to the field of heart disease prediction. The proposed framework provides a secure and effective solution for improving heart disease diagnosis and management.

Keywords Heart disease prediction · Data privacy · Machine learning · Principal Component Analysis · Isolation Forest · Long short-term memory

# Introduction

Heart disease, synonymous with cardiovascular disease (CVD), remains a leading global health challenge, claiming millions of lives annually [1]. The World Heart Federation underscores this grim reality, reporting that one in three deaths is attributed to CVD [1]. Projections by the World Health Organization (WHO) are equally alarming, predicting over 23.6 million CVD-related deaths by 2030, primarily from strokes and heart failure [2]. The United States also grapples with a significant heart disease burden, accounting for substantial healthcare expenditures and lost productivity [3].

A complex interplay of factors, including stress, alcohol, smoking, poor diet, sedentary lifestyle, and comorbidities like hypertension and diabetes, contribute to heart disease development [2]. Early detection is key, as most CVD cases are treatable in their initial stages [4]. Timely and accurate diagnosis is crucial for effective treatment and improved patient outcomes, particularly in regions with limited access to modern technology and medical expertise [5, 6].

Existing diagnostic methods often fall short in terms of early detection due to limitations in accuracy, efficiency, and interpretability. Traditional models may struggle to capture complex interactions between features and temporal dependencies within the data, leading to suboptimal predictive performance. Additionally, concerns around data privacy hinder the development of robust models.

To address these challenges, this study proposes a novel hybrid approach that combines the strengths of ensemble methods, dimensionality reduction, anomaly detection, and deep learning. By integrating RF, PCA, iForest, and LSTM, we aim to create a more robust and accurate predictive model. The LSTM component is particularly crucial for capturing potential temporal patterns in patient data, which traditional ML models might overlook. Furthermore, our emphasis on data security, including encryption and differential privacy, ensures patient confidentiality while enabling effective model development.

This hybrid approach distinguishes itself from existing methods by simultaneously considering multiple feature perspectives (PCA and iForest), capturing temporal dynamics (LSTM), and leveraging the ensemble power of RF. This comprehensive approach is expected to enhance predictive accuracy and provide valuable insights into the underlying patterns of heart disease.

By developing a more accurate and efficient heart disease prediction model, this research aims to improve patient outcomes by enabling earlier detection and intervention. This could lead to reduced morbidity and mortality rates associated with heart disease, as well as decreased healthcare costs. Additionally, the proposed model's emphasis on data security and privacy aligns with emerging healthcare regulations and patient expectations, fostering trust in healthcare systems. Ultimately, the successful implementation of this model has the potential to significantly impact public health by reducing the global burden of heart disease.

The paper systematically evaluates the performance of various ML models and their combinations using a comprehensive set of metrics, including accuracy, precision, recall, F1-score, and AUC. The impact of different training set sizes on model performance is also investigated to assess the robustness of the proposed approach.

To abbreviate terms for better readability, a list of abbreviations used in this paper is provided in Table 1:

Table 1 Table of abbreviation

|  |  |
| --- | --- |
| CVD | Cardiovascular Disease |
| ML | Machine Learning |
| LR | Logistic Regression |
| NB | Naïve Bayes Classifier |
| DT | Decision Tree |
| RF | Random Forest Classifier |
| KNN | K-Nearest Neighbors |
| SVM | Support Vector Machines |
| XGB | Extreme Gradient Boosting |
| SGD | Stochastic Gradient Descent |
| QMBC | Quine McCluskey Binary Classifier |
| ANN | Artificial Neural Networks |
| DNN | Deep Neural Networks |
| RNN | Recurrent Neural Network |
| LSTM | Long Short-Term Memory |
| PCA | Principal Component Analysis |
| iForest | Isolation Forest |
| AUC | Area Under the Curve |
| MRMR | Minimum Redundancy and Maximum Relevance |
| LASSO | Least Absolute Shrinkage and Selection Operator |
| SMOTE | Synthetic Minority Oversampling Technique |
| GridSearchCV | Grid Search Cross-Validation |
| ANOVA | Analysis of Variance |
| PPV | Positive Predictive Value |
| NPV | Negative Predictive Value |

# Literature Review

Heart disease prediction has been a focal point of numerous studies, with researchers exploring various ML techniques and feature selection strategies to enhance diagnostic accuracy. One investigation explored the optimization of feature selection and classifier performance in heart disease diagnosis [7]. The study compared established techniques (Relief, MRMR, LASSO, LLBFS) with a novel FCMIM algorithm, evaluating their influence on various classifiers, including LR, KNN, ANN, SVM, NB, and DT. The findings revealed that the combination of FCMIM and SVM achieved the highest accuracy of 92.37% using Leave-One-Out Cross-Validation. While DNN were explored, limited data hindered their performance.

Another study introduced an HRFLM model combining RF and Linear Model for improved heart disease classification [8]. Experimental results showcased significantly higher accuracy compared to traditional methods, underscoring the effectiveness of this approach.

Another investigation aimed to achieve accurate prediction not only for the presence or absence of CVD but also for its severity levels [9]. This research employed ML models like SVM, KNN, LR, SGD, and tree-based ensembles to address the challenge of imbalanced data using SMOTE and hyperparameter optimization with HPO. Notably, tree-based ensemble models outperformed the others, achieving impressive accuracies of 99.2% and 98.52% for CVD presence/absence and 95.73% for severity level prediction.

