End-to-End Secure Heart Disease Prediction over Encrypted Medical Data

Abstract

This research presents a privacy-first framework for heart disease prediction that safeguards sensitive medical data throughout its entire lifecycle. Leveraging ElGamal encryption, the system enables machine learning models to be trained and evaluated directly on encrypted data, eliminating the need to expose raw patient records at any stage. A range of classification algorithms—including Naïve Bayes, Decision Trees, Random Forests, K-Nearest Neighbors, Support Vector Machines, and LSTM—are evaluated in combination with techniques like Principal Component Analysis for dimensionality reduction and Isolation Forests for anomaly detection. After careful hyperparameter tuning, the best-performing configuration—Random Forest paired with Isolation Forest—achieves a robust F1-score of 96.76%. To further enhance security, all communications between users and the platform are encrypted using AES, with keys exchanged via RSA and verified through digital signatures. This ensures end-to-end confidentiality, integrity, and authenticity. By combining encrypted data processing, secure communication protocols, and advanced machine learning, the proposed framework offers accurate predictions without compromising medical data privacy—marking a significant step toward secure, privacy-conscious healthcare analytics.

Keywords Medical data privacy · heart disease prediction · ElGamal encryption · machine learning · secure communication

1. Introduction

Heart disease, one of the major public health challenges afflicting nearly 26 million people all over the world [[1]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References), remains the leading cause of death across the globe and is likely to escalate in prevalence in the future due to the growing populations of aged individuals [[2]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_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]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References). However, innovations in diagnostic tools and medical management have not improved the situation. Patients, especially from rural areas, cannot expect to have such access. Therefore, they rely on intuition from physicians to seek recommendations and treatment early [[4]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_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]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_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, maintaining data privacy and verifying the results pose concerns about building strong models.

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

The proposed hybrid method 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 proposed approach is evaluated by viewing the effect of variation of training set size on the performance of the model. Again, it distinguishes itself from existing methods by multiple features (*i*. *e*. PCA and iForest), temporal dynamics (*i*. *e*. LSTM), and ensemble power (*i*. *e*. 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 improve the outcomes through fast identification and interventions while maintaining the data integrity. Notably, the RF model with iForest achieved the best F1-score of 96.76%. Additionally, AES and RSA encryption methods, coupled with direct RSA signing, ensured robust data security, achieving efficient communication with encrypted data sizes of 36 bytes and 77 bytes for AES-encrypted input and output, respectively.

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 patients' 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.

Section [2](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Literature_Review) reviews existing approaches to secure machine learning and their limitations. Section [3](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Preliminaries) provides the necessary background on cryptographic tools and machine learning techniques. Section [4](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_System_Model_and) outlines the design of the secure system, explaining how it handles data encryption, privacy, and model training. Section [5](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Experimental_Analysis) presents and analyzes experimental results, including performance, accuracy, and efficiency of various models. Section [6](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Security_Threats_and) discusses security threats and implemented countermeasures. Section [7](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Limitations) acknowledges the practical limitations encountered during implementation and Section [8](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Conclusion) concludes the paper.

1. Literature Review

Heart disease prediction is a primary area where much research 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 diagnosing heart diseases [[7]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_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 has made their performance poor. However, the study did not consider data confidentiality or system-level data protection—key concerns when dealing with sensitive medical records.

Another study proposed an HRFLM model that hybridizes RF and the Linear Model to provide improved heart disease classification [[8]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References). Here, the experimental results show accuracy much more elevated than traditional methods, thus proving the superiority of this method. Still, the approach mainly focuses on predictive performance without implementing any safeguards for patient data privacy or secure communication within the system.

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]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_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 the 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]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References). The top scorer was scored by XGB with 91.30% accuracy and an 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]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References). Despite working with large-scale datasets, none of these models address data integrity verification or encryption during training—both of which are critical when working with hospital systems.

Subsequent studies compared LR, KNN, SVM, and XGB classifiers, optimized with Grid Search Cross-Validation (GridSearchCV) [[12-14]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References). RF consistently demonstrated superior performance, with accuracy 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]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References). Nevertheless, these improvements are model-centric, with no integration of cryptographic methods or secure user-system interactions.

