A Collaborative Framework for Heart Disease Prediction Using Machine Learning Models on ElGamal Encrypted Patient Data

Abstract

This research presents a new framework to predict heart disease while prioritizing patient privacy. It involves multiple entities working together to provide a platform for the effective data collection, secured management, preprocessing, and analysis of medical records. The framework incorporates ElGamal encryption to ensure that sensitive patient information is well-guarded throughout the entire lifecycle of the data. A comprehensive set of machine learning algorithms, such as Naïve Bayes, Decision Trees, Random Forests, K-Nearest Neighbors, Support Vector Machines, and Long Short-Term Memory networks, are applied to the preprocessed data. To improve the effectiveness of predictions, dimensionality reduction techniques such as Principal Component Analysis and anomaly detection through Isolation Forest are applied. Hyperparameter optimization tuning is performed to ensure that the models are fine-tuned for optimal results. Empirical evaluation demonstrates the strong performance of the proposed framework, with Random Forest combined with Isolation Forest achieving an F1-score of 94.14%. The robust performance emanates from the well-effective combination of feature engineering, model selection, and optimization of hyperparameters. Overall, this research makes a valuable contribution to heart disease prediction by not only focusing on the high F1-score achieved through advanced machine learning techniques but also ensuring the privacy of sensitive data.

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

# Introduction

Heart disease remains the leading cause of deaths across the globe and one of the major public health challenges afflicting nearly 26 million people all over the world [[1]](#_References) and is likely to escalate prevalence in the future due to the growing populations of aged individuals [[2]](#_References). Cardiovascular Disease (CVD) is becoming an increasingly important burden on health-care systems, and in the United States alone, it accounts for 17 percent of total health expenditures [[3]](#_References). Innovations in diagnostic tools and medical management, however, have not improved the situation. Modern equipment patients cannot expect to have such access, especially those who are based in rural areas. Therefore, they rely on intuition from physicians to seek recommendations and treatment early [[4]](#_References). Pattern recognition from high-quality datasets has recently shown promising results in machine learning (ML) in terms of the transformative capability of improving the diagnosis of diseases and prediction accuracy [[5, 6]](#_References). Accordingly, the current study proposes a powerful ML system for heart disease prediction to cover the diagnostic gap with patient safety and reduced healthcare costs.

Most current diagnostic methods fall behind diagnosis, particularly at early stages, owing to accuracy, efficiency, and interpretability. Traditional models also tend to miss complex interactions of features along with temporal dependence in the data, which eventually leads to poor predictive performance. Also, privacy issues related to data poses concern about building strong models.

A hybridization model is being proposed in the present study, which includes the use of ensemble methods, dimensionality reduction, anomaly detection, and deep learning for solving the remaining issues. More robust and accurate prediction models are established, to this effect, integrating Random Forest Classifier (RF), Principal Component Analysis (PCA), Isolation Forest (iForest), and Long Short-Term Memory (LSTM). The LSTM component enables the capture of temporal patterns in patient data that might be overlooked by traditional ML models. Ensuring data security is also a focus, with encryption and privacy measures in place to safeguard patient confidentiality while supporting the development of effective models.

The hybrid method here distinguishes itself from other existing methods by multiple features (PCA and iForest); temporal dynamics (LSTM); and ensemble power (RF). Also, the data security part is fortified with ElGamal encryption, which ensures that the intermediate parties cannot change the data while it is encrypted during the application of ML techniques. All these are expected to help enhance predictive F1-score, shed more light on the underlying patterns related to heart disease, and ultimately translate to improved patient outcomes through earlier interventions. It can notably translate to reduced morbidity and mortality associated with heart disease and reduced healthcare costs. Additionally, the focus of the model on data security and privacy aligns with emerging healthcare regulations and patient expectations and, therefore, builds trust in the healthcare system. Therefore, if it succeeds in implementing the model, there is a big possibility of benefiting public health by reducing the global burden of heart diseases.

This paper attempts to assess the efficiency of various models of ML, along with their combinations, by employing the entire range of metrics like accuracy, specificity, recall, Positive Predictive Value (PPV), Negative Predictive Value (NPV), F1-score, and Area Under Curve (AUC). The strength of the aforementioned proposed approach is evaluated by viewing the effect of variation of training set size on the performance of the model.

# Literature Review

Heart disease prediction is a primary area where a lot of research work has been done through different ML techniques and feature selection methods to enhance diagnostic accuracy. One of the studies optimized the feature selection along with classifier performance in the diagnosis of heart diseases [[7]](#_References).

This study compares established techniques like: Relief, Minimum Redundancy and Maximum Relevance (MRMR), Least Absolute Shrinkage and Selection Operator (LASSO), LLBFS to novel FCMIM algorithm for their influence on classifiers such as Logistic Regression (LR), KNN, Artificial Neural Networks (ANN), SVM, NB, and DT. The combined best results found by this study were FCMIM and SVM which achieved the maximum accuracy of 92.37% using Leave-One-Out Cross-Validation. Though the Deep Neural Networks (DNN) have practically performed well, the limited data available have made their performance poor. Another study proposed an HRFLM model which hybridizes RF and Linear Model to provide improved heart disease classification [[8]](#_References). Here, the experimental results show accuracy much more elevated than traditional methods, thus proving the superiority of this method.

Another research work was conducted to achieve proper prediction regarding not just the presence or absence of CVD; it also tried to predict severity levels [[9]](#_References). Here, some ML models like SVM, KNN, LR, Stochastic Gradient Descent (SGD), and tree-based ensembles were employed to overcome the problem concerning imbalanced data via Synthetic Minority Oversampling Technique (SMOTE) and hyperparameter optimization with HPO. The remarkable point here is that the tree-based ensemble models do outperform the others in terms of accuracies concerning CVD presence/absence that amounts to 99.2% and 98.52%, respectively, and for severity level predictions at 95.73%. The continuing research compared six models including Extreme Gradient Boosting (XGB), Bagging, RF, DT, KNN, and NB using a dataset of more than 300,000 cases [[10]](#_References). The top scorer was scored by XGB with 91.30% accuracy 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]](#_References).

Subsequent studies compared LR, KNN, SVM, and XGB classifiers, optimized with Grid Search Cross-Validation (GridSearchCV) [[12, 13, 14]](#_References). RF consistently demonstrated superior performance, with accuracies reaching 99% and 98.53%. A more recent study introduced ensemble methods and a novel Quine McCluskey Binary Classifier (QMBC) classifier, along with feature selection techniques like Chi-Square and Analysis of Variance (ANOVA), further improving prediction accuracy [[15]](#_References).