Further research compared six models, including XGB, Bagging, RF, DT, KNN, and NB, on a dataset of over 300,000 cases [10]. XGB emerged as the top performer with an accuracy of 91.30% and AUC of 0.83. Feature selection using sequential backward selection and KNN was also investigated, achieving a classification accuracy of 90% with six selected features [11].

Subsequent studies compared LR, KNN, SVM, and XGB classifiers, optimized with GridSearchCV [12, 13, 14]. RF consistently demonstrated superior performance, with accuracies reaching 99% and 98.53%. A more recent study introduced ensemble methods and a novel QMBC classifier, along with feature selection techniques like Chi-Square and ANOVA, further improving prediction accuracy [15].

While these studies provide valuable insights, this research distinguishes itself by introducing a novel framework prioritizing data security and privacy, encompassing a complete system lifecycle, and employing rigorous evaluation methods.

# Preliminaries

## Naïve Bayes Classifier

Naïve Bayes is a simple probabilistic classifier that assumes feature independence given the class label [16]. Despite this assumption, it often performs well due to its focus on finding the class with the highest probability. Recent research suggests that Naïve Bayes can be optimal even when feature independence is violated, as long as dependencies balance out across classes [16].

## Decision Tree (DT)

A DT is a model used for classification and regression tasks that makes decisions by splitting data into subsets based on feature values. Each internal node represents a decision point, where a specific feature is selected to split the data. The terminal leaves represent the final decision or output. The tree structure allows for easy interpretation and visualization, but it can become complex with very large datasets [17].

## Random Forest Classifier (RF)

RF is an ensemble method that builds multiple DTs, each trained on a random subset of features and samples. During classification, each tree votes for a class, and the majority vote determines the final output. This approach leverages bagging, where each tree is trained on a bootstrap sample, and uses the Gini Index to evaluate the impurity of nodes [18]. By aggregating the predictions from many trees, RF improves accuracy and reduces overfitting compared to single DTs [18].

## K-Nearest Neighbors (KNN)

The k-Nearest Neighbors (k-NN) algorithm is a simple yet effective supervised learning technique for both classification and regression tasks. Given a query point, k-NN finds the k closest data points in the training set and assigns the query point to the most frequent class among those neighbors (for classification) or the mean of the neighbor values (for regression) [19].

## Support Vector Machine (SVM)

Support Vector Machines (SVMs) are a robust supervised learning technique for classification tasks. They work by identifying the optimal hyperplane that maximizes the margin between different classes. Data points nearest to the hyperplane, known as support vectors, significantly influence its placement. To handle complex, non-linearly separable datasets, SVMs employ the kernel trick. This technique maps the data into a higher-dimensional space, where it becomes linearly separable [20]. SVMs commonly employ kernel functions such as linear, polynomial, RBF, and sigmoid.

## Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique that identifies patterns in data and transforms it into a new coordinate system. Key steps include standardization, covariance matrix calculation, eigenvalue decomposition, principal component selection, and data projection. PCA is used for various purposes like dimensionality reduction, feature extraction, noise reduction, visualization, and outlier detection. The Nonlinear Iterative Partial Least Squares (NIPALS) algorithm is an efficient iterative method for calculating principal components [21].

## Isolation Forest (iForest)

Isolation Forest (iForest) is an unsupervised anomaly detection method that isolates anomalies using a forest of isolation trees. Anomalies are identified based on their shorter path lengths to isolation compared to normal instances. iForest offers advantages like linear time complexity, handling high-dimensional data, and robustness to noise and outliers. It has shown superior performance in accuracy and efficiency compared to other methods [22].

## Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to address the vanishing gradient problem, which hinders traditional RNNs from learning long-term dependencies in sequential data. LSTMs achieve this by employing a unique cell structure with input, output, and forget gates. These gates regulate the flow of information into and out of the cell, enabling the network to store and retrieve information over extended time periods [23]. LSTMs are effective for tasks like natural language processing and speech recognition, where understanding context is crucial.

## Train-Test Split

The train-test split is a method for evaluating ML models by dividing data into a training set (to fit the model) and a test set (to assess performance). Common split ratios are 70:30 or 80:20. Choosing the right ratio is essential, as an imbalanced split can skew performance metrics by underutilizing data or providing insufficient test data [24].

Key considerations for train-test split:

*Randomization:* It's crucial to ensure that both the training and test sets accurately represent the overall data distribution.

*Stratification:* For imbalanced datasets, stratified sampling can be used to maintain the same class proportions in both training and testing sets.

*Multiple splits:* To obtain more robust performance estimates, cross-validation can be employed, involving multiple train-test splits and averaging the results.

# System Model and Functional Roles

This section presents a detailed architectural blueprint of the system, as visualized in Figure 2. It formally defines the system's constituent components, their collaborative relationships, and the functions they perform. Additionally, the section elaborates on the sequential flow of activities within the system.

## Roles of each entity

The proposed system comprises five primary entities: Client Company (CC), System Developer (SD), Service Provider (SP), System User (SU), and System (SYS) as illustrated in Figure 1.

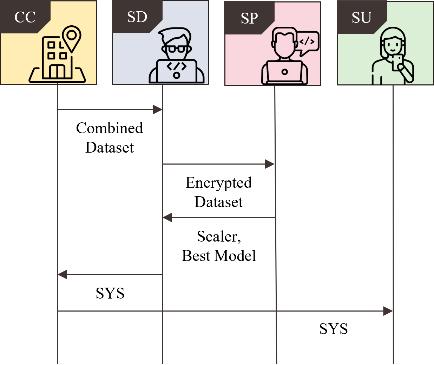


Fig. 1 Interactions between entities

*CC*The CC collects datasets related to heart disease prediction from various hospitals and combines them. Since medical data is confidential, CC hands it over to SD for securing. After SD returns the SYS to CC, CC releases the SYS for use by SUs.