While these studies offer valuable insights into heart disease prediction using various machine learning models and feature selection techniques, they all share a significant limitation: the lack of attention to data security, privacy protection, and secure system design. In contrast, this research presents a privacy-preserving framework that trains models directly on encrypted data and ensures secure interactions between users and the system. The approach combines secure data transmission, integrity verification, and user-level interpretability—addressing both predictive performance and the often-overlooked need for protecting sensitive medical information.

1. Preliminaries

To perform the evaluation, *i*.*e*. for ensuring data security, ElGamal cryptosystem is used and for heart disease prediction, different ML techniques and feature selection methods are used in this study. They are discussed as follows.

* 1. ElGamal Cryptosystem

ElGamal is a public-key encryption scheme based on the discrete log problem. Using a group *G* of prime order *p* and generator *g*, a private key *a* is picked from , and the public key is . A message *m ∈ G* is encrypted as using random *k*. Decryption is done via [[16]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. AES Cryptosystem

AES is a fast symmetric encryption method using 128-bit blocks and key sizes of 128, 192, or 256 bits. It applies rounds of SubBytes, ShiftRows, MixColumns, and AddRoundKey, with decryption reversing the process [[17]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. RSA Cryptosystem

RSA is a public-key encryption method built on number theory. It generates two large primes, *p* and *q*, then computes and . A public key *e* is chosen such that , and the private key *d* satisfies . Encryption is , and decryption is [[17]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. RSA Digital Signature

A digital signature helps verify that a message is authentic and unchanged. In RSA, the sender signs a message using their private key, and the recipient verifies it with the sender’s public key. To sign a message *m*, the sender first hashes it to get *H(m)*, then encrypts the hash with their private key:

where *d* is the private key.

To verify, the recipient decrypts *S* using the sender’s public key and checks if it matches their own computed *H(m)*. If they match, the signature is valid—proving the message’s integrity and confirming the sender’s identity.

* 1. Naïve Bayes Classifier (NB)

NB is a probabilistic model that assumes features are independent given the class. It works well in practice, and can still perform optimally even if this independence doesn’t hold—provided dependencies balance across classes [[18]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. Decision Tree (DT)

DT is a tree-structured model that splits data based on feature values. Internal nodes represent decisions; leaves give outcomes. It’s easy to interpret but can grow complex on large datasets [[19]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. Random Forest Classifier (RF)

RF creates multiple decision trees, each trained on random data subsets. Each tree votes, and the majority vote gives the final prediction. This ensemble method reduces overfitting and improves accuracy compared to a single tree [[20]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. K-Nearest Neighbors (KNN)

KNN assigns a query point to the most frequent class or the mean of the k nearest data points in the training set, making it effective for both classification and regression tasks [[21]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. Support Vector Machine (SVM)

SVM finds the optimal hyperplane that maximizes the margin between classes, with support vectors influencing its placement. For non-linearly separable data, SVM uses the kernel trick to map it into a higher-dimensional space [[22]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. Principal Component Analysis (PCA)

PCA reduces data dimensions to recognize patterns by transforming it into a new coordinate system. It involves standardization, covariance matrix computation, eigenvalue decomposition, and projection. PCA is used for dimensionality reduction, feature extraction, noise removal, and outlier detection [[23]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. Isolation Forest (iForest)

iForest isolates anomalies using random trees, where outliers have shorter path lengths. It handles high-dimensional data well and is robust to noise and outliers [[24]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. Long Short-Term Memory (LSTM)

LSTM is a type of RNN built to handle long-term dependencies by using memory cells with input, output, and forget gates. These gates control the flow of information, making LSTM effective for sequential data [[25]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. Train-Test Split

Train-test split divides data into training and test sets (e.g., 70:30 or 80:20) to evaluate ML models. The training set fits the model; the test set checks performance. Proper split ratios help avoid misleading results [[26]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. K-Fold Cross-Validation

K-fold cross-validation splits data into *k* parts. Each time, *k–1* folds train the model, and the remaining one tests it. This repeats *k* times so every point is used for both training and testing [[27]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

* 1. Performance Evaluation Metrics

A confusion matrix, as detailed in [[15]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References), is a table that summarizes the performance of a classification model by comparing its predicted values with actual values. Predictions are grouped into 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.