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

## ElGamal Cryptosystem

The ElGamal cryptosystem is a public-key encryption scheme based on the discrete logarithm problem. Given a cyclic group *G* of prime order *p* with generator *g*, a private key *a* is chosen randomly from *{1, ..., p−1}*. The public key is *ga mod p*. To encrypt *a* message *m ∈ G*, a random integer *k* is chosen, and the ciphertext is *(c1, c2) = (gk mod p, m ∗ (ga)k mod p)*. Decryption is done by computing *c2 / (c1a) mod p* [[16]](#_References).

## Naïve Bayes Classifier (NB)

NB is a simple probabilistic classifier that assumes feature independence given the class label. Despite this assumption, it often performs well due to its focus on finding the class with the highest probability. Recent research suggests that NB can be optimal even when feature independence is violated, as long as dependencies balance out across classes [[17]](#_References).

## Decision Tree (DT)

The DT is a model for classification and regression that decides based on partitioning the data into subsets, using feature-valued attributes for this purpose. 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 [[18]](#_References).

## 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. By aggregating the predictions from many trees, RF improves accuracy and reduces overfitting compared to single DTs [[19]](#_References).

## K-Nearest Neighbors (KNN)

The KNN algorithm is a simple yet effective supervised learning technique for both classification and regression tasks. Given a query point, KNN 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) [[20]](#_References).

## Support Vector Machine (SVM)

SVM is 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. SVM employs the kernel trick for handling more complex, non-linearly separable datasets. Through this technique, data is mapped into higher dimensional space where it becomes linearly separable [[21]](#_References). SVMs typically use kernel functions such as linear, polynomial, RBF, and sigmoid.

## Principal Component Analysis (PCA)

This PCA is a technique that is mostly known for reducing the dimensions of data, making it recognize patterns within data and subsequently reforming it into a new coordinate system. It is made up of standardization, covariance matrix computation, eigenvalue decomposition, selection of principal components, and projection of results into the new system. PCA has many uses including dimensionality reduction, feature extraction, noise removal, visualization, and outlier detection [[22]](#_References).

## Isolation Forest (iForest)

iForest uses a collection of isolation trees to separate abnormal observations from the distribution. Path lengths into isolation are shorter for anomalies than for normal observations. iForest has several advantages: its linear time complexities, capability to deal with high-dimensional data and tolerance against noise and outliers. It has outperformed many methods with respect to accuracy and efficiency [[23]](#_References).

## Long Short-Term Memory (LSTM)

LSTM networks are a type of recurrent neural network (RNN) specifically designed to solve the vanishing gradient problem, which prevents standard RNNs from displaying effective performance for long-term dependencies on sequential data. This is accomplished through a special cell structure that contains input, output, and forget gates. These gates allow information to flow into and out of cells, storing and making it possible to retrieve information over long-term history [[24]](#_References). They can perform a task very efficiently when the length of many input sequences provides context, such as in NLP tasks and speech recognition.

## Train-Test Split

Train-test split is a method of evaluation of ML models. In this approach, data is divided into a training set (to fit the model) and a test set (to measure performance). Some common ratios for the split are 70:30 or 80:20. Well-chosen ratios are important because poor ratios can lead to misinterpretation of performance metrics due to a shortage or an over-abundance of test data [[25]](#_References).

## K-Fold Cross-Validation

K-fold cross-validation is one of the ways to evaluate the performance of models by splitting the dataset into k equal sections or folds. k−1 folds are used to train the model while the last remaining fold serves as a testing set. This process is repeated on every fold so that every data point is included for both training and validation [[26]](#_References).

# System Model and Functional Roles

This section provides a detailed architectural blueprint of the system, as shown in Figure 2. It defines the system's components, their interactions, and respective functions. Additionally, it outlines the sequential flow of activities within the system.

## Roles of each entity

The proposed system comprises six primary entities: Central Coordinator (*CC*), Cryptographer (*CR*), Service Provider (*SP*), System (*SYS*) and System User (*SU*), as illustrated in Figure 1.

*CC*The *CC* oversees the entire process, starting by gathering raw datasets related to heart disease from various hospitals and combining them into a unified dataset. It generates public-private key pairs, shares the public keys with the *CR* for data encryption, and securely stores the private keys. Once the encrypted dataset is received from the *CR*, the *CC* validates its integrity by decrypting it with the private keys and verifying the data. If the dataset is deemed valid, the *CC* forwards it to the *SP* for model training. After receiving the trained models, test datasets, preprocessing scalers, and F1-scores from the *SP*, the *CC* tests the models on its own encrypted dataset and compares the F1-scores with those reported by the *SP*. If the models pass validation, the *CC* develops the SYS using the best-performing model and preprocessing scaler. This SYS is then deployed for SUs to use.