*SD* The SD generates public keys and encrypts the dataset using these keys to ensure its security. The encrypted dataset is then handed over to SP for model training. After SP provides the best ML model to SD, SD tests this model with actual data to verify its accuracy. If the model performs well, SD develops the SYS, which processes actual data to determine if the SU has heart disease, and hands it over to CC.

*SP* The SP receives the encrypted dataset from SD, preprocesses it, and trains several models. SP compares the F1-scores of different models and selects the one with the highest score. The trained model with the highest F1-scores and the scaler for preprocessing are then handed over to SD.

*SU* The SU utilizes the SYS to benefit from the heart disease prediction model.

*SYS* The SYS, developed by SD, is designed to receive data from the SU. Using public keys, SYS encrypts this data, preprocesses it with the scaler, and predicts the outcome using the model provided by SP.

|  |
| --- |
|  |
| Fig. 2 Proposed Heart Disease Prediction System |

## Roles of each entity

Table 2 Notation Used

|  |  |
| --- | --- |
| *Symbol* | *Description* |
| *Δi​* | Set for a single raw dataset for heart disease prediction |
| *d* | No. of features |
| *ξi​ ∈ Rd* | Feature vector for the *i*-th data point |
| *ωi​ ∈ Ω* | Value of the *i*-th feature |
| *Ω* | Set containing all possible value types |
| *∅* | Missing value |
| *Φ* | Subset of features crucial |
| *ψ(ωi​)* | Transformation function for categorical encoding consistency |
| *E(.)* | ElGamal encryption function |
| *κi* | ​Unique public key for feature *i (fixed\_keyi, qi, hi, gi)* |
| *Ci* | Encrypted version of *ξi* |
| *cij* | Encrypted value of *j*-th element in *ξi* |
| *λ* | Label value (Categorial) |
| *λ'* | Encoded label value |
| *Θ* | *Θ = {C1, C2, ..., C11, λ'}* |
| *Zi* | Standardized version of *Ci* |
| *M* | ML models |
| *{M1, ..., M5} ∈ M* | {NB, DT, RF, KNN, SVC} |
| *Mα ∈ M* | Top performing model 1 |
| *Mβ ∈ M* | Top performing model 2 |
| *Mγ ∈ M* | Best performing model |
| *Π* | Set of performance metrics |
| *{Π1, …, Π8} ∈ Π* | {Confusion Matrix, Accuracy, Specificity, Recall, PPV, NPV, F1-score, AUC} |
| *τ* | Ratio of training data size (0.1 to 0.9) |
| *S* | Scaler used for standardization |

## Individual stages

The proposed framework consists of eight distinct stages. These are: (1) Data Collection, (2) Data Encryption, (3) Data Preprocessing, (4) Model Training, (5) Model Optimization, (6) Final Model selection, (7) Model verification and (8) System Development. These stages are detailed below, with notations referenced from Table 2:

4.3.1 Data Collection

The Client Company (CC) is designed to collect datasets from various hospitals to support heart disease prediction. However, for this research, data was sourced from seven publicly accessible Kaggle repositories [24-30]. The collected datasets, denoted as Δ1​, Δ2​, ..., Δ7​, exhibit some inconsistencies. Some Δi*​* encompass irrelevant features beyond those necessary for prediction. Feature values are numerical and categorical and some are encoded differently across Δi*​*.

To ensure uniformity and facilitate data processing, CC undertakes some pre-processing steps. Only Φ ⊂ {ξi​} are retained. Irrelevant features are discarded. Encoded values are transformed using ψ(ωi​) to ensure consistent representation within the set Ω. For instance, ψ converts all instances of a specific numerical encoding for a category into a designated categorical value. The pre-processed Δi are merged into a single, unified dataset Δ = ​, where Δi′​ represents the pre-processed version of Δi. To maintain data confidentiality, Δ is then securely transferred to the SD. Δ has following attributes as detailed in Table 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| **(a)** | **(b)** | **(c)** | **(d)** | **(e)** | **(f)** |
|  |  |  |  |  |  |
| **(g)** | **(h)** | **(i)** | **(j)** | **(k)** |  |
| Fig. 3 Distribution of Heart Disease Prediction Attributes | | | | | |

Table 3 Heart Disease Prediction Attributes

|  |  |  |
| --- | --- | --- |
| *Attribute* | *Unit* | *Type of Data* |
| Age | Numeric | In Years |
| Sex | Nominal | 1. Male 2. Female |
| Chest Pain Type | Nominal | 1. Typical Angina 2. Atypical Angina 3. Non-anginal Pain 4. Asymptomatic |
| Resting Blood Pressure | Numeric | 94-200 (mm HG) |
| Cholesterol | Numeric | 126-564 (mg/dl) |
| Fasting Blood Sugar | Binary | Yes / No > 120 mg/dl |
| Resting ECG | Nominal | 1. Normal 2. Abnormal ST-T Wave 3. Left Ventricular Hypertrophy |
| Maximum Heart Rate | Numeric | 71-202 |
| Exercise Angina | Binary | Yes / No |
| Oldpeak | Numeric | 0 - 6.2 |
| ST Slope | Nominal | 1. Upsloping 2. Flat 3. Downsloping |

Figure 3 provides a visual representation of the distribution of key attributes within the dataset.