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

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

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

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

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

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

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

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

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

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

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

*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.

|  |
| --- |
|  |
| Fig. 1 Proposed Heart Disease Prediction System |

1. System Model and Functional Roles

This section provides a detailed architectural blueprint of the system, as in Fig. 1. It defines the system's components, their interactions, and respective functions. Additionally, it outlines the sequential flow of overall activities.

* 1. Roles of each entity

The proposed system comprises five primary entities namely Central Coordinator (*CC*), Cryptographer (*CR*), Service Provider (*SP*), System (*SYS*) and System User (*SU*), as illustrated in Fig. 2.

*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 a public-private key pair, shares the public key with *CR*s, and securely stores the private key. Once the dataset is encrypted by the *CR*s, the *CC* validates its integrity. If the dataset is deemed valid, the *CC* forwards the encrypted dataset 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 *SU*s to use.

*CR* − After receiving the public key from the *CC*, *CR*s generate a unique ephemeral key for each column of the dataset. They use these keys to encrypt the data securely. The encrypted dataset is then sent back to the *CC* for validation.

|  |
| --- |
| Fig. 2 Interactions among entities |

*SP* − The *SP* preprocesses the encrypted dataset received from *CC* and trains several ML models. After evaluating these models using metrics such as F1-score, the *SP* selects the best-performing model and shares it with *CC*.

*SU* − The *SU* generates an RSA key pair and shares the public key with the *SYS*. To send input, the *SU* encrypts it using AES, encrypts the AES key using *SYS*’s public key, and signs both using their private key. The *SU* then sends the encrypted input, encrypted key, and digital signature to the *SYS*. After receiving the prediction, the *SU* verifies the signature using *SYS*’s public key, decrypts the AES key with their private key, and finally retrieves the original output by decrypting it with the recovered AES key.

*SYS* − The *SYS* generates another RSA key pair when a new *SU* account is created and shares the public key with the *SU*. After receiving the AES-encrypted input, encrypted AES key, and digital signature from the *SU*, *SYS* first verifies the signature using the *SU*’s public key. It then decrypts the AES key using its private key and uses that AES key to decrypt the input. The decrypted input is then column-wise encrypted with public keys, preprocessed, and passed through the trained model for prediction. The result is encrypted with AES, the AES key is encrypted using the *SU*’s public key, and both are signed using *SYS*’s private key before being sent to the *SU*.

* 1. Table of Notations

This section provides a list of notations used in this system, as shown in Table 1.

Table 1. Notations 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 |
| *κSYS* | *SYS*’s public key |
| *ρSYS* | *SYS*’s private key |
| *κSU* | *SU*’s public key |
| *ρSU* | *SU*’s private key |
| *χ* | AES key |

* 1. Individual stages

The proposed framework consists of ten distinct stages. They are: (1) Data Collection, (2) Key Generation, (3) Data Encryption, (4) Data Decryption and Verification, (5) Data Preprocessing, (6) Model Training, (7) Model Optimization, (8) Final Model selection, (9) Model verification, and (10) System Development. The stages are detailed below with the notations from Table 1.

4.3.1 Data Collection

The *CC* collects anonymized heart disease datasets from various hospitals, ensuring all personal information, such as names, addresses, and identifiers like medical record numbers or dates, is removed beforehand. This is done under strict ethical approvals and formal data-sharing agreements, in full compliance with data protection regulations. For this research, data was gathered from seven publicly available Kaggle repositories [[28-34]](file:///C:\\Users\\USER\\Downloads\\paper_neural%2013.4.25.docx" \l "_References). The datasets, denoted as *ω*1​, *ω*2, ..., *ω*7, contain several inconsistencies. These include irrelevancy for this work, numerical and categorical values, and some 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 =* .

*Δ* has the 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 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 |

Fig. 3 provides a visual representation of the distribution of key attributes within the dataset, while Fig. 4 highlights the relationships between attributes using a correlation heatmap.