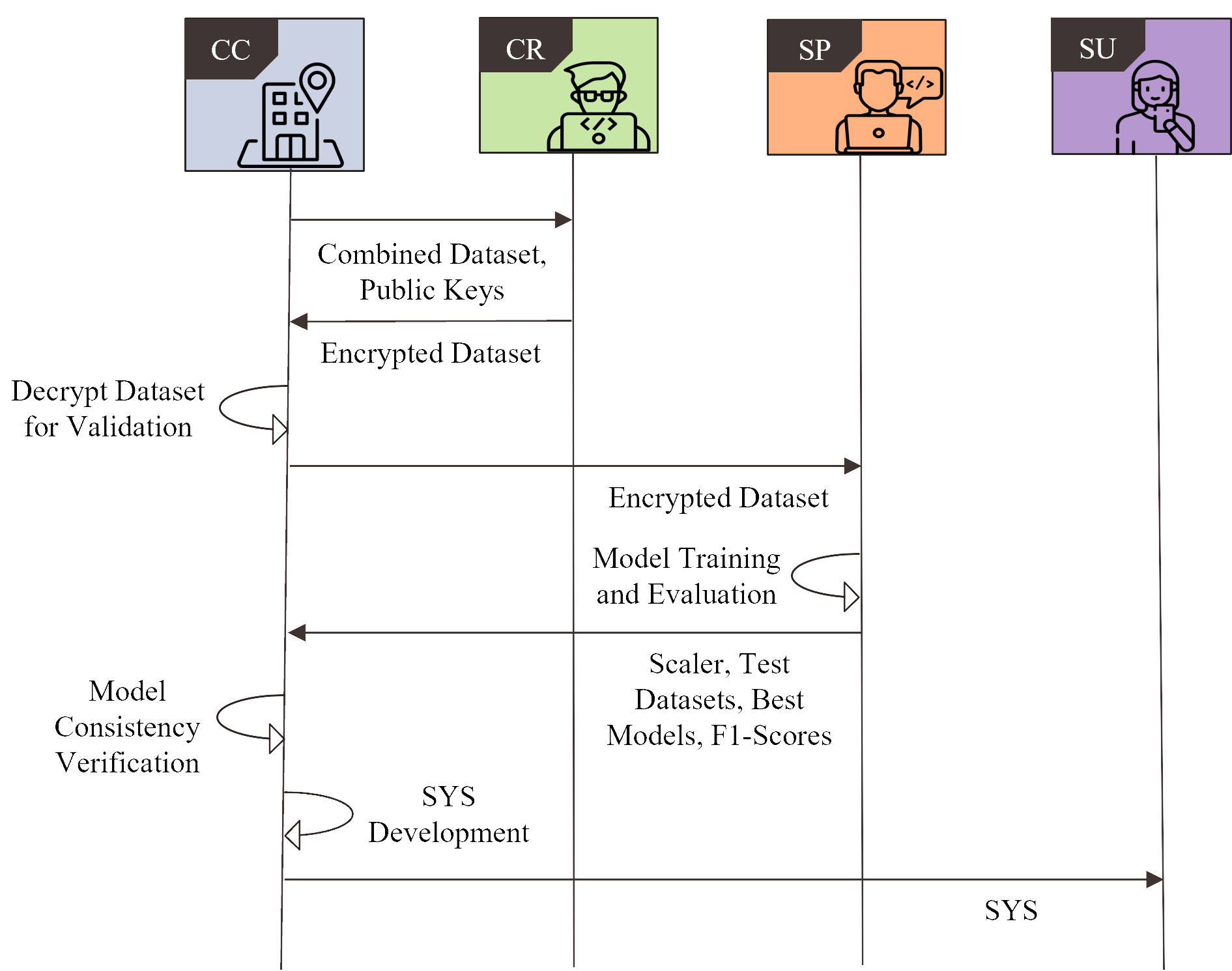


Fig. 1 Interactions among entities

*CR* The *CR* receives the unified dataset from *CC* and encrypts it using the public keys shared by *CC*, ensuring the data remains confidential. Once encrypted, the dataset is sent back to *CC*.

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

*SP* The *SP* preprocesses the encrypted dataset received from *CC* and trains several ML models. After evaluating these models using metrics like F1-score, the *SP* selects the best-performing model. The trained models, test datasets, preprocessing scalers, and F1-scores for the selected model are then shared with *CC*.

*SYS* The *SYS* is the final application developed by *CC*, using the trained model and preprocessing scaler provided by *SP*. It takes input data from *SU*s, encrypts it with the public keys shared by *CC*, preprocesses the data, and then makes predictions based on the trained model.

*SU* The *SU*s interact with the *SYS* to check whether they have heart disease.

## Table of Notations

Table 1 Notation Used

|  |  |
| --- | --- |
| *Symbol* | *Description* |
| *ωk* | Raw dataset from the *k*-th Kaggle repository, where *k* ∈ {1, 2, ..., 7} |
| *ω* | *ω*1, *ω*2, …, *ω*7 |
| *Φ* | Dataset features |
| *d* | No. of crucial features |
| *r* | No. of medical records |
| *ξ* | Crucial features (*ξ*1,*ξ*2, …, *ξd*) |
| *λ* | Label value (Categorial) |
| *i ∈ d* | *i*-th column |
| *j ∈ r* | *j*-th row |
| *Δ* | Combined Dataset (*Δ*1, *Δ*2, …, *Δd*, *λ*) |
| *∅* | Missing value |
| *κ* | ​Public keys (*pi, gi, hi*) |
| *ρ* | Private keys (*pi, gi, ai*) |
| *E(κi, ξi)* | ElGamal encryption function |
| *λ'* | Encoded label value |
| *Θ* | Encrypted dataset (*Θ*1­,*Θ*2,​ …, *Θd*, *λ’*) |
| *D(ρi, Θi)* | ElGamal decryption function |
| *Υ* | Decrypted dataset (*Υ*1­,*Υ*2,​ …, *Υd*, *λ’’*) |
| *θ* | Standardized Dataset (*θ*1­,*θ*2,​ …, *θd*, *λ’*) |
| *η*1*, η*2*, η*3*, η*4 *η*5 | ML models: NB, DT, RF, KNN, SVM |
| *η*α | Best performing model |
| *ηα*1*, ηα*2*, ηα*3*, ηα*4 | Base model, *ηα* with PCA, *ηα* with iForest, *ηα* with LSTM |
| *{ηαk*1*, ..., ηαk*5*} ∈ ηαk* | Models of *ηαk* for 5-fold cross validation, where *k* ∈ {1, 2, 3, 4} |
| *ηβ ∈ {ηα*1*, ηα*2*, ηα*3*, ηα*4*}* | Best configuration of {*ηα*1, *ηα*2, *ηα*3, *ηα*4} |
| *{ηβ*1*, ..., ηβ*5*} ∈ ηβ* | Models of *ηβ* for 5-fold cross validation |
| *ηγ* | Best performing model |
| *Π*1*, Π*2*, Π*3*,Π*4*, Π*5*,Π*6*,Π*7*,Π*8 | Confusion Matrix, Accuracy, Specificity, Recall, Positive Predictive Value (PPV), Negative Predictive Value (NPV), F1-score, AUC |
| *τ* | Train-test split ratios (0.1:0.9 to 0.9:0.1) |
| *δ* | Preprocessing transformer |

## Individual stages

The proposed framework consists of nine distinct stages. These are: (1) Data Collection, (2) Data Encryption, (3) Data Decryption and Verification, (4) Data Preprocessing, (5) Model Training, (6) Model Optimization, (7) Final Model selection, (8) Model verification and (9) System Development. These stages are detailed below, with notations referenced from Table 1:

4.3.1 Data Collection

The *CC* is responsible for collecting datasets from various hospitals for heart disease prediction. Hospitals may share anonymized or de-identified medical data with the *CC* for research or clinical studies, under strict ethical approvals, consent from patients, and formal data-sharing agreements. For this research, data was gathered from seven publicly available Kaggle repositories [[27-33]](#_References). The datasets, denoted as *ω*1​, *ω*2, ..., *ω*7, contain several inconsistencies. These include irrelevant features that aren't necessary for prediction. The features consist of both numerical and categorical values, and some of them are encoded differently across the *ω*.

To ensure consistency, *CC* pre-processes the data by retaining only the relevant features (*ξ*, where *ξ ⊂ Φ*) from all available features (*Φ*), while discarding the irrelevant ones. *CC* manually transforms all the values across the datasets to ensure a uniform representation, standardizing the types of data within the entire set. To ensure consistency, *CC* reviews the Kaggle dataset descriptions to identify the data type for each *ξi* and *λ* in *ω*. A standard data type is then chosen for all features and label, and the values in each dataset are converted to this standardized format, ensuring uniformity in attribute names and value types across all datasets. Once pre-processed, *ω* are combined into a single, unified dataset, *Δ =* .