4.3.2 Data Encryption

SD utilizes the E(.) to safeguard Δ. Δ encompasses 11 ξ along with a λ. To ensure security, SD generates a κi = (fixed\_keyi, qi, hi, gi), for each individual feature i (i = 1, 2, ..., 11). This unique κ assignment is crucial because employing a single, identical κ for all ξ would lead to predictable data. With identical κi, encryption would result in the same cij for identical ωi across various ξ, compromising data security.

E(.) transforms each ωi into cij using the corresponding κi and E(.).These values are moded by a large prime number to ensure they remain integers suitable for ML algorithms. For λ, SD uses a simple binary encoding scheme λ'. λ' along with Ci are transmitted to the SP for training.

|  |
| --- |
| Algorithm 1Key Generation and Data Encryption |
| 1. **function** *Generate\_Key( ):* 2. *q ← random\_integer(1020, 1050)* 3. *g ← random\_integer(2, q-1)* 4. *fixed\_key* ← *random\_integer(1020, q)* 5. *h ← power(g, fixed\_key)* mod *q* 6. *κ* ← (*q, h, g, fixed\_key*) 7. **return** *κ* 8. **end function** 9. **function** *E(v, κ):* 10. *κ ← q, h, g, fixed\_key* 11. *s ← power(h, fixed\_key)* mod *q* 12. *p ← power(g, fixed\_key)* mod *q* 13. **return** *s \* v, p* 14. **end function** 15. **function** *Encrypt\_Data(Δ, κ):* 16. **for** *i* **from** 0 **to** *d* **do** 17. *κi ← Generate\_Key( )* 18. **end for** 19. **for** *i* **from** 0 **to** *|Δ|* **do** 20. **for** *j* **from** 0 **to** *d* **do** 21. *Cij ← E(ωij, κj)* 22. *Ci.append(Cij)* 23. **end for** 24. *Θ.append(Ci)* 25. **end for** 26. **return** *Θ* 27. **end function** |

4.3.3 Data Preprocessing

SP handles missing values (∅) within the feature set (Θ) by imputing the mean value for each feature in Θ. Subsequently, Ci undergo standardization using the StandardScaler technique to get Zi.

*Standardization*, , where,

*Mean*, , and,

*Standard Deviation,*

4.3.4 Model Training

SP partitions Θ into Θtrain and Θtest. Subsequently, SP evaluates the performance of a selection of Μ = {M1, M2, ..., M5}. These models are selected due to their ability to handle categorical data and their potential to function without requiring decrypted features. For each model Mi, SP calculates Π = {Π1, Π2, ..., Π8}. Furthermore, SP performs hyperparameter optimization, specifically focusing on the "random\_state" parameter, to maximize Π7 for each model Mi. Finally, based on the achieved Π7, SP selects the two top-performing models Mα and Mβ for further evaluation and potential hyperparameter tuning.

4.3.5 Model Optimization

SP conducts a more in-depth evaluation on Mα and Mβ. These models possess the optimal hyperparameters that yielded the highest Π7. SP employs three additional techniques for analysis: PCA, IF, and LSTM.

Additionally, SP investigates the impact of τ on model performance. For both Mα and Mβ, SP calculates Π7 across various τ. These sizes range from 10% (τ = 0.1) to 90% (τ = 0.9) of Θ.

4.3.6 Final Model Selection

SP conducts a comprehensive evaluation on eight different model configurations: Mα, Mα incorporating PCA, Mα incorporating IF, Mα incorporating LSTM, Mβ, Mβ incorporating PCA, Mβ incorporating IF, and Mβ incorporating LSTM. For each configuration, SP calculates the average Π7 (Π7’) across various τ, ranging from 10% (τ = 0.1) to 90% (τ = 0.9) of Θ. Finally, SD selects Mγ that achieves the highest Π7’. Mγ along with S is finalized and delivered back to the SD.

4.3.7 Model Verification

SD performs a validation step to verify the effectiveness of Mγ. SD identifies a subset of rows (C1, C2, …, Ck) from Θ. Here, ∅ ∉ Cij for all i from 1 to k and j from 1 to d. For each Ci, SD utilizes S to get Zi, and Mγ to predict the corresponding λi'.

Based on (C1, C2, …, Ck) from Θ, the SD identifies the corresponding rows (ξ1, ξ2, ..., ξk) from Δ. For each instance ξi, the SD encrypts ξi using the corresponding encryption key κi to generate the encrypted feature vector Ci. Subsequently, SD utilizes S to get Zi, and Mγ to predict λi' for these Zi.

SD compares λi' obtained from both the Θ and Δ. If λi' is the same for both Θ and Δ, it signifies that Mγ and S function correctly while maintaining data security. In this case, SD proceeds with system development using Mγ and S. Conversely, if any inconsistencies arise in λi', it suggests potential issues with Mγ or S. SD then returns Θ to the SP for further investigation.

|  |
| --- |
| Algorithm 2Model Verification |
| 1. **function** *ModelVerification(Δ, Θ, Mγ, S, κ)* 2. *cnt* ← 0 3. **for** *i* **from** 0 **to** *|Δ|* **do** 4. *flag* ← **true** 5. **for** *j* **from** 0 **to** *d* **do** 6. **if** *missing(ωij)* **then** 7. *flag* ← **false** 8. **break** 9. **end if** 10. **end for** 11. **if** *flag* **then** 12. **for** *j* **from** 0 **to** *d* **do** 13. *κj ← κ[j]* 14. **if** *is\_string(ωij)* **then** 15. *num\_Ωij* ← **sum**(**ord**(*char*) **for** *char* **in** *ωij*) 16. **else** 17. *num\_ωij ← ωij* 18. **end if** 19. *cij, \_ ← E(κj, num\_ωij)* mod 999999999 20. **end for** 21. *zi ← S.transform(ci)* 22. *λi\_predγ ← Mγ.predict(zi)[0]* 23. *λi\_Θ ← Θi[d + 1]* 24. **if** *λi\_predγ ≠ λi\_Θ* **then** 25. *cnt* ←*cnt* + 1 26. **end if** 27. **end if** 28. **if** *cnt =* 0 **then** 29. **return** "Model is verified." 30. **end if** 31. **end function** |