4.3.2 Key Generation

Medical data is sensitive, and *CC* cannot fully trust *SP* with it. To mitigate this risk, *SP* must train a model that works on encrypted data. Therefore, *CC* provides encrypted data to *SP*, ensuring data remains protected throughout the process.

To achieve this, *CC* generates a pair of public and private keys (*κ* and *ρ*) using the ElGamal cryptosystem, as described in Section [3.1](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_ElGamal_Cryptosystem). The key generation process is detailed in Algorithm 1.

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

Since *CC* lacks the time and processing power to encrypt the entire dataset, *CR* handles encryption. However, giving a *CR* full access to the dataset poses a risk of leakage. To mitigate this, *CC* keeps one column for himself while splitting the remaining dataset into two parts and distributing them between two *CR*s. Specifically, one part of the dataset, along with the public key, is sent to *CR*1, while the other part, along with the public key, goes to *CR*2. With limited resources, *CC* selects the column with the smallest amount of data for direct encryption.

4.3.3 Data Encryption

*CC* and the two *CR*s secure their respective parts of the dataset, which together contain *d* feature vectors (*ξ1, ξ2, ..., ξd*) and their corresponding label *λ*. Each feature *ξi* is encrypted using the encryption function *E(κ, ξi)* with the public key *κ*, provided by *CC*, resulting in its encrypted form *Θi*. For each feature, *CC* and the *CR*s generate new ephemeral keys during encryption. The encryption process follows the steps outlined in Algorithm 2.

|  |
| --- |
| Algorithm 2 Data Encryption |
| 1. **Input:** Public key *κ,* Features *ξ* 2. **Output:** Encrypted Column *Θi* 3. **function** *E(κ, ξ):* 4. *p, g, h* ← *κ['p'], κ['g'], κ['h']* 5. *ki* ← **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 × hki 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** |

For the labels *λ*, *CR* applies a simple binary encoding, converting 'Yes' to 1 and 'No' to 0, producing *λ′*. Once encryption is complete, *CC* combines all encrypted parts into a single dataset, *Θ*, and verifies its integrity. The *CC* verifies the dataset before sending it to the *SP*, as detecting issues after model training would require repeating the entire process.

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect.  Fig. 4 Correlation Heatmap of Features for Heart Disease Prediction |

4.3.4 Data Decryption and Verification

The *CC* uses the private key *ρ* and the decryption function *D(ρ, Θi)* to decrypt *Θi*, obtaining the decrypted features *Υi* as described in section [3.1](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_ElGamal_Cryptosystem), following the steps in Algorithm 3. The *CC* then compares *Υi*​ with the original *Δi* to ensure no data was altered during encryption or transmission by the *CR*. If all *Δi* values match their corresponding *Υi* values, the *CC* forwards *Θ* to the *SP* for model training.

|  |
| --- |
| Algorithm 3 Data Verification |
| 1. **Input:** Private key*ρ,* Encrypted Column *Θi* 2. **Output:** "Valid" or "Invalid" 3. **function** *D(ρ, Θi):* 4. *p, g, a ← ρ['p'], ρ['g'], ρ['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** |

4.3.5 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:

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

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*​ as below:

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

where, Mean,

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

and, Standard Deviation,

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

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

4.3.6 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](file:///C:\\Users\\USER\\Downloads\\paper_neural%2013.4.25.docx" \l "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.

4.3.7 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. In this work, *SP* also uses three advanced techniques: RF, 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.

4.3.8 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.

4.3.9 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.*

4.3.10 System Development

*CC* develops a user-friendly system, *SYS*, to allow secure interaction with *SU*s. When an *SU* creates an account, *SYS* first generates an RSA key pair—its public key *κSYS*​ and private key *ρSYS*​. *κSYS*​ is shared with the SU, while *ρSYS*​ remains securely stored on *SYS*’s side. Meanwhile, the *SU* also generates their own RSA key pair—public key *κSU*​ and private key *ρSU*​—keeping *ρSU* secret and sharing *κSU* with *SYS*.

Since this public key exchange step is the most vulnerable to man-in-the-middle (MITM) attacks, it is crucial that both *κSYS*​ and *κSU*​ are exchanged through a secure and trusted channel—ideally a trusted third party. These keys are generated and exchanged only once, and once they’re securely shared, the communication channel is protected against any future MITM attempts.