Even though the data is anonymized, sharing it with the *SP* in its original form could still expose sensitive patterns or correlations that may lead to the re-identification of individuals. Encrypting the dataset ensures that this risk is minimized. So, *CC* generates d pairs of public and private keys (*κi* and *ρi*) for *d* attributes following the ElGamal cryptosystem as described in section [3.1](#_ElGamal_Cryptosystem). The key generation process is detailed in Algorithm 1. The *Δ*, along with the *κ*, is then transferred to *CR* to make the dataset secure.

*CC* trusts *CR* as a reliable entity, but since *CR* handles the original data during encryption, there's a risk of leakage. To mitigate this, *CC* checks the integrity of the encrypted dataset before sending it to the *SP*.

|  |
| --- |
| Algorithm 1Key Generation |
| 1. **Input:** *None* 2. **Output:** Public keys *κ,* Private keys *ρ* 3. **function** *gen\_keys():* 4. *ρ, κ = [], []* 5. **for** *i* ∈ *d* **do**: 6. *p* ← **random prime** [1020, 1050] 7. *g* ← **random int** [2, *p*-1] 8. *a* ← **random int** [1, *p*-2] 9. *h* ← *ga mod p* 10. *ρ.append({p, g, a})* 11. *κ.append({p, g, h})* 12. **end for** 13. **return** *κ, ρ* 14. **end function** |

Δ has the following attributes as detailed in Table 2:

Table 2 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, while Figure 4 highlights the relationships between attributes using a correlation heatmap.

4.3.2 Data Encryption

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

The *CR* secures the dataset *Δ*, which contains *d* feature vectors (*ξ*1, *ξ*2, …, *ξ*d​) and their associated label *λ*, by applying the encryption function *E(κi, ξi)*. Each *ξi* is encrypted as described in section [3.1](#_ElGamal_Cryptosystem), using its corresponding public key *κi*, provided by *CC*, resulting in its encrypted form *Θi*​, following the steps outlined in Algorithm 2. For *λ*, *CR* employs a straightforward binary encoding scheme, converting 'Yes' to 1 and 'No' to 0, producing *λ′*. The fully encrypted dataset *Θ* is then sent to *CC* for validation.

|  |
| --- |
| Algorithm 2 Data Encryption |
| 1. **Input:** Public key *κi,* Private key *ξi* 2. **Output:** Encrypted Column *Θi* 3. **function** *E(κi, ξi):* 4. *p, g, h* ← *κi['p'], κi['g'], κi['h']* 5. *k* ← **random int** [1, *p*-2] 6. *Θi ← []* 7. **for** *j* ∈ *r* **do**: 8. **if** *ξij* *≠* *∅* **then**: 9. *ξ'ij ← datatype\_to\_int(ξij)* 10. *c1* ← *gk mod p* 11. *c2 ← (ξ'ij × hk mod p) mod p* 12. *Θi.append((c1, c2))* 13. **else**: 14. *Θi.append(∅)* 15. **end if** 16. **end for** 17. **return** *Θi* 18. **end function** |

4.3.3 Data Decryption and Verification

The *CC* uses the private keys *ρi* and the decryption function *D(ρi, Θi)* to decrypt *Θi*, obtaining the decrypted features *Υi* as described in section [3.1](#_ElGamal_Cryptosystem), following the steps in Algorithm 3. The *CC* also decodes the labels *λ′*, converting 1 back to ‘Yes’ and 0 to ‘No,’ to produce *λ′′*. The *CC* then compares *Υi*​ and *λ′′* with the original *Δi* and *λ* to ensure no data was altered during encryption or transmission by the *CR*. If all *Δi* and *λ* values match their corresponding *Υi* and *λ′′* values, the *CC* forwards *Θ* to the *SP* for model training.

|  |
| --- |
| Algorithm 3 Data Verification |
| 1. **Input:** Private key*ρi,* Encrypted Column *Θi* 2. **Output:** "Valid" or "Invalid" 3. **function** *D(ρi, Θi):* 4. *p, g, a ← ρi['p'], ρi['g'], ρi['a']* 5. *dec, miss ← [], []* 6. **for** *j* ∈ *r*: 7. **if** *ξij ≠ ∅:* 8. *c1, c2 ← split(Θij)* 9. *temp* ← *(c1a mod p)-1 mod p* 10. *val* ← *int\_to\_datatype((c2 \* temp) mod p)* 11. *dec.append(val)* 12. **if** *val ≠ ξij*: 13. *miss.append(j)* 14. **end if** 15. **end if** 16. **end for** 17. **return** "Valid" **if** *miss.len*= 0 **else** "Invalid" 18. **end function** |

|  |
| --- |
| Fig. 4 Correlation Heatmap of Features for Heart Disease Prediction |

4.3.4 Data Preprocessing

The *SP* addresses missing values *∅* in *Θ* by imputing the mean value for each *Θi*​. Missing values in *Θij*​ are replaced with the mean of all non-missing values in *Θi*​, calculated as:

|  |  |
| --- | --- |
|  | (1) |

After imputation, *Θij*​ is standardized using the StandardScaler technique, denoted as δ. Standardization ensures all features have a mean of zero and a variance of one, transforming *Θij*​ into *θij*​:

|  |  |
| --- | --- |
|  | (2) |

where, Mean,

|  |  |
| --- | --- |
|  | (3) |

and, Standard Deviation,

|  |  |
| --- | --- |
|  | (4) |

The entire process, handling missing values through mean imputation and scaling the data via standardization, is denoted as *δ* for reference.

4.3.5 Model Training

The *SP* partitions *θ* into training (*θtrain*) and testing (*θtest*) subsets. It then evaluates a selection of ML models, including NB (*η*1), DT(*η*2), RF(*η*3), KNN(*η*4), and SVM(*η*5), chosen for their ability to handle categorical data without requiring decrypted features. For each model, the *SP* computes various performance metrics: Confusion Matrix (*Π*1), Accuracy(*Π*2), Specificity(*Π*3), Recall(*Π*4), PPV(*Π*5), NPV(*Π*6), F1-score(*Π*7) and AUC(*Π*8).

Additionally, the *SP* performs hyperparameter optimization to maximize the *Π*7 for each model, as the rationale for selecting Π7​ is explained in section [6](#Results). To further refine the analysis, the *SP* explores the impact of different train-test split ratios (*τ*) on model performance, evaluating *Π*7​ for *τ* values ranging from 0.1:0.9 to 0.9:0.1 across all models. Based on the highest *Π*7​ values, the *SP* selects the best-performing model (*ηα*​) for further evaluation and potential hyperparameter tuning.