4.3.8 System Development

SD delivers a user-friendly interface for SU interaction. This interface features input fields corresponding to each of the d features (ξi) within the model, where SU enters their data directly into these fields. The system then encrypts ξi using the corresponding κi to generate Ci. This Ci undergoes processing through S for standardization to get Zi. Subsequently, Mγ leverages Zi to predict λi'. Once complete, the SD delivers the finalized system to CC for broadcasting, making it readily available for SU.

# Security Threats and Solutions

In our proposed heart system, several potential security threats were identified, and corresponding measures were implemented to address these threats. This section outlines the primary threats and the solutions incorporated in our methodology to ensure data integrity and system security.

## Data Confidentiality Breach

*Threat:* Unauthorized access to sensitive medical data could occur during transmission or storage, leading to potential data leaks.

*Solution:* To address this, SD uses the ElGamal encryption function E(.) to encrypt the dataset with unique keys for each feature. This ensures that the data remains confidential even if intercepted.

## Predictable Encryption

*Threat:* Using a single encryption key for all features could create predictable encryption patterns, making the data vulnerable to attacks.

*Solution:* To mitigate this risk, SD generate unique encryption keys for each feature. This prevents identical ciphertexts for identical values, thereby enhancing encryption security.

## Integrity of Model Training Data

*Threat:* Tampering with the training data could result in inaccurate model predictions or biases.

*Solution:* The system incorporates a validation step where the SD verifies the model’s integrity by comparing predictions from encrypted and original datasets. Any inconsistencies prompt further investigation to ensure the model's accuracy and reliability.

## Replay Attacks

*Threat:* Captured encrypted data might be replayed to the system, potentially leading to unauthorized access or manipulation.

*Solution:* To prevent replay attacks, unique encryption keys are generated for each feature, and data is modded by a large prime number. This ensures that each dataset is unique, even if the underlying data is unchanged.

## Replay Attacks

*Threat:* Sensitive information might be inferred through model queries.

*Solution:* To mitigate this risk, models are selected based on their ability to function without requiring decrypted features. This ensures that sensitive data remains encrypted throughout the inference process.

# Results

In our analysis, we evaluated the performance of ML models using several key metrics. To comprehensively assess model accuracy, we utilized a confusion matrix, which provides a tabular representation of the model's predictions against the actual ground truth values.

A confusion matrix, as detailed in [15], is a table that summarizes the performance of a classification model by comparing its predicted values with actual values. It categorizes predictions into four groups: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

We calculated the following performance metrics based on the confusion matrix:

*Accuracy* represents the proportion of correct predictions (both positive and negative) out of the total number of instances.

*Accuracy =*

*Recall (Sensitivity)* refers to the percentage of true positive instances that are correctly identified by the model.

*Recall =*

*Specificity* measures the proportion of true negatives correctly identified as such out of all actual negative cases.

*Specificity =*

*Precision or Positive Predictive Value (PPV)* measures the proportion of correctly predicted positive cases out of all instances predicted as positive.

*Precision or PPV =*

*Negative Predictive Value (NPV)* measures the proportion of actual negative cases that are correctly predicted as negative out of all predicted negative cases.

*NPV =*

*F1-Score* combines precision and recall into a single metric by calculating their harmonic mean, balancing both aspects of model performance.

*F1-Score =*

*Area Under the Curve (AUC)* represents the overall performance of a classification model across all possible classification thresholds. A higher AUC value indicates better model discriminative power.

Training ML models on encrypted data is challenging due to the loss of interpretability. Traditional models rely on specific features, which are obscured by encryption. However, the consistent nature of encryption can be exploited to identify relationships within the data.

While challenging, some ML algorithms like NB, DT, RF, SVM, and KNN can be adapted to encrypted data. NB leverages feature independence, DT and RF utilize relative feature ordering, SVM compares relative features, and KNN exploits distance relationships between data points.

To assess model robustness and generalization, we employed a range of train-test splits, varying the ratio of training to testing data from 1:9 to 9:1. Models were trained on training data and evaluated on corresponding test data using metrics like accuracy, precision, recall, F1-score, and AUC-ROC. This helped identify overfitting or underfitting issues and ensured model reliability and generalization.

|  |
| --- |
|  |
| Fig. 4 Performance Metrics of Classification Algorithms with Varying Train-Test Split |

As the training set size increases, the performance of most models improves, as seen in the upward trends in the graphs. However, some models might be more sensitive to changes in training set size than others.

Overfitting can be observed when a model performs exceptionally well on the training data but poorly on the test data, especially with larger training sets. Underfitting occurs when a model consistently performs poorly, regardless of the training set size.

Figure 4 illustrates how the performance of different machine learning models on test data varies with changes in train-test split. The evaluated models are NB, DT, RF, KNN, and SVM.

RF consistently outperforms other algorithms across all evaluation metrics. It achieves excellent performance in accuracy (average 89.98%), specificity (average 89.31%), recall (average 90.54%), PPV (average 91.55%), NPV (87.98%), F1-score (average 91.04%), and AUC (average 94.63%). These results indicate that RF is highly effective in classifying the dataset.