When submitting input data, the *SU* first encrypts it using AES with a randomly generated symmetric key *χ*. The AES key *χ* is then encrypted with *κSYS*​. To ensure data integrity and authenticity, the SU generates a digital signature over the AES-encrypted input and the encrypted *χ*, signing them with *ρSU*​. This package—including the AES-encrypted input, the encrypted *χ*, and the digital signature—is sent to *SYS*.

Upon receiving the data, *SYS* first verifies the digital signature using *κSU*​. If valid, *SYS* decrypts the *χ* using *ρSYS*​, and uses *χ* to decrypt the *SU*'s input.

Each decrypted feature value is re-encrypted using the ephemeral public keys that were already generated by CC and CRs—one unique key for each feature. These re-encrypted values are then preprocessed using *δ* and passed into the *ηγ*​, which returns a prediction.

The prediction result includes:

1. A decoded label (e.g., "Yes" or "No"),
2. The percentage likelihoods for both outcomes,
3. A detailed contribution score from each input feature, indicating its impact on the decision.

This information is then encrypted again using AES with a new *χ*, and the AES key is encrypted with *κSU*​. *SYS* signs both the encrypted result and the encrypted *χ* with *ρSYS*​, then sends the signed package to the *SU*.

The *SU* verifies the signature using *κSYS*​, decrypts *χ* using *ρSU*​, and finally decrypts the output using the recovered *χ*. *SYS* also enables the *SU* to visualize the results through local graphical summaries—particularly useful for understanding feature contributions and outcome probabilities.

1. Experimental Analysis  
   1. Experimental Setup

All experiments were conducted on a system running a 64-bit operating system with an Intel Core i5-8265U processor and 8 GB of RAM. The development and testing of the cryptographic components—including AES, RSA, ElGamal, and RSA digital signatures—were carried out alongside the implementation of ML models for heart disease prediction. Visual Studio Code and Jupyter Notebook (.ipynb) served as the primary development environments, utilizing relevant libraries to support cryptographic functions and ML workflows.

* 1. Consistency Analysis

To understand how encryption affects model performance, the F1-scores of the original and encrypted datasets were compared across five different folds, summarized in Table 3. The same dataset, preprocessing steps, and hyperparameters were used in both cases to ensure a fair and meaningful comparison.

Table 3. 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 on the encrypted dataset remain fairly close to the scores on the original dataset, showing that encryption preserves the model’s effectiveness while keeping the data secure.

* 1. Experimental Results and Comparative Analyses

This section presents results regarding the time needed for key generation, encryption, and model training. For each of these tasks, three attempts were made to ensure consistency and account for any potential variations in performance.

At first, Table 4 shows the key generation times, which range from 0.02 to 0.04 seconds, with both private and public keys consistently sized at 0.87 KB.

Table 4. Time Required for Key Generation

|  |  |  |  |
| --- | --- | --- | --- |
| *Attempt* | *Generation Time*  *(sec)* | *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 5 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. Time Required for Encryption and Decryption

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

Table 6 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. Time Required for Model Training

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

Table 7 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 7. Time for Training RF with Different Methods

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

The performance of ML models was evaluated using several key metrics. As discussed in Section [3.15](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Performance_Evaluation_Metrics), a confusion matrix was used to provide a detailed comparison between the model’s predictions and the actual ground truth values. This tabular representation helps assess accuracy, precision, recall, and other important evaluation metrics.

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.

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

Model robustness and generalization were assessed by employing various train-test splits, as described in Section [3.13](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_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.

Fig. 5 illustrates how the performance of different ML models on test data varies with changes in the train-test split, while Fig. 6 summarizes their average performance metrics across these splits.

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

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%), an impressive AUC (95.14%) etc., RF proves to be highly reliable for classifying the dataset and delivering top-tier performance.

SVM also performs well, particularly in metrics like recall, PPV, NPV, and F1-score as in Fig. 6. Although its accuracy, specificity, and AUC are slightly behind RF, SVM remains a strong contender, offering robust and reliable classification results. Also, DT provides a balanced performance across metrics, which indicates DT as a dependable model with consistently good classification capabilities. Again, 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.