4.3.6 Model Optimization

The *SP* conducts a thorough evaluation of *ηα*, the model with the best hyperparameters that produced the highest *Π*7. *SP* applies 10-fold cross-validation, splitting the *θ* into 10 parts. Each part is used once as a validation set, while the others are used for training, ensuring an accurate and unbiased assessment. *SP* also uses three advanced techniques: PCA, iForest, and LSTM, for deeper analysis.

The following hyperparameters are fine-tuned for the models:

1. RF: The number of trees (n\_estimators), maximum tree depth (max\_depth), minimum samples required to split nodes (min\_samples\_split), and split criterion (Gini).
2. PCA: The number of principal components to retain (n\_components) to balance dimensionality reduction and variance.
3. iForest: The number of trees (n\_estimators), contamination rate (the proportion of outliers expected in the data), and sample size (max\_samples) for anomaly detection.
4. LSTM: The number of LSTM units, learning rate, dropout rate, number of epochs, and batch size for sequential tasks.

4.3.7 Final Model Selection

The *SP* conducts a comprehensive evaluation of four configurations of the best-performing model ηα: the base model (*ηα*1), *ηα* with PCA (*ηα*2), *ηα* with iForest (*ηα*3), and *ηα* with LSTM (*ηα*4). For each configuration, *SP* applies 10-fold cross-validation, generating 10 sets of *θtrain* and *θtest*, where each set includes the randomly selected rows along with their corresponding serial numbers. Additionally, 10 trained models (*ηαk*1, ..., *ηαk*10, where *k* ∈ {1, 2, 3, 4}), and 10 corresponding *Π*7 scores. The average *Π*7 score across the 10 folds, denoted as *Π*7’, is calculated for each configuration.

The *SP* selects the configuration with the highest average *Π*7’ and identifies it as *ηβ* from {*ηα*1, *ηα*2, *ηα*3, *ηα*4}. All 10 models of *ηβ*, i.e., {*ηβ*1, ..., *ηβ*10}, along with their corresponding *θtest*, *Π*7, and *δ*, are then delivered to the *CC* for further validation.

4.3.8 Model Verification

The *CC* validates the models *ηβ* trained by the *SP*. For this, the *CC* randomly selects some or all of the *θtest* and retrieves the corresponding serial numbers from *θ*. Using these serial numbers, the *CC* extracts the corresponding rows from its own *Θ*. This subset of *Θ* is preprocessed using *δ* to produce *θ* and is then trained using the corresponding *ηβk*.

The *CC* then calculates the performance metric *Π*7 and compares it with the *Π*7 provided by the *SP*. If the calculated *Π*7 matches exactly with those from the *SP*, the *CC* proceeds to develop the *SYS*. The *ηβk* with the highest *Π*₇ will be selected as the best-performing model, *ηγ.*

4.3.9 System Development

The *CC* develops a user-friendly interface for *SU* interaction. This interface includes input fields for each of the model's d features (*ξi*), allowing *SU*s to input their data directly. The system encrypts each *ξi* using its corresponding public key *κi*, producing *Θi*. These encrypted values are then processed using δ to generate *θi*, which is fed into the *ηγ* to predict *λi'*. This prediction will inform the *SU*s whether they have heart disease or not. Once finalized, the *CC* makes the *SYS* publicly available for *SU*s.

# Security Threats and Solutions

In our proposed 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.

## Predictable Label Encoding

*Threat: If the CR applies label encoding to categorical features, the SP might infer the original values based on their alphabetical order. For example, if the feature is "Gender" and the encoded values are 0 and 1, the SP could guess that "Female = 0" and "Male = 1" based on their order in the alphabet. This could lead to unintended information leakage, potentially exposing sensitive attributes.*

*Solution:* By encrypting categorical values using the ElGamal encryption scheme, the encoded values will not follow any predictable pattern. Since encryption produces unique and non-sequential outputs, the *SP* cannot infer any order or derive the original values. This ensures that categorical attributes remain private and protected from unwanted inference.

## Data Confidentiality Breach

*Threat: The CR could tamper with the dataset during encryption, or the SP could potentially learn confidential information from the encrypted dataset during the model training process, compromising the data’s confidentiality and integrity.*

*Solution:* In section [4.3.1](#_Individual_stages), the *CC* generates a pair of keys and shares only the public ones along with the dataset to *CR* for encryption and decryption which excludes others from decrypting. Then, *CC* uses the private keys in section [4.3.3](#_Individual_stages) to verify the integrity of the encrypted dataset to confirm that the *CR* has not tampered with it. The *CC* forwards the encrypted dataset to the *SP* for model training, with the data remaining confidential and unchanged as it flows.

## Predictable Encrypted Data

*Threat: When the same encryption key is applied to many different features, the identical values in various features may generate the same encrypted result. This could allow an attacker to recognize patterns between features and infer that they might be related. If such an attacker were able to predict or decrypt one feature, personal values on some of the related ones could be compromised.*

*Solution:* By using distinct encryption keys for every feature this matter is taken care of in section [4.3.2](#_Individual_stages). In that case, if two features have the same value, their encodings will be different, so no attacker can identify any correlation between them. This will keep each feature independently secure, even if one of them is compromised.

## Model Manipulation

*Threat: The SP could sway the model or generate false performance measures thereby invalidating the model.*

*Solution:* In section [4.3.8](#_Individual_stages), *CC* validates the models by randomly selecting test data, extracting corresponding encrypted data, and comparing the performance metrics calculated by *CC* with those provided by the SP to ensure accuracy and integrity.

## Corruption Risk

*Threat: Corruption of CC, CR, or SP may result into manipulated data or wrong models or violation of privacy that will finally shake the accuracy and credibility of the heart disease prediction system.*

*Solution:* *CC* has a strong incentive to maintain the integrity of the system because if it doesn't work correctly, *SU*s will reject it, leading to the company's loss. *CC* ensures data integrity by decrypting and verifying the models and data. If corruption is detected, *CC* can take corrective actions, such as penalizing or replacing the corrupted parties (*CR* or *SP*), ensuring the system remains functional and trustworthy.

# Experimental Analysis

## Experimental Setup

The experiments were conducted on a system with an Intel Core i5-8265U, 8 GB RAM, and a 64-bit operating system. The development of models and their testing were conducted using Visual Studio Code and Jupyter Notebook (.ipynb) applying relevant machine learning libraries for heart disease prediction.