SVM demonstrates strong performance, particularly in accuracy (79.24%), specificity (71.91%), recall (84.88%), PPV (79.67%), NPV (78.61%), and F1-score (82.18%). While not surpassing RF, SVM provides robust classification results.

DT exhibits acceptable performance across most metrics, with accuracy (85.94%), specificity (85.17%), recall (86.57%), PPV (87.41%), NPV (84.34%), F1-score (86.96%), and AUC (85.87%) falling within the good performance range.

NB and KNN demonstrate similar performance patterns, with strengths in accuracy, specificity, and recall but weaknesses in predictive values. Both models show room for improvement, especially in terms of positive and negative predictive values.

Increasing training set size generally improves performance for all algorithms, but the rate of improvement diminishes after a certain point. RF and SVM exhibit more consistent performance across different training set sizes compared to other algorithms.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 4 Performance Comparison of Classification Models with Varying Training Set Sizes and Hyperparameters (F1-Score in %) | | | | | | | | | | | | | | |
| *Classifiers* | *Hyper-parameters* | | | *Train-Test Ratio* | | | | | | | | | *Mean* | |
| *1:9* | *2:8* | *3:7* | *4:6* | *5:5* | *6:4* | *7:3* | *8:2* | *9:1* |
| *DT* | Data Splitting (train\_test\_split)   * random\_state: 4749 | DecisionTreeClassifier   * random\_state: 436 |  | 76.06 | 80.89 | 82.93 | 84.92 | 88.31 | 89.81 | 91.47 | 94.13 | 96.57 | 87.23 |
| *DT PCA* | Data Splitting (train\_test\_split)   * random\_state: 1418 | PCA   * n\_components: 9 | DecisionTreeClassifier   * random\_state: 2066 | 72.65 | 75.21 | 78.01 | 81.20 | 83.03 | 85.30 | 87.43 | 87.94 | 91.58 | 82.48 |
| *DT iForest* | Data Splitting (train\_test\_split)   * random\_state: 2009 | IsolationForest:   * n\_estimators: 173 * contamination: 0.08 * random\_state: 2009 * max\_samples: 1 | DecisionTreeClassifier   * random\_state: 2009 * criterion: gini | 78.57 | 80.05 | 80.90 | 85.46 | 88.60 | 89.69 | 90.42 | 92.27 | 93.13 | 86.56 |
| *DT LSTM* | Data Splitting (train\_test\_split)   * random\_state: 274 | Optimizer (Adam)   * learning\_rate: ~0.1429 (1/7) | Model Training (model.fit)   * epochs: 47 * batch\_size: 10000 * verbose: 0 | 74.04 | 77.88 | 79.50 | 81.76 | 84.25 | 84.65 | 86.92 | 87.14 | 85.90 | 82.45 |
| Sequential  Dropout   * rate: 0.04   Dense   * units: 1 * activation: sigmoid | Model Compilation (model.compile)   * loss: binary\_crossentropy | DecisionTreeClassifier   * random\_state: 274 |
| *RF* | Data Splitting (train\_test\_split)   * random\_state: 1694 * n\_estimators: 95 | RandomForestClassifier   * random\_state: 1694 |  | 82.69 | 85.31 | 87.81 | 90.71 | 92.51 | 93.75 | 94.42 | 95.24 | 95.82 | 90.92 |
| *RF PCA* | Data Splitting (train\_test\_split)   * random\_state: 983 | PCA   * n\_components: 9 | RandomForestClassifier   * random\_state: 983 * n\_estimators: 91 | 79.24 | 82.58 | 85.18 | 87.06 | 89.03 | 90.86 | 91.56 | 93.01 | 92.75 | 87.92 |
| *RF iForest* | Data Splitting (train\_test\_split)   * random\_state: 1341 | IsolationForest:   * n\_estimators: 26 * contamination: 0.01 * random\_state: 530 * max\_samples: 105 | RandomForestClassifier   * random\_state: 530 * n\_estimators: 63 * criterion: gini | 83.22 | 85.00 | 88.08 | 90.54 | 92.55 | 93.62 | 94.38 | 95.84 | 95.96 | 91.02 |
| *RF LSTM* | Data Splitting (train\_test\_split)   * random\_state: 47 | Optimizer (Adam)   * learning\_rate: ~0.0667 (1/15) | Model Training (model.fit)   * epochs: 107 * batch\_size: 10000 * verbose: 0 | 74.30 | 79.53 | 79.56 | 81.98 | 83.95 | 85.91 | 87.50 | 87.96 | 89.91 | 83.40 |
| Sequential  Dropout   * rate: 0.05   Dense   * units: 1 * activation: sigmoid | Model Compilation (model.compile)   * loss: binary\_crossentropy | RandomForestClassifier   * random\_state: 47 * n\_estimators: 47 |

While all these metrics provide valuable insights, we prioritize the F1-score for comparing our models. This is because the F1-score effectively balances precision and recall, which are both crucial in our context. Precision ensures that positive predictions are accurate, while recall guarantees that a high proportion of actual positive cases are correctly identified. By optimizing the F1-score, we aim to achieve a model that effectively minimizes both false positives and false negatives.

Although accuracy might seem appealing, it can be misleading in imbalanced datasets, where the majority class dominates the evaluation. PPV focuses on the accuracy of positive predictions, but it might not capture the ability of the model to identify all positive cases. NPV is valuable for assessing the reliability of negative predictions, but it's less critical in our scenario. AUC provides a global measure of model performance, but it doesn't directly address the trade-off between precision and recall. Therefore, the F1-score emerges as the most suitable metric for our comparative analysis.