Here, 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.

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 this context. 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 this comparative analysis.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 8. 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 |

Table 8 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 [3.14](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_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.

Table 9 summarizes the sizes of various components exchanged between the *SU* and *SYS* during secure communication. RSA key generation involves two 4-byte primes, producing an 8-byte modulus. The AES key, a single byte representing values from 1 to 255, is encrypted using RSA, resulting in an 8-byte ciphertext. The *SU* sends an encrypted AES key (8 bytes), AES-encrypted input (28 bytes), and a digital signature (104 bytes). Similarly, the *SYS* responds with its own encrypted AES key (8 bytes), AES-encrypted output (69 bytes), and a digital signature (216 bytes).

Table 9. Component Sizes in SU–SYS Communication

|  |  |  |
| --- | --- | --- |
| *Component* | *Size (in bits)* | *Size (in bytes)* |
| RSA p, q | 32 each | 4 each |
| RSA modulus (n) | 64 | 8 |
| AES key | 8 | 1 |
| *SU*'s RSA-encrypted data | 28 × 64 | 224 |
| *SU*'s AES-encrypted data | 28 characters | 28 |
| *SYS*'s RSA-encrypted data | 69 × 64 | 552 |
| *SYS*'s AES-encrypted data | 69 characters | 69 |
| RSA-encrypted AES key | 64 | 8 |
| *SU*'s RSA digital signature | 13 × 64 | 104 |
| *SYS*'s RSA digital signature | 27 × 64 | 216 |

Both RSA and AES encryption methods were implemented and compared. *SU*’s RSA-encrypted input requires 224 bytes, while AES-encrypted input with an RSA-encrypted AES key needs only 36 bytes (28 + 8). Similarly, *SYS*’s RSA-encrypted output occupies 552 bytes, compared to just 77 bytes (69 + 8) using AES. Given this significant difference in space efficiency, the AES-based approach was chosen for encrypting data, ensuring lightweight and faster communication without compromising security.

For digital signatures, hashing was intentionally avoided. Since the encrypted data itself is not large, introducing hashing would have added unnecessary computational steps. Direct RSA signing proved sufficient and more efficient for the intended context.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table 10. Comparative Analysis of Proposed System with Existing Methods | | | | | | |
| *Attributes* | [*[7]*](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References) | [*[8]*](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References) | [*[9]*](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References) | [*[12]*](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References) | [*[15]*](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References) | *Proposed System* |
| *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 |
| *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*s |
| *Integrity of Dataset* | - | - | - | - | - | Encrypted dataset verified by *CC* |
| *Model trained on* | Raw Data | Raw Data | Raw Data | Raw Data | Raw Data | 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 | Missing values handling, standardization, scaling, PCA, iForest, 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 | 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* | - | - | - | - | - | Validated by *CC* after trained by *SP* |
| *Entities and Responsibilities of Data* | Single authority | Single authority | Single entity | Single entity | Single entity | Five entities: *CC*, *SP*, *CR*, *SU*, and *SYS.*  *CC* oversees dataset encryption, training, validation and *SYS* development |
| *User Data Security* | - | - | - | - | - | End-to-end encryption with digital signature |
| *Input Authenticity* | - | - | - | - | - | Verified using *SU*'s digital signature |
| *Output Confidentiality* | - | - | - | - | - | Encrypted result delivery with digital signature |
| *End-to-End Anonymity* | - | - | - | - | - | Maintained across encryption, model training, and *SU*–*SYS* communication |

The comparative analysis between the proposed system and the existing methodologies is described in Table 10. This table presents a systematic evaluation of various attributes, including characteristics of datasets, data security and integrity, preprocessing techniques, feature engineering approaches, ML algorithms, hyperparameter tuning strategies, evaluation metrics, model performance, validation processes, entity responsibilities, user data security, input authenticity, output confidentiality, and end-to-end anonymity. The proposed system utilizes a combination of seven Kaggle datasets, which significantly enhances both the diversity and robustness of the data, setting it apart from prior studies that rely on smaller, more limited datasets [[7, 8]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

Extensive preprocessing, including standardization, scaling, and feature selection, has differentiated the proposed method from the others that utilize simple methods or techniques such as SMOTE [[9, 12]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References). By incorporating multiple ML algorithms and fine-tuning hyperparameters, the system demonstrates superior performance in F1-score when compared to models like HRFLM [[8]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References) and Hyperband [[9]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References).