## Results

This section presents the results regarding the time needed for key generation, encryption, and model training.

In Table 3, key generation times are impressively quick, ranging from 0.02 to 0.04 seconds, with both private and public keys consistently sized at 0.87 KB.

Table 4 reveals that encrypting a dataset of about 296.81 KB takes between 11.56 and 14.56 seconds, while decryption times range from 14.21 to 16.34 seconds, highlighting the reliability of the encryption process.

Table 5 showcases the training times for various ML algorithms. NB and DT stand out for their speed, averaging between 0.50 and 0.64 seconds. In contrast, SVM requires significantly more time, ranging from 19.27 to 20.65 seconds.

Table 6 compares advanced training methods for RF. It shows that RF combined with PCA and iForest is quite efficient, taking only 2.61 to 4.16 seconds, whereas RF with LSTM takes considerably longer, up to 21.36 seconds.

Table 3: Time Required for Key Generation

|  |  |  |  |
| --- | --- | --- | --- |
| *Attempt* | *Generation Time (s)* | *Private Key Size (KB)* | *Public Key Size (KB)* |
| 1 | 0.04 | 0.87 | 0.87 |
| 2 | 0.02 | 0.87 | 0.87 |
| 3 | 0.03 | 0.87 | 0.87 |

Table 4: Time Required for Encryption and Decryption

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Attempt* | *Dataset Size (KB)* | *Encryption Time (s)* | *Encrypted Dataset Size (KB)* | *Decryption Time (s)* |
| 1 | 296.81 | 11.56 | 4487.03 | 14.21 |
| 2 | 13.80 | 4479.46 | 16.34 |
| 3 | 14.56 | 4493.29 | 16.31 |

Table 5: Time Required for Model Training

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Attempt* | *NB* | *DT* | *RF* | *KNN* | *SVM* |
| 1 | 0.54 s | 0.64 s | 5.62 s | 2.82 s | 20.65 s |
| 2 | 0.52 s | 0.61 s | 5.48 s | 2.78 s | 20.30 s |
| 3 | 0.50 s | 0.57 s | 4.90 s | 3.15 s | 19.27 s |

Table 6: Time Required for Training RF with Different Methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Attempt* | *RF* | *RF + PCA* | *RF + iForest* | *RF + LSTM* |
| 1 | 2.61 s | 0.51 s | 4.16 s | 21.12 s |
| 2 | 2.63 s | 0.52 s | 3.92 s | 21.02 s |
| 3 | 2.66 s | 0.53 s | 4.02 s | 21.36 s |

The performance of ML models was evaluated using several key metrics. To comprehensively assess model accuracy, a confusion matrix was employed, offering a tabular representation of the model's predictions compared to the actual ground truth values.

A confusion matrix, as detailed in [[15]](#_References), is a table that summarizes the performance of a classification model by comparing its predicted values with actual values. Predictions are grouped into one of four possible outcomes: true positive (TP), true negative (TN), false positive (FP), or false negative (FN).

Based on the confusion matrix, the following performance metrics were calculated:

*Accuracy* is the number of correct predictions (both positive and negative) divided by the total number of instances.

|  |  |
| --- | --- |
|  | (5) |

*Recall or Sensitivity* is the number of true positives correctly identified by the classifier model.

|  |  |
| --- | --- |
|  | (6) |

*Specificity* indicates how many true negatives were correctly identified divided by all false negatives.

|  |  |
| --- | --- |
|  | (7) |

*Precision or Positive Predictive Value (PPV)* indicates the proportion of correct predictions over all positive predicted instances.

|  |  |
| --- | --- |
|  | (8) |

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

|  |  |
| --- | --- |
|  | (9) |

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

|  |  |
| --- | --- |
|  | (10) |

*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.

|  |
| --- |
|  |
| Fig. 5 Performance Metrics of Classification Algorithms with Varying Train-Test Split |

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.

Model robustness and generalization were assessed by employing various train-test splits, as described in section [3.10](#_Train-Test_Split), with the training-to-testing data ratio ranging from 1:9 to 9:1. Models were trained on the training data and evaluated on the corresponding test data using metrics such as accuracy, precision, recall, PPV, NPV, F1-score, and AUC.

Figure 5 illustrates how the performance of different ML models on test data varies with changes in the train-test split, while Figure 6 summarizes their average performance metrics across these splits. The models evaluated include NB, DT, RF, KNN, and SVM.

RF stands out as the most effective algorithm, achieving consistently excellent results across all metrics. With high accuracy (90.44%), specificity (88.73%), recall (91.76%), PPV (91.47%), NPV (89.10%), F1-score (91.60%), and an impressive AUC (95.14%), RF proves to be highly reliable for classifying the dataset and delivering top-tier performance.



Fig. 6 Average Performance Metrics of Classification Algorithms with Varying Train-Test Split

SVM also performs well, particularly in metrics like recall (89.74%), PPV (76.47%), NPV (81.77%), and F1-score (82.63%). Although its accuracy (78.15%), specificity (62.50%), and AUC (85.59%) are slightly behind RF, SVM remains a strong contender, offering robust and reliable classification results.

DT provides a balanced performance across metrics, with accuracy (86.20%), specificity (85.82%), recall (86.48%), PPV (88.83%), NPV (83.04%), F1-score (87.63%), and AUC (86.15%). These results indicate that DT is a dependable model with consistently good classification capabilities.

NB and KNN share similar trends in their performance. Both models excel in metrics like recall, PPV, and AUC, but fall short in accuracy, specificity, and NPV. While they deliver satisfactory results in some areas, there is room for improvement, particularly in boosting specificity and NPV to strengthen their overall performance.

Increasing the training set size generally improves performance for all algorithms, but the rate of improvement diminishes after a certain point. Increasing the train-test split ratio from 0.1 to 0.9 helped identify trends in model behavior and provided evidence of stable generalization across varying dataset sizes.

RF is chosen as the best model due to its consistently superior performance across all metrics, indicating its robustness and generalization ability. In contrast, DT, NB, KNN and SVM perform relatively well in some metrics, but their performance is less consistent and shows weaknesses. These limitations make DT, NB, KNN and SVM less reliable for precise and consistent classification.