Table 4 presents a comparative analysis of various classification algorithms, including DT and RF, with and without dimensionality reduction (PCA) and anomaly detection (Isolation Forest), as well as hybrid models incorporating LSTM. The models were evaluated across different training set sizes ranging from 10% to 90%.

RF and its ensemble variants consistently outperformed other models, achieving mean F1-scores ranging from 83.40% to 91.02%. Notably, the combination of RF with iForest demonstrated the highest mean F1-score of 91.02%, surpassing other configurations. While DT achieved respectable mean F1-scores (82.45% to 87.23%), their performance was generally lower compared to RF-based models.

Dimensionality reduction through PCA yielded mixed results, with improvements observed for smaller training set sizes but diminishing returns for larger datasets. LSTM-based models showed potential but did not match the performance of ensemble methods.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table 5: Comparative Analysis of Proposed System with Existing Methods | | | | | | |
| *Attributes* | *[7]* | *[8]* | *[9]* | *[12]* | *[15]* | *Proposed System* |
| *Dataset Size and Diversity* | Cleveland dataset | Cleveland dataset | Two public datasets | Cleveland, Hungary, Switzerland, Long Beach V, UCI Kaggle | Cleveland, Comprehensive, CVD datasets | Combined hospital data |
| *Data Quality and Preprocessing Techniques* | Missing values removed, SS, Min-Max | Missing values removal, Binary conversion | SMOTE, Splitting, Balancing | Quality implication, Standard preprocessing | Data type conversion, Outliers handling, SMOTE, Under-sampling | Standardization, Scaling, Encoding, PCA, iForest, Data cleaning, Feature selection |
| *Feature Engineering* | Relief, MRMR, LASSO | HRFLM method | Significant features | 14-attribute dataset created | Chi-Square, ANOVA, FS, FE | - |
| *ML Algorithms* | ANN, LR, KNN, SVM, DT, NB | NB, GLM, LR, DL, DT, RF, GBT, SVM | SVM, SGD, KNN, ET, XGB, LR | LR, KNN, SVM, GBC | LR, DT, RF, KNN, NB, SVM, MLP, QMBC | NB, DT, RF, KNN, SVM |
| *Hyperparameter Tuning* | Leave-one-subject-out | - | Hyperband, SMOTE | GridSearchCV | Minimal, Basic evaluation | Optimized "random\_state" |
| *Evaluation Metrics* | Accuracy, Specificity, Sensitivity, MCC | Accuracy, Sensitivity, Specificity, Precision, F-Measure | Accuracy, F1-Score, MCC | Accuracy, Precision, Recall, F1-Score | Accuracy, Sensitivity, Specificity, Precision, Recall, F1-Score, AUC | Accuracy, Recall, Specificity, Precision, F1-Score, AUC |
| *ML Model, Train-Test Ratio, Evaluation Metric* | SVM+Relief+ LASSO+FCMIM+LLBFS, Accuracy (92.37%) | HRFLM (RF+LM), 7:3, Accuracy (88.7%) | HB+SMOTE+ET, 3:1, F1-Score (95.78%) | XGB+ GridSearchCV, 7:3, Accuracy (99.03%) | Cleveland Dataset: QMBC+Anova+ PCA, 8:2, F1-Score (98.59%) | RF+iForest, 8:2, F1-Score (95.84%) |
| CVD Dataset: QMBC+Chi-Square+PCA, 8:2. F1-Score (99.92%) |
| HD Dataset: QMBC+Anova+ PCA, 8:2. F1-Score (98.42%) |
| *Novelty of Approach* | Fast mutual information | HRFLM hybrid model | Hyperband, balancing | Near-zero prediction error, unique model | Novel QMBC, improved performance | Encryption, integrated techniques |

A comparative analysis of the proposed system with existing methodologies is presented in Table 5. This table systematically contrasts various attributes, including dataset characteristics, preprocessing techniques, feature engineering, ML algorithms, hyperparameter tuning, evaluation metrics, and performance across different approaches.

Our proposed system utilizes a combined hospital dataset, enhancing data diversity and robustness. This contrasts with smaller datasets employed in previous studies [7, 8]. Rigorous preprocessing, including standardization, scaling, and feature selection, differentiates our approach from those using basic methods or techniques like SMOTE [9, 12].

By incorporating a diverse set of ML algorithms and optimizing hyperparameters, our system achieves superior performance metrics, such as accuracy, sensitivity, specificity, and F1-score, compared to existing models like HRFLM [8] and Hyperband [9]. The results presented in Table 4 underscore the effectiveness and reliability of our proposed system, positioning it as a significant advancement in the field.

# Conclusion and Future Works

This paper presents a framework for heart disease prediction, prioritizing data privacy and security. It employs a multi-step approach, including data collection, encryption, preprocessing, model training (DT, RF, KNN, SVM), and evaluation. By combining these techniques with dimensionality reduction and anomaly detection, the framework aims to enhance model performance and robustness.

The experimental results demonstrate the effectiveness of RF and its ensemble variants. Notably, the combination of RF with iForest achieved the highest mean F1-score of 91.02%, surpassing other configurations. While DT achieved respectable mean F1-scores (82.45% to 87.23%), their performance was generally lower compared to RF- based models. Dimensionality reduction through PCA yielded mixed results, with improvements observed for smaller training set sizes (up to 5%) but diminishing returns for larger datasets. LSTM-based models showed potential but did not match the performance of ensemble methods (achieving an average 10% lower F1-score).

The proposed system prioritizes data privacy by encrypting sensitive patient information and ensuring that model training is conducted on encrypted data. This approach safeguards patient confidentiality while enabling accurate heart disease prediction.

Future work could explore the integration of advanced deep learning architectures, such as CNNs and RNNs, to potentially improve prediction accuracy. Additionally, investigating the impact of different encryption algorithms and their computational efficiency would be valuable.

By addressing the challenges of data privacy, model accuracy (achieving over 90% F1-score for RF ensembles), and computational efficiency, this research contributes to the development of reliable and secure heart disease prediction systems.

# References

1. Thanga Selvi R, Muthulakshmi I. RETRACTED ARTICLE: An optimal artificial neural network based big data application for heart disease diagnosis and classification model. Journal of Ambient Intelligence and Humanized Computing. 2021 Jun;12(6):6129-39.
2. Bazoukis G, Stavrakis S, Zhou J, Bollepalli SC, Tse G, Zhang Q, Singh JP, Armoundas AA. Machine learning versus conventional clinical methods in guiding management of heart failure patients—a systematic review. Heart failure reviews. 2021 Jan;26:23-34.
3. Heidenreich PA, Trogdon JG, Khavjou OA, Butler J, Dracup K, Ezekowitz MD, Finkelstein EA, Hong Y, Johnston SC, Khera A, Lloyd-Jones DM. Forecasting the future of cardiovascular disease in the United States: a policy statement from the American Heart Association. Circulation. 2011 Mar 1;123(8):933-44.
4. Makhlouf A, Boudouane I, Saadia N, Ramdane Cherif A. Ambient assistance service for fall and heart problem detection. Journal of Ambient Intelligence and Humanized Computing. 2019 Apr 12;10:1527-46.
5. Ghwanmeh S, Mohammad A, Al-Ibrahim A. Innovative artificial neural networks-based decision support system for heart diseases diagnosis.
6. Al-Shayea QK. Artificial neural networks in medical diagnosis. International Journal of Computer Science Issues. 2011 Mar 1;8(2):150-4.
7. Li JP, Haq AU, Din SU, Khan J, Khan A, Saboor A. Heart disease identification method using machine learning classification in e-healthcare. IEEE access. 2020 Jun 9;8:107562-82.
8. Mohan S, Thirumalai C, Srivastava G. Effective heart disease prediction using hybrid machine learning techniques. IEEE access. 2019 Jun 19;7:81542-54.
9. Abdellatif A, Abdellatef H, Kanesan J, Chow CO, Chuah JH, Gheni HM. An effective heart disease detection and severity level classification model using machine learning and hyperparameter optimization methods. ieee access. 2022 Jul 18;10:79974-85.
10. Das RC, Das MC, Hossain MA, Rahman MA, Hossen MH, Hasan R. Heart disease detection using ml. In2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC) 2023 Mar 8 (pp. 0983-0987). IEEE.
11. Haq AU, Li J, Memon MH, Memon MH, Khan J, Marium SM. Heart disease prediction system using model of machine learning and sequential backward selection algorithm for features selection. In2019 IEEE 5th International Conference for Convergence in Technology (I2CT) 2019 Mar 29 (pp. 1-4). IEEE.
12. Ahmad GN, Fatima H, Ullah S, Saidi AS. Efficient medical diagnosis of human heart diseases using machine learning techniques with and without GridSearchCV. IEEE Access. 2022 Apr 8;10:80151-73.
13. Sharma V, Yadav S, Gupta M. Heart disease prediction using machine learning techniques. In2020 2nd international conference on advances in computing, communication control and networking (ICACCCN) 2020 Dec 18 (pp. 177-181). IEEE.
14. Patidar S, Jain A, Gupta A. Comparative analysis of machine learning algorithms for heart disease predictions. In2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS) 2022 May 25 (pp. 1340-1344). IEEE.
15. Kapila R, Ragunathan T, Saleti S, Lakshmi TJ, Ahmad MW. Heart disease prediction using novel quine McCluskey binary classifier (QMBC). IEEE Access. 2023 Jun 26;11:64324-47.
16. Zhang H. The optimality of naive Bayes. Aa. 2004 Jan 1;1(2):3.
17. Hall LO, Chawla N, Bowyer KW. Decision tree learning on very large data sets. InSMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218) 1998 Oct 14 (Vol. 3, pp. 2579-2584). IEEE.
18. Pal M. Random forest classifier for remote sensing classification. International journal of remote sensing. 2005 Jan 1;26(1):217-22.
19. Zhang S, Li X, Zong M, Zhu X, Wang R. Efficient kNN classification with different numbers of nearest neighbors. IEEE transactions on neural networks and learning systems. 2017 Apr 12;29(5):1774-85.
20. Jakkula V. Tutorial on support vector machine (svm). School of EECS, Washington State University. 2006;37(2.5):3.
21. Wold S, Esbensen K, Geladi P. Principal component analysis. Chemometrics and intelligent laboratory systems. 1987 Aug 1;2(1-3):37-52.
22. Liu FT, Ting KM, Zhou ZH. Isolation forest. In2008 eighth ieee international conference on data mining 2008 Dec 15 (pp. 413-422). IEEE.
23. Schmidhuber J, Hochreiter S. Long short-term memory. Neural Comput. 1997 Nov 15;9(8):1735-80.
24. Muraina I. Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts. In7th international Mardin Artuklu scientific research conference 2022 May 1 (pp. 496-504).