While the F1-score achieved by the proposed system may not match the top-performing models in [[12]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References) and [[15]](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_References), it still represents a commendable result, particularly given the added security measures integrated into the system. Unlike many existing methodologies, which overlook robust security mechanisms, the proposed system ensures end-to-end data protection. Key features include end-to-end encryption, digital signatures for input and output validation, and the maintenance of user anonymity throughout the process. These security measures provide an additional layer of trust and confidentiality, offering a unique advantage over other methods.

Ultimately, the system's strong security framework, alongside its competitive performance, marks a significant advancement in the field.

1. Security Threats and Solutions

In the 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 the methodology to ensure data integrity and system security.

* 1. Predictable Encoding

*Threat: If the CR applies traditional encoding techniques—such as label encoding, one-hot encoding, or ordinal encoding—to categorical features, the SP might infer the original values by analyzing patterns in the encoded data. For example:*

1. *With label encoding, 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 alphabetical order.*
2. *With one-hot encoding, the number of columns directly reveals how many categories exist, and the position of 1s may leak ordering or structure.*
3. *With ordinal encoding, categorical values such as "Yes" = 1 and "No" = 0 may suggest a meaningful hierarchy. The SP could infer that "Yes" is ranked higher than "No", and the original values could be easily derived based on the numeric relationship.*

*These patterns can lead to unintended information leakage, potentially exposing sensitive attributes.*

*Solution:* By encrypting categorical values using the ElGamal encryption scheme, as described in Section [4.3.3](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Individual_stages), the outputs become random and non-sequential, removing any patterns, fixed positions, or order. This ensures that the *SP* cannot infer any relationship between encoded values or their original categories. The encryption completely eliminates the potential for identifying values or making educated guesses about the data, thus providing strong privacy protection and ensuring categorical attributes remain secure against inference attacks.

* 1. Data Confidentiality Breach

*Threat: A 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:* To prevent any single *CR* from gaining full access to the dataset, the *CC* splits it into two parts and sends them to two separate *CR*s (*CR*1 and *CR*2), along with the public key for encryption, as described in Section [4.3.2](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Individual_stages). The *CC* retains two columns—one feature column and the label column—for direct encryption. This setup ensures that even if both *CR*s were to collaborate, they would still not have complete access to the dataset. Once encryption is completed, the *CC* verifies the integrity of the encrypted data using its private key, as detailed in Section [4.3.3](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Individual_stages), ensuring that the *CR*s have not tampered with the data. The verified dataset is then forwarded to the *SP* for model training, maintaining its confidentiality and integrity throughout the process.

* 1. 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 generating new ephemeral keys for each feature, this approach ensures that no two features share the same encryption key, as described in Section [4.3.3](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Individual_stages). This means that even if two features have identical values, their encrypted representations will still differ, preventing any attacker from identifying correlations between them. As a result, each feature remains securely encrypted on its own, protecting its integrity even if one feature is compromised.

* 1. Model Manipulation

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

*Solution:* In Section [4.3.8](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_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.

* 1. Corruption Risk

*Threat: Corruption of CC, CRs, 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 fails to function properly, *SU*s are likely to reject it, leading to a loss for the *CC*. To ensure data integrity, the *CC* decrypts the data in Section [4.3.3](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Individual_stages) and verifies the models and data in Section [4.3.8](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Individual_stages). If corruption is detected, *CC* can take corrective actions, such as penalizing or replacing the corrupted parties (*CR*s or *SP*), ensuring the system remains functional and trustworthy.