While all the performance metrics offer valuable insights, the F1-score is prioritized for comparing the models. This is due to its ability to balance precision and recall, both of which are critical in this context. Precision ensures that positive predictions are accurate, while recall ensures that a high proportion of actual positive cases are identified. By optimizing the F1-score, the goal is to develop a model that minimizes both false positives and false negatives.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 7: Performance Comparison of Classification Models with Varying Training-Testing Sets and Hyperparameters (F1-Score in %) | | | | | | | | |
| *Classifiers* | *Hyper-parameters* | | *5-Fold Validation* | | | | | *Mean ± Standard Deviation* |
| *K=1* | *K=2* | *K=3* | *K=4* | *K=5* |
| RF | StratifiedKFold   * n\_splits: 5 * shuffle=True * random\_state: 3838 | RandomForestClassifier   * random\_state: 3838 | 96.27 | 95.80 | 95.83 | 96.41 | 96.81 | 96.22 ± 0.004 |
| RF + PCA | StratifiedKFold   * n\_splits: 5 * shuffle=True * random\_state: 1114 | RandomForestClassifier   * random\_state: 1114 * n\_estimators: 78 | 96.72 | 96.94 | 95.57 | 95.77 | 95.89 | 96.18 ± 0.005 |
| PCA   * n\_components: 0.95 * random\_state: 1114 | |
| RF + iForest | StratifiedKFold   * n\_splits: 5 * shuffle=True * random\_state: 3022 | RandomForestClassifier   * random\_state: 3022 * n\_estimators: 83 * criterion: gini | 97.07 | 95.45 | 97.10 | 98.11 | 96.08 | 96.76 ± 0.009 |
| IsolationForest:   * contamination: 0.05 * random\_state: 3022 * max\_samples: 256 | |
| RF + LSTM | StratifiedKFold   * n\_splits: 5 * shuffle=True * random\_state: 52 | Optimizer (Adam)   * learning\_rate: ~0.143 (1/7) | 94.34 | 91.64 | 93.66 | 93.44 | 92.74 | 93.17 ± 0.009 |
| Model Training  (model.fit)   * epochs: 220 * batch\_size: 1200 * verbose: 0 | Sequential  Dropout   * rate: 0.05   Dense   * units: 1 * activation: sigmoid |
| RandomForestClassifier   * random\_state: 52 * n\_estimators: 61 | Model Compilation (model.compile)   * loss: binary\_crossentropy |

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 7 presents a comparative analysis of various RF-based models, including RF, RF with dimensionality reduction (PCA), RF with anomaly detection (iForest), and RF with LSTM, evaluated using 10-fold cross-validation as described in section [3.11](#_K-Fold_Cross-Validation).

RF with Isolation Forest demonstrated the highest mean F1-score of 96.76%, outperforming other configurations. RF achieved a mean F1-score of 96.22%, while RF with PCA showed a mean F1-score of 96.18%. RF with LSTM achieved the lowest mean F1-score of 93.17%.

These results show that RF with iForest delivers the best performance, slightly outperforming standalone RF model. RF with PCA showed a minor decline in performance, due to dimensionality reduction, which led to the loss of important information. RF with LSTM had the lowest score, as the integration of LSTM’s temporal learning with the RF model caused instability and overfitting, hindering its effectiveness.

The comparative analysis between the proposed system and the existing methodologies is described in Table 8. The table consists of a systematic comparison of diverse attributes such as characteristics of datasets, preprocessing methods, types of features engineered, ML algorithms, hyper-parameter tuning, evaluation metrics, and performances over different approaches. Our proposed system adopts a combined hospital dataset that adds to the diversity and robustness of the data as opposed to previous studies' smaller datasets [[7, 8]](#_References). Extensive preprocessing including standardization, scaling, and feature selection has differentiated our method from the others that utilize simple methods or techniques such as SMOTE [[9, 12]](#_References).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table 8: Comparative Analysis of Proposed System with Existing Methods | | | | | | |
| *Attributes* | [*[7]*](#_References) | [*[8]*](#_References) | [*[9]*](#_References) | [*[12]*](#_References) | [*[15]*](#_References) | *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 Kaggle Datasets |
| *Dataset Security* | - | - | - | - | - | Dataset encrypted by *CR* and verified by *CC*  Model trained on encrypted 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, ElGamal Encryption |
| *Feature Engineering* | Relief, MRMR, LASSO | HRFLM method | Significant features | 14-attribute dataset created | Chi-Square, ANOVA, FS, FE | 11 common features selected from 7 Kaggle datasets |
| *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 | - |
| *Evaluation Matrices* | 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, PPV, NPV, F1-Score, AUC |
| *ML Model, Result* | SVM+Relief + LASSO + FCMIM + LLBFS,  Accuracy (92.37%) | HRFLM (RF+LM),  Accuracy (88.7%) | HB+SMOTE+ET,  F1-Score (95.78%) | XGB + GridSearchCV,  Accuracy (99.03%) | Cleveland Dataset: QMBC+Anova+ PCA,  F1-Score (98.59%) | RF+iForest,  F1-Score (96.76%) |
| CVD Dataset: QMBC+Chi-Square+PCA,  F1-Score (99.92%) |
| HD Dataset: QMBC+Anova+ PCA,  F1-Score (98.42%) |
| *Model Validation* | - | - | - | - | - | Model trained by *SP*, and validated by *CC* |
| *Novelty of Approach* | Fast mutual information | HRFLM hybrid model | Hyperband, balancing | Near-zero prediction error, unique model | Novel QMBC, improved performance | Data privacy, Model training on encrypted data |
| *Entities and Responsibilities*  *Data* | Single authority | Single authority | Single entity | Collaborative validation | Collaborative preprocessing and training | Five entities: *CC*, *SP*, *CR*, *SU*, and *SYS*.  Centralized *CC* oversees dataset encryption, training, validation and *SYS* development. |
| *End-to-End Anonymity* | - | - | - | - | - | Anonymity maintained across encryption, verification, and training phases. |

By integrating several ML algorithms and tuning hyperparameters, our system demonstrates improved scoring measures including accuracy, sensitivity, specificity, and F1-score compared to models HRFLM [[8]](#_References) and Hyperband [[9]](#_References). Thus, the results in Table 8 are proof of the effectiveness and reliability of our proposed system, which is considered to be a big advancement in the field.

## Consistency Analysis

To understand how encryption affects model performance, the F1-scores of the original and encrypted datasets were compared across five different folds. The same dataset, preprocessing steps, and hyperparameters were used in both cases to ensure a fair and meaningful comparison. The results are summarized in Table 9.

Table 9: F1-score Comparison Between Original and Encrypted Datasets

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *1* | *2* | *3* | *4* | *5* |
| *Original Dataset (Δ)* | 96.61 | 97.09 | 98.22 | 98.38 | 97.73 |
| *Encrypted Dataset (Θ)* | 97.07 | 95.45 | 97.10 | 98.11 | 96.08 |
| *Difference*  *(xi)* | -0.47 | 1.64 | 1.12 | 0.27 | 1.65 |

Thus, the mathematical relationship can be expressed as:

This indicates that encryption introduces only a minor variation in model performance. The scores remain fairly close to the original dataset, showing that encryption preserves the model’s effectiveness while keeping the data secure.