* 1. Data Exposure Risk

*Threat: When the SU submits input and receives the result from the SYS, an attacker might intercept the communication and try to uncover the original data or prediction result. This can lead to a serious breach of privacy.*

*Solution:* As outlined in Section [4.3.10](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Individual_stages), all data exchanged between the *SU* and *SYS* is protected using a layered encryption approach. The *SU* encrypts their input with AES using a randomly generated key, then encrypts that key with the *SYS*'s public key. The *SYS* follows the same process when sending back results. This ensures that even if someone intercepts the message, they cannot access the original content, keeping the data secure from end to end.

* 1. Data Tampering and Forgery

*Threat: There’s also a risk that an attacker could alter the data or results in transit—injecting false inputs or tampering with outputs—leading to incorrect predictions or loss of trust in the system.*

*Solution:* To prevent tampering and ensure authenticity, both the *SU* and *SYS* sign their respective messages using their private keys, as detailed in Section [4.3.10](file:///C:\Users\USER\Downloads\paper_neural%2013.4.25.docx#_Individual_stages). Each package includes the encrypted content and a digital signature that can be verified by the recipient using the sender’s public key. This guarantees that the data hasn’t been modified and truly came from the claimed source, safeguarding the system from forgery and manipulation.

1. Limitations

While the proposed system effectively secures sensitive data and supports a range of ML models, a few limitations were observed during implementation.

The system uses a static ephemeral key for each feature, meaning the same key is applied to all instances of a feature. While this provides strong encryption, it may be a limitation if an attacker exploits patterns from using the same key across multiple data points.

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.

1. Conclusion

This work 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. To maintain the privacy of patients, this system encrypts patient data and allows the training of models on encrypted data. The system provides a suitable balance between the two competing measures of data security and prediction accuracy.

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.

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.

Additionally, the proposed system workflow ensures secure communication between the *CC* and *SU*s. By using RSA and AES encryption, along with digital signatures for data integrity, the system guarantees the protection of patient data throughout the prediction process. This integration of secure key management and encryption techniques further strengthens the framework, maintaining privacy while delivering accurate predictions.

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.
2. 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.
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. Al-Shayea QK. Artificial neural networks in medical diagnosis. International Journal of Computer Science Issues. 2011 Mar 1;8(2):150-4.
5. Wang S, Summers RM. Machine learning and radiology. Medical image analysis. 2012 Jul 1;16(5):933-51.
6. Ghwanmeh S, Mohammad A, Al-Ibrahim A. Innovative artificial neural networks-based decision support system for heart diseases diagnosis.
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. 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.
17. Mahajan P, Sachdeva A. A study of encryption algorithms AES, DES and RSA for security. Global journal of computer science and technology. 2013 Dec;13(15):15-22.
18. Zhang H. The optimality of naive Bayes. Aa. 2004 Jan 1;1(2):3.
19. 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.
20. Pal M. Random forest classifier for remote sensing classification. International journal of remote sensing. 2005 Jan 1;26(1):217-22.
21. 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.
22. Jakkula V. Tutorial on support vector machine (svm). School of EECS, Washington State University. 2006;37(2.5):3.
23. Wold S, Esbensen K, Geladi P. Principal component analysis. Chemometrics and intelligent laboratory systems. 1987 Aug 1;2(1-3):37-52.
24. Liu FT, Ting KM, Zhou ZH. Isolation forest. In2008 eighth ieee international conference on data mining 2008 Dec 15 (pp. 413-422). IEEE.
25. Schmidhuber J, Hochreiter S. Long short-term memory. Neural Comput. 1997 Nov 15;9(8):1735-80.
26. 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).
27. 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).
28. [Online]. Available: https://www.kaggle.com/da­tasets/abhishek14398/heart-disease-classification
29. [Online]. Available: https://www.kaggle.com/da­tasets/fedesoriano/heart-failure-prediction
30. [Online]. Available: https://www.kaggle.com/da­tasets/jocelyndumlao/cardiovascular-disease-dataset
31. [Online]. Available: https://www.kaggle.com/da­tasets/rishidamarla/heart-disease-prediction
32. [Online]. Available: https://www.kaggle.com/da­tasets/redwankarimsony/heart-disease-data
33. [Online]. Available: https://www.kaggle.com/da­tasets/ineubytes/heart-disease-dataset
34. [Online]. Available: https://www.kaggle.com/da­tasets/johnsmith88/heart-disease-dataset