# Limitations

For NB, the encryption may affect the independence assumption between features, leading to less accurate predictions. KNN, relying on distance metrics, may face difficulty in calculating meaningful distances between encrypted values, slowing down the prediction process. Similarly, SVM’s ability to find optimal hyperplanes can be hindered by encryption, as the distance and boundary calculations become less precise.

When using decision tree-based models such as RF or DT, the splitting process typically groups similar values into nodes, leading to faster calculations. However, when dealing with encrypted values, the relationship between data points becomes less straightforward. This can result in deeper trees, increasing computational complexity and slowing down the model.

Despite this limitation, the achieved F1-score for RF remains close to that of the original dataset, with only a minor difference in performance.

# Conclusion

This paper describes a heart disease prediction model that works under a comprehensive framework with data privacy and security. The approach embodies steps that span data collection, encryption, preprocessing, and model training with the use of NB, DT, RF, KNN, and SVM techniques, as well as their evaluation. Improvements in model performance and robustness have been examined through the inclusion of dimensionality reduction and anomaly detection techniques, all the while being considerate of sensitive information about patients.

The results of experiments indicate the performance of single models and their collections as well. Respectively, the distribution transformed by the DT gave it a fair average mean F1-score of 87.63%, while the SVM showed a mean F1-score of 82.55%. Even though both models performed well, their results were generally lower compared to those given by the models based on RF. Among all the RF-based models, the RF with iForest has shown the best performance with an average F1-score value of 96.76%, which exceeds the performances of all other configurations. Next, for RF only, the mean F1-score is 96.22%, while that for RF with PCA is 96.18%. Lastly, RF together with LSTM had the lowest mean F1-score of 93.17%. Such results provide proof that RF with iForest is the most powerful, taking the benefits of anomaly detection to augment predictive strength.

To maintain the privacy of patients, the proposed system encrypts patient data and allows the training of models on encrypted data without exposing patient information. The system provides a suitable balance between these two competing measures of data security and prediction accuracy.

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. Ponikowski P, Anker SD, AlHabib KF, Cowie MR, Force TL, Hu S, Jaarsma T, Krum H, Rastogi V, Rohde LE, Samal UC. Heart failure: preventing disease and death worldwide. ESC heart failure. 2014 Sep;1(1):4-25.

<https://doi.org/10.1002/ehf2.12005>

1. 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.

<https://doi.org/10.1007/s12652-018-0724-4>

1. 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.

<https://doi.org/10.1161/CIR.0b013e31820a55f5>

1. Al-Shayea QK. Artificial neural networks in medical diagnosis. International Journal of Computer Science Issues. 2011 Mar 1;8(2):150-4.
2. Wang S, Summers RM. Machine learning and radiology. Medical image analysis. 2012 Jul 1;16(5):933-51.

<https://doi.org/10.1016/j.media.2012.02.005>

1. Ghwanmeh S, Mohammad A, Al-Ibrahim A. Innovative artificial neural networks-based decision support system for heart diseases diagnosis.
2. 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.

<https://doi.org/10.1109/ACCESS.2020.3001149>

1. Mohan S, Thirumalai C, Srivastava G. Effective heart disease prediction using hybrid machine learning techniques. IEEE access. 2019 Jun 19;7:81542-54.

<https://doi.org/10.1109/ACCESS.2019.2923707>

1. 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.

<https://doi.org/10.1109/ACCESS.2022.3191669>

1. 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.

<https://doi.org/10.1109/CCWC57344.2023.10099294>

1. 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.

<https://doi.org/10.1109/I2CT45611.2019.9033683>

1. 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.

<https://doi.org/10.1109/ACCESS.2022.3165792>

1. 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.

<https://doi.org/10.1109/ICACCCN51052.2020.9362842>

1. 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.

<https://doi.org/10.1109/ICICCS53718.2022.9788408>

1. 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.

<https://doi.org/10.1109/ACCESS.2023.3289584>

1. Haraty RA, Otrok H, El-Kassar AN. A comparitive study of elgamal based cryptographic algorithms. InInternational Conference on Enterprise Information Systems 2004 Apr 14 (Vol. 4, pp. 79-84). SCITEPRESS.
2. Zhang H. The optimality of naive Bayes. Aa. 2004 Jan 1;1(2):3.
3. 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.

<https://doi.org/10.1109/ICSMC.1998.725047>

1. Pal M. Random forest classifier for remote sensing classification. International journal of remote sensing. 2005 Jan 1;26(1):217-22.

<https://doi.org/10.1080/01431160412331269698>

1. 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.

<https://doi.org/10.1109/TNNLS.2017.2673241>

1. Jakkula V. Tutorial on support vector machine (svm). School of EECS, Washington State University. 2006;37(2.5):3.
2. Wold S, Esbensen K, Geladi P. Principal component analysis. Chemometrics and intelligent laboratory systems. 1987 Aug 1;2(1-3):37-52.

<https://doi.org/10.1016/0169-7439(87)80084-9>

1. Liu FT, Ting KM, Zhou ZH. Isolation forest. In2008 eighth ieee international conference on data mining 2008 Dec 15 (pp. 413-422). IEEE.

<https://doi.org/10.1109/ICDM.2008.17>

1. Schmidhuber J, Hochreiter S. Long short-term memory. Neural Comput. 1997 Nov 15;9(8):1735-80.

<https://doi.org/10.1162/neco.1997.9.8.1735>

1. 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).
2. Anguita D, Ghelardoni L, Ghio A, Oneto L, Ridella S. The'K'in K-fold Cross Validation. InESANN 2012 Apr 25 (Vol. 102, pp. 441-446).
3. Abhishek14398. Heart Disease Classification. Kaggle. Available at:

<https://www.kaggle.com/datasets/abhishek14398/heart-disease-classification>

1. Fedesoriano. Heart Failure Prediction. Kaggle. Available at:

<https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction>

1. Jocelyndumlao. Cardiovascular Disease Dataset. Kaggle. Available at:

<https://www.kaggle.com/datasets/jocelyndumlao/cardiovascular-disease-dataset>

1. Rishidamarla. Heart Disease Prediction. Kaggle. Available at:

<https://www.kaggle.com/datasets/rishidamarla/heart-disease-prediction>

1. Redwankarimsony. Heart Disease Data. Kaggle. Available at:

<https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data>

1. Ineubytes. Heart Disease Dataset. Kaggle. Available at:

<https://www.kaggle.com/datasets/ineubytes/heart-disease-dataset>

1. Johnsmith88. Heart Disease Dataset. Kaggle. Available at:

<https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset>