**End-to-End Secure Heart Disease Prediction with Verification over Encrypted Data**

Israt Tasnim Eshaa, Dola Dasa\*, Kazi Md. Rokibul Alama

aDepartment of Computer Science and Engineering (CSE),  
Khulna University of Engineering & Technology (KUET),  
Khulna-9203, Bangladesh.

**Email addresses:**

israttasnimesha1046@gmail.com (Israt Tasnim Esha)

dola.das@cse.kuet.ac.bd (Dola Das)

rokib@cse.kuet.ac.bd (Kazi Md. Rokibul Alam)

**ORCID iDs:**

Israt Tasnim Esha: [0009-0008-3936-1803]

Dola Das: [0000-0001-9001-4593]

Kazi Md. Rokibul Alam: [0000-0003-4895-7234]

**Acknowledgment:**  
The authors thank Kaggle and the respective dataset contributors for providing open access to the data used in this research.

\*Corresponding author: Dola Das

Email: dola.das@cse.kuet.ac.bd

Phone: +8801679502532

**Abstract**

This research presents a privacy-first framework for heart disease prediction that safeguards sensitive medical data throughout its entire lifecycle. To support encrypted processing, all numerical features are first categorized into discrete groups. Leveraging ElGamal encryption, the system allows machine learning models to be trained and evaluated directly on encrypted data, ensuring that raw patient records are never exposed. A range of classification algorithms—including Decision Trees, Extreme Gradient Boosting, and Random Forests —are evaluated alongside dimensionality reduction using Principal Component Analysis and anomaly detection via Isolation Forests. The combined use of Random Forest and Extreme Gradient Boosting achieves a robust F1-score of 94.78%. 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

# **Introduction**

Heart disease, one of the major public health challenges afflicting nearly 26 million people all over the world [[1]](#_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]](#_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). 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]](#_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, 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, and anomaly detection on encrypted data. More robust and accurate prediction models are established to this effect, integrating the Random Forest Classifier (RF), Principal Component Analysis (PCA), Isolation Forest (iForest), and Extreme Gradient Boosting (XGB). Various combinations—such as RF, RF with PCA, RF with iForest, RF with XGB, and RF with both iForest and XGB—are explored to enhance predictive performance and reliability across different perspectives.

To improve model compatibility and reduce value exposure, all numerical features are categorized into 3–4 discrete groups prior to processing. Ensuring data security is also a focus, with the 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 ML models and their combinations by employing the entire range of metrics like Accuracy, Specificity, Recall, Precision, 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) and ensemble power (i.e., RF and XGB). 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 XGB achieved the best F1-score of 94.78%. Additionally, AES and RSA encryption methods, coupled with direct RSA signing, ensured robust data security, achieving efficient communication with encrypted data sizes of 29 bytes and 75 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 presents the literature review, covering existing approaches to secure ML and their limitations. Section [3](#_Preliminaries) provides the necessary background on cryptographic tools and ML techniques. Section [4](#_System_Model_and) outlines the design of the secure system, explaining how it handles data encryption, privacy, and model training. Section 5 presents the experimental analysis, detailing the setup, obtained results, consistency evaluation, comparisons across encrypted data types, and benchmarking against existing approaches. Section [6](#_Security_Threats_and) discusses security threats and implemented countermeasures. Section [7](#_Limitations) acknowledges the practical limitations encountered during implementation. Section [8](#_Conclusion) concludes the paper.

# **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]](#_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), k-Nearest Neighbors (KNN), Artificial Neural Networks (ANN), Support Vector Machine (SVM), Naïve Bias Classifier (NB), and Decision Trees (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]](#_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]](#_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 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 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]](#_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 explored a range of classifiers and hyperparameter optimization techniques to enhance prediction outcomes. One such study applied K-modes clustering with the Huang initialization method to a Kaggle heart disease dataset consisting of 70,000 rows and 11 features [[12]](#_References). After categorizing data and removing outliers, classifiers such as DT, RF, MLP, and XGB were trained using GridSearchCV. Among them, the MLP model demonstrated the best performance, achieving an accuracy of 87.28%. However, this work primarily focuses on model accuracy without addressing encryption or secure data handling.

Another research used both the Cleveland and IEEE Dataport datasets to evaluate models including RF, KNN, LR, NB, GB, AB, and Soft Voting Ensemble Classifier (SVE) [[13]](#_References). The study optimized each model with GridSearchCV and measured accuracy, precision, recall, and F1-score. The SVE model achieved superior results, recording 93.44% accuracy on Cleveland and 95% on the IEEE dataset. Still, the study overlooks security aspects such as encrypted model training or system-level data integrity verification.

A more extensive study utilized data from four standard datasets—Cleveland, Hungary, Switzerland, and Long Beach V [[14]](#_References). Preprocessing included handling missing values, removing duplicates and outliers, and normalizing the data. A diverse set of models such as Extra Trees, RF, XGB, and CatBoost were trained with both GridSearchCV and RandomizedSearchCV. Extra Trees stood out with an accuracy of 98.15%, outperforming other configurations. Yet, despite thorough preprocessing and strong performance, the study does not incorporate any cryptographic safeguards or secure system interactions.

A more recent study introduced ensemble methods and a novel Quine McCluskey Binary Classifier (QMBC), accompanied by robust feature selection techniques such as Chi-Square and Analysis of Variance (ANOVA), which significantly boosted prediction performance [[15]](#_References). The study used multiple heart disease datasets, including Cleveland, CVD, and HD, applying preprocessing steps like data type conversion, outlier handling, SMOTE, and under-sampling. For the Cleveland dataset, the combination of QMBC, ANOVA, and PCA achieved an F1-score of 98.59%, while the same model with Chi-Square and PCA yielded an F1-score of 99.92% for the CVD dataset. On the HD dataset, QMBC with ANOVA and PCA attained an F1-score of 98.42%. These results underscore the model's capacity for near-zero prediction error across diverse datasets. Nevertheless, these enhancements remain limited to predictive accuracy, with no attention given to securing datasets or facilitating encrypted model training and secure communication—elements that are essential for medical applications.

While these studies offer valuable insights into heart disease prediction using various ML 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.

# **Preliminaries**

To ensure data security, the ElGamal cryptosystem is employed, while heart disease prediction relies on various ML models, dimensionality reduction (PCA), anomaly detection (iForest), train-test splitting, and standard evaluation metrics. These components are discussed as follows.

## **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 {1, …, *p*-1}, and the public key is *ga mod p*. A message *m* ∈ *G* is encrypted as *(c*1*, c*2*) = (gk mod p, m\*(ga)k mod p)* using random *k*. Decryption is done via *c*2*/(c*1*a) mod p* [[16]](#_References).

## **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]](#_References).

## **RSA Cryptosystem**

RSA is a public-key encryption method built on number theory. It generates two large primes, *p* and *q*, then computes *n=p×q* and *ϕ*=(*p-*1)(*q*-1). A public key *e* is chosen such that *gcd*(*e,ϕ*)=1, and the private key *d* satisfies *ed*=1 *mod ϕ*. Encryption is *c*=*me mod n*, and decryption is *m*=*cd mod n* [[17]](#_References).

## **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, *S*=*H*(*m*)*d mod n*, 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.

## **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 [[18]](#_References).

## **Extreme Gradient Boosting (XGB)**

XGB is a scalable tree boosting system designed for efficiency and performance. It uses a sparsity-aware algorithm and approximate split finding to handle large datasets. A regularized objective helps prevent overfitting, and out-of-core computation allows processing billions of examples [[19]](#_References).

## **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]](#_References).

## **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 [[21]](#_References).

## **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 [[22]](#_References).

## **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 [[23]](#_References).

## **Performance Evaluation Metrics**

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 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.

# **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.

## **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, and converting numerical values into categorized form. *CC* generates a public-private key pair, shares the public key with *CR*s, and securely stores the private key. The *CC* also selects one column for direct encryption, generates an ephemeral key, and encrypts it using ElGamal with the public key. Additionally, *CC* encodes the label column.Once the encrypted columns are received from the *CR*s, the *CC* combines the encrypted data and then validates the integrity of the complete dataset. If the dataset is deemed valid, the *CC* forwards the encrypted dataset to the *SP* for model training. After receiving the best-performing trained model, corresponding test dataset, preprocessing scaler, and F1-score from the *SP*, the *CC* tests the model on its own encrypted dataset and compares the F1-score with that reported by the *SP*. If the model passes validation, the *CC* develops the *SYS* using that model and the 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.

*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.

|  |
| --- |
|  |
| **Fig. 1** Proposed Heart Disease Prediction System |
| **Fig. 2** Interactions among entities |

*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*.

## **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 | ML models: DT, XGB, RF |
| *η*α | Best of {*η*1, *η*2, *η*3} |
| *ηα*1*, ηα*2*, ηα*3*, ηα*4, *ηα*5 | RF, RF+PCA, RF+iForest, RF+XGB, RF+iForest+XGB |
| *ηβ* | Best performing model |
| *Π*1*, Π*2*, Π*3*,Π*4*, Π*5*,Π*6*,Π*7*,Π*8 | Confusion Matrix, Accuracy, Specificity, Recall, Precision, NPV, F1-score, AUC |
| *τ* | Train-test split ratios (0.1:0.9 to 0.9:0.1) |
| *δ* | Preprocessing transformer |
| *(κSYS, ρSYS)* | *SYS*’s public-private key pair |
| *(κSU, ρSU)* | *SU*’s public-private key pair |
| *χ* | AES key |

## **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. These stages are detailed below using 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 [[24-30]](#_References) originating from multiple sources, including Cleveland, Hungary, Switzerland, Long Beach V, UCI, Stalog (Heart) Data Set, and a multispecialty hospital in India, comprising a total of 5,461 records. 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*. Additionally, some datasets share common sources, resulting in duplicate records and further inconsistencies.

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 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* =. After removing duplicate entries caused by overlapping sources, the final *Δ* contains 2,705 records.

For training, the *CC* collaborates with the *SP*, but since medical data is highly sensitive, the *SP* only receives encrypted data for training. However, directly encrypting numeric values can destroy important mathematical relationships needed by many ML models. For example, encrypted numbers lose their order, which affects models that depend on comparisons or value ranges. Even tree-based models like RF or XGB may struggle, as encryption can create many unique values that reduce split efficiency. To address this, the *CC* categorizes all numeric features into 3–4 groups before encryption. This balances privacy with model compatibility. A performance comparison between encrypted categorized and directly encrypted numeric data is presented in Section [5.4](#_Comparative_Analysis_of) to explore the effectiveness of this choice.

Table 2 shows each attribute, its original data type, and its categorized version.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 2** Heart Disease Prediction Attributes | | | |
| *Attribute* | *Unit* | *Type of Data* | *Categorized Data* |
| Age | Nominal | In Years | Young (≤40), Middle-aged (41–50), Senior (51–60), Elderly (>60) |
| Sex | Nominal | Male, Female | Male, Female |
| Chest Pain Type | Nominal | Typical Angina, Atypical Angina, Non-anginal Pain, Asymptomatic | Typical Angina, Atypical Angina, Non-anginal Pain, Asymptomatic |
| Resting Blood Pressure | Nominal | 94-200 (mm HG) | Low (<110 mm Hg), Normal (110–130 mm Hg), Pre-high (131–150 mm Hg), High (>150 mm Hg) |
| Cholesterol | Nominal | 126-564 (mg/dl) | Desirable (<200 mg/dL), Borderline High (200–239 mg/dL), High (≥240 mg/dL) |
| Fasting Blood Sugar | Binary | Yes / No >120 mg/dl | Yes / No >120 mg/dl |
| Resting ECG | Nominal | Normal, Abnormal ST-T Wave, Left Ventricular Hypertrophy | Normal, Abnormal ST-T Wave, Left Ventricular Hypertrophy |
| Maximum Heart Rate | Nominal | 71-202 | Low (<100 bpm), Moderate (100–140 bpm), High (>140 bpm) |
| Exercise Angina | Binary | Yes / No | Yes / No |
| Oldpeak | Nominal | 0 - 6.2 | None (0), Mild (0.1–1.0), Moderate (1.1–2.0), Severe (>2.0) |
| ST Slope | Nominal | Upsloping, Flat, Downsloping | Upsloping, Flat, 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.

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

**4.3.2 Key Generation**

The *CC* generates a pair of public and private keys (*κ* and *ρ*) using the ElGamal cryptosystem, as described in Section [3.1](#_ElGamal_Cryptosystem). The key generation process is detailed in Algorithm 1. 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.

|  |
| --- |
| 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** |

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

**4.3.3 Data Encryption**

*CC* and the two *CR*s secure their respective parts of the dataset, which together contain *d* feature vectors (*ξ*1*, ξ*2*, ..., 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* ← *str\_to\_int*(*ξij*) 10. *c1* ← *gk mod p* 11. *c2* ← (*ξ'ij* × *hki mod p*) *mod p* 12. *Θi*.*append*((*c*1, *c*2)) 13. **else**: 14. *Θi*.*append*(*∅*) 15. **end if** 16. **end for** 17. **return** *Θi* 18. **end function** |

For the labels *λ*, *CC* applies a simple binary encoding, converting 'Yes' to 1 and 'No' to 0, producing *λ′*. Once encryption is complete, each *CR* sends the encrypted attributes along with the corresponding ephemeral keys to the *CC*. The *CC* then 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.

**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](#_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*s. 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. *c*1*, c*2← *split(Θij)* 9. *temp* ← (*c*1*a mod p*)-1 *mod p* 10. *val* ← *int\_to\_str*((*c*2 *\* 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 removing any *Θij* entries containing them. Once all missing values are eliminated, the remaining data 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:

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

where, Mean,

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

and, Standard Deviation,

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

**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 DT (*η*1), XGB (*η*2), and RF(*η*3), 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), Precision (*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. The rationale for selecting Π7​ is explained in Section [5.2](#_Experimental_Results_and). 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* evaluates *ηα*, the model setup that yields the highest *Π*₇. To identify the optimal configuration, *SP* follows a stage-wise grid search approach—tuning one hyperparameter at a time across several rounds. In this work, SP explores five model combinations—RF, RF with PCA, RF with iForest, RF with XGB, and RF with iForest and XGB—for a deeper analysis of performance.

The following hyperparameters are fine-tuned for the models:

1. RF: Number of trees (n\_estimators) and random seed (random\_state).
2. RF+PCA: n\_estimators, n\_components (number of principal components), random\_state.
3. RF+iForest: n\_estimators, contamination (outlier rate), random\_state.
4. RF+XGB: n\_estimators, use\_label\_encoder, eval\_metric, random\_state.
5. RF+iForest+XGB: n\_estimators, contamination, use\_label\_encoder, eval\_metric, random\_state.

**4.3.8 Final Model Selection**

The *SP* conducts a comprehensive evaluation of five configurations of the best-performing model *ηα*: the base model RF (*ηα*1), RF with PCA (*ηα*2), RF with iForest (*ηα*3), RF with XGB (*ηα*4), and RF with iForest and XGB (*ηα*5). For each configuration, *SP* applies an 8:2 train-test split to obtain *θtrain* and *θtest*. The configuration that yields the best *Π*7 is selected as the optimal model, *ηβ*. The best model *ηβ*, along with its corresponding *θtest*, *Π*7, and the preprocessing transformer *δ*, is 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 rows from *θtest* and retrieves their corresponding serial numbers from *θ*. Using these serial numbers, the *CC* first removes *∅* from its own *Θ*, then extracts the matching rows. This subset of *Θ* is preprocessed using *δ* to produce *θ* and evaluated using *ηβ*.

The *CC* then calculates the performance metric *Π*7 and compares it with the *Π*7 provided by the *SP*. If the values match exactly, the *CC* proceeds to develop the *SYS* using *ηβ*.

**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 his 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 categorized, then re-encrypted using the ephemeral public keys that were already generated by *CC* and *CR*s—one unique key for each feature. These re-encrypted values are then preprocessed using *δ* and passed into the *ηβ*, 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.

# **Experimental Analysis**

## **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. The complete source code is available at [31].

## **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 3 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 3** Key Generation Time and Key Size | | | |
| *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 4 shows that encrypting the full dataset (split across *CR*1, *CR*2, and *CC*) — totaling approximately 1.72 MB — takes between 5.45 seconds (sum of all minimum times) and 5.55 seconds (sum of all maximum times) across the three parts. The encryption process significantly increases the dataset size, with the combined encrypted size ranging from approximately 4.24 MB to 4.25 MB. Decryption times remain consistent, ranging from 0.91 to 0.96 seconds, highlighting the efficiency and reliability of the process over multiple attempts.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Table 4** Dataset Size, and Time Required for Encryption and Decryption | | | | | | | | | | |
| *Attempt* | *Dataset Size (KB)* | | | *Encryption Time (sec)* | | | *Encrypted Dataset Size (KB)* | | | *Decryption Time (sec)* |
| *CR*1 | *CR*2 | *CC* | *CR*1 | *CR*2 | *CC* | *CR*1 | *CR*2 | *CC* |
| 1 | 749.36 | 698.96 | 273.04 | 2.601 | 2.424 | 0.4618 | 1998.41 | 1888.65 | 361.94 | 0.9197 |
| 2 | 749.36 | 698.96 | 273.04 | 2.6109 | 2.4151 | 0.4378 | 2000.16 | 1893.66 | 360.67 | 0.9068 |
| 3 | 749.36 | 698.96 | 273.04 | 2.602 | 2.4822 | 0.4543 | 2001.46 | 1888.79 | 357.77 | 0.9573 |

Table 5 presents the training times for selected ML algorithms. DT trains the fastest, averaging around 0.46 seconds. XGB takes slightly longer, close to 1.90 seconds, while RF requires about 2.35 seconds on average.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 5** Time Required for Model Training | | | |
| *Attempt* | *DT (in sec)* | *XGB (in sec)* | *RF (in sec)* |
| 1 | 0.4419 | 1.9025 | 2.418 |
| 2 | 0.4683 | 1.9238 | 2.3508 |
| 3 | 0.4701 | 1.8655 | 2.2898 |

Table 6 shows the training times for RF with various combinations. RF alone takes around 0.27 seconds. When combined with PCA, training time is about 0.60 seconds, and with iForest, it is near 0.44 seconds. RF with XGB requires approximately 0.35 seconds, while the combination of RF, iForest, and XGB takes close to 0.54 seconds.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 6** Time for Training RF with Different Methods | | | | | |
| *Attempt* | *RF (in sec)* | *RF+PCA (in sec)* | *RF+iForest (in sec)* | *RF+XGB (in sec)* | *RF+iForest+XGB (in sec)* |
| 1 | 0.2704 | 0.6072 | 0.4361 | 0.3512 | 0.5583 |
| 2 | 0.2781 | 0.5676 | 0.4437 | 0.3589 | 0.5197 |
| 3 | 0.2738 | 0.6215 | 0.4273 | 0.352 | 0.5395 |

The performance of ML models was evaluated using several key metrics. As discussed in Section [3.11](#_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.

For training, tree-based models such as DT, XGB, and RF were chosen due to their ability to effectively capture complex, non-linear patterns within the data. These models are well-suited to handle features that may be encrypted or transformed, as they do not rely heavily on feature scaling or strict assumptions about data distribution.

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, specificity, precision, recall, 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.

The models evaluated include DT, XGB, and RF. Among these, RF emerges as the top-performing algorithm, achieving the highest accuracy (86.78%), recall (93.07%), F1-score (90.66%), and AUC (91.33%). These results demonstrate RF's strong capability to generalize and deliver consistent classification performance across key metrics.

XGB also performs well, offering a strong balance between precision and recall. It achieves an accuracy of 85.17%, F1-score of 89.41%, and a notably high AUC of 90.26%, confirming its reliability in handling the classification task effectively.

DT shows comparatively modest performance, with an accuracy of 80.98% and AUC of 77.98%. While DT maintains a decent balance across metrics like recall (87.15%) and F1-score (86.07%), its lower specificity (68.38%) and NPV (72.02%) highlight some limitations in distinguishing negative cases.

|  |
| --- |
|  |
| **Fig. 5** Performance Metrics of Classification Algorithms with Varying Train-Test Split |
| **Fig. 6** Average Performance Metrics of Classification Algorithms with Varying Train-Test Split |

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 selected as the best model due to its consistent superiority across all evaluation metrics. XGB also demonstrates strong and stable performance, making it a close contender. DT, while dependable in some aspects, falls short in key metrics and is therefore considered less robust 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. precision 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 7 presents a comparative analysis of RF-based approaches, including standalone RF, RF with dimensionality reduction (PCA), RF with anomaly detection (iForest), RF combined with XGB, and a hybrid integrating RF, iForest, and XGB as an ensemble.

Among all configurations, RF combined with XGB achieved the highest F1-score of 94.78%, showing a slight edge in performance. The standalone RF model also performed strongly, with a very close F1-score of 94.56%, confirming its robustness and reliability. The RF with iForest and XGB hybrid followed with 94.22%, indicating that combining anomaly detection and boosting may enhance performance, but only marginally. Interestingly, RF with iForest alone achieved a slightly lower F1-score of 94.22%, still competitive, but not as effective as when paired with XGB. The RF with PCA model had the lowest F1-score of 93.85%, suggesting that dimensionality reduction may have led to minor information loss, slightly reducing model effectiveness. Overall, the RF with XGB ensemble stands out as the most effective configuration, achieving the best balance of accuracy and generalization.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 7** Performance Comparison of Classification Models Using 8:2 Train-Test Split with Tuned Hyperparameters | | | | | |
| *Classifiers* | *Hyper-parameters* | | | | *F1-score (in %)* |
| RF | train\_test\_split   * random\_state: 1786 | RandomForestClassifier   * random\_state: 1397 * n\_estimators: 100 |  |  | 94.56% |
| RF+PCA | train\_test\_split   * random\_state: 159 | RandomForestClassifier   * random\_state: 1745 * n\_estimators: 100 | PCA   * n\_components: 0.95 * random\_state: 1745 |  | 93.85% |
| RF+iForest | train\_test\_split   * random\_state: 159 | RandomForestClassifier   * random\_state: 227 * n\_estimators: 94 | IsolationForest:   * contamination: 0.05 * random\_state: 227 * n\_estimators: 100 |  | 94.22% |
| RF+XGB | train\_test\_split   * random\_state: 1786 | RandomForestClassifier   * random\_state: 1397 * n\_estimators: 99 | XGBClassifier   * random\_state: 1397 * use\_label\_encoder: False * eval\_metric: 'logloss' |  | 94.78% |
| RF+iForest+XGB | train\_test\_split   * random\_state: 227 | RandomForestClassifier   * random\_state: 227 * n\_estimators: 94 | IsolationForest:   * contamination: 0.05 * random\_state: 227 * n\_estimators: 100 | XGBClassifier   * random\_state: 227 * use\_label\_encoder: False * eval\_metric: 'logloss' | 94.22% |

Table 8 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 (21 bytes), and a digital signature (72 bytes). Similarly, the *SYS* responds with its own encrypted AES key (8 bytes), AES-encrypted output (67 bytes), and a digital signature (144 bytes).

|  |  |  |
| --- | --- | --- |
| **Table 8** 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 | 21 × 64 | 168 |
| *SU*'s AES-encrypted data | 21 characters | 21 |
| *SYS*'s RSA-encrypted data | 67 × 64 | 536 |
| *SYS*'s AES-encrypted data | 67 characters | 67 |
| RSA-encrypted AES key | 64 | 8 |
| *SU*'s RSA digital signature | 9 × 64 | 72 |
| *SYS*'s RSA digital signature | 18 × 64 | 144 |

Both the RSA and AES encryption methods were implemented and compared. *SU*’s RSA-encrypted input requires 168 bytes, while AES-encrypted input with an RSA-encrypted AES key needs only 29 bytes (21 + 8). Similarly, *SYS*’s RSA-encrypted output occupies 536 bytes, compared to just 75 bytes (67 + 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.

## **Consistency Analysis**

Table 9 presents a comparative analysis of classification performance across three different data handling approaches: the Combined Dataset, the Structured Dataset, and the Encrypted Dataset. All models were trained using the RF with XGB configuration with consistent hyperparameter optimization via stage-wise grid search and an 8:2 train-test split.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 9** Classification Performance Comparison Across Different Data Handling Approaches | | | | |
| *Dataset Type* | *Accuracy (%)* | *Precision (%)* | *Recall (%)* | *F1-Score (%)* |
| Combined Dataset | 91.64 | 96.49 | 96.88 | 93.36 |
| Structured Dataset | 91.49 | 92.89 | 98.14 | 94.09 |
| Encrypted Dataset | 92.40 | 94.32 | 98.20 | 94.78 |

The Combined Dataset refers to the raw Kaggle datasets that were first merged and then preprocessed by removing duplicate entries. For model training, additional steps included missing value removal, unification of numerical value ranges, categorical label encoding, standardization, and hyperparameter tuning. This configuration resulted in an accuracy of 91.64%, precision of 96.49%, recall of 96.88%, and F1-score of 93.36%.

The Structured Dataset was prepared by first combining the Kaggle datasets and removing duplicate records. Numerical data were categorized based on defined value ranges to better reflect feature semantics. During training, missing values were removed, all categorical features were label-encoded, and standardization was applied. Hyperparameter tuning was performed to optimize model performance. This setup achieved an accuracy of 91.49%, precision of 92.89%, recall of 98.14%, and F1-score of 94.09%.

The Encrypted Dataset reflects the proposed method, where all input features were categorized and encrypted before training. Despite the added cryptographic complexity, the model maintained competitive performance—achieving an accuracy of 92.4%, precision of 94.32%, recall of 98.2%, and F1-score of 94.78%. These results indicate that the encryption process does not significantly hinder the effectiveness of model training, validating the practicality of the proposed secure approach.

Although the Encrypted Dataset shows slightly higher values in several metrics, it is worth mentioning that the Combined and Structured Datasets might achieve comparable or even better results if more diverse models were tested and the best chosen. However, the aim here is not to identify the absolute best model, but to demonstrate that the proposed encryption method allows secure training without significantly affecting model performance. As shown in Table 9, the encrypted approach performs competitively with traditional methods, supporting its practicality for privacy-conscious machine learning applications.

## **Comparative Analysis of Encrypted Data Types**

To evaluate the impact of encryption strategies on model performance, two types of encrypted data were compared: Encrypted Numerical Data and Encrypted Categorized Data. Both data types were processed using the same modeling framework to determine which representation better supports accurate classification while maintaining data confidentiality.

For encrypted numerical data, initially, DT, XGB, and RF models were trained on encrypted numerical data. Among these, RF achieved the highest F1-score of 86.55%, slightly outperforming XGB (85.20%) and DT (80.27%). Based on this result, RF was selected as the baseline for further optimization.

Several RF-based configurations were then examined, including RF with PCA, RF with iForest, RF with XGB, and a hybrid RF with iForest and XGB model. Both RF and RF with XGB achieved an F1-score of 91.22%, while RF with iForest reached 90.31%, and RF with PCA scored 88.84%. The hybrid RF with iForest and XGB model produced the best performance with an F1-score of 92.84%, and was therefore selected as the optimal model for encrypted numerical data. In all cases, hyperparameter optimization was conducted using stage-wise grid search to ensure each configuration was fairly tuned for maximum performance.

Table 10 presents a summary of the best performing models on both encrypted numerical and categorized datasets. The results show that the model trained on encrypted categorized data outperforms the numerical counterpart in all key metrics. The RF with XGB model, applied to categorized data, achieved higher accuracy, precision, recall, and F1-score, demonstrating that securely encoding categorical features preserves essential information.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 10** Performance of the Best Models on Encrypted Numerical and Categorized Data | | | | | |
| *Data Type* | *Best Performing Model* | *F1-score (%)* | *Accuracy (%)* | *Precision (%)* | *Recall (%)* |
| Encrypted Numerical Data | RF+iForest+XGB | 92.84 | 89.56 | 92.52 | 94.93 |
| Encrypted Categorized Data | RF+XGB | 94.78 | 92.4 | 94.32 | 98.2 |

## **Comparative Analysis with Existing Methods**

The comparative analysis between the proposed system and the existing methodologies is described in Table 11. It provides a concise evaluation of core aspects such as data handling, model development, security mechanisms, validation strategies, and user privacy.

The proposed system utilizes a combination of seven Kaggle datasets, offering greater diversity and robustness compared to prior studies that relied on smaller, more constrained datasets [[7, 8]](#_References). Unlike [[9]](#_References), which primarily uses the Kaggle dataset and applies outlier removal and categorical conversions, the proposed method adopts a more extensive preprocessing pipeline. This includes handling duplicates and missing values, standardization, PCA, and encryption, thereby ensuring both data quality and privacy.

While [[9]](#_References) and [[12]](#_References) apply common methods or minimal preprocessing, the proposed system stands out by encrypting features using ElGamal and applying PCA to reduce dimensionality securely. Moreover, the use of stage-wise grid search for hyperparameter tuning adds another layer of performance optimization, in contrast to the fixed strategies like GridSearchCV or RandomizedSearchCV used in [[9]](#_References), [[12]](#_References), and [[15]](#_References).

The accuracy achieved by the proposed system (92.4%) remains competitive, even though it may fall slightly behind the results reported in [[12]](#_References) (93.44%, 95%) and [[15]](#_References) (98.15%). However, it’s important to note that the proposed system delivers this performance while incorporating strong security guarantees. It implements encrypted model training, end-to-end encryption, and rigorous validation using digital signatures—security features not found in the other approaches.

Entities within the system, including *CC*, *SP*, *CR*, *SU*, and *SYS*, handle distinct responsibilities to preserve confidentiality and ensure secure model development. With verified input and output using *SU*'s signature and anonymized interactions throughout, the system delivers a trustworthy and secure experience, which sets it apart from existing models focused solely on accuracy.

Ultimately, the proposed method strikes a balance between performance and privacy, making it a notable advancement in secure machine learning over encrypted data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table 11** Comparative Analysis of Proposed System with Existing Methods | | | | | | |
| *Attributes* | [*[7]*](#_References) | [*[8]*](#_References) | [*[12]*](#_References) | [*[13]*](#_References) | [*[14]*](#_References) | *Proposed System* |
| *Novelty of Approach* | FCMIM for faster, accurate classification | HRFLM hybrid model | K-modes clustering with Huang start | - | - | Data privacy, Model training on encrypted data |
| *Dataset source and size* | Cleveland (303 rows, 13 features) | Cleveland (303 rows, 13 features) | Kaggle dataset (70,000 rows, 11 features) | Cleveland (303 rows, 13 features), IEEE Dataport (1,190 rows, 11 features) | Cleveland, Hungary, Switzerland, Long Beach V (1625 rows, 13 features) | Cleveland, Hungary, Switzerland, Long Beach V, UCI, Stalog (Heart), a hospital in India (5,461 rows, 11 features) |
| *Dataset Security* | - | - | - | - | - | Dataset encrypted by *CR*s |
| *Integrity of Dataset* | - | - | - | - | - | Encrypted dataset verified by *CC* |
| *Model trained-on* | Raw data | Raw data | Categorized data | Raw data | Raw data | Encrypted categorized data |
| *Data Quality and Preprocessing Techniques* | Missing values removal, SS, Min-Max | Missing values removal, Binary conversion | Outlier removal | - | Missing values handling, Duplicates, outliers removal, Normaization | Duplicate, missing values removal, Standardization, PCA, iForest, ElGamal Encryption |
| *Feature Engineering* | Relief, MRMR, LASSO, LLBFS | HRFLM method | - | - | - | - |
| *ML Algorithms* | ANN, LR, KNN, SVM, DT, NB | DT, LM, SVM, RF, NB, NN, KNN | DT, RF, MLP, XGB | SVE, RF, KNN, LR, NB, GB, AB, | Extra Trees, RF, XGB, CatBoos | DT, XGB, RF |
| *Hyperparameter Tuning* | Leave-one-subject-out | - | GridSearchCV | GridSearchCV | GridSearchCV, RandomizedSearchCV | Stage-wise grid search |
| *Evaluation Matrices* | Accuracy, Specificity, Recall, MCC | Accuracy, Recall, Specificity, Precision, F-Measure | Accuracy, Precision, Recall, F-measure | Accuracy, Precision, Recall, F1-score | Accuracy, Recall, Precision, F1-score, Cohen’s kappa, AUC-ROC | Accuracy, Recall, Specificity, Precision, NPV, F1-score, AUC |
| *Best Performing Model, Result* | FCMIM-SVM,  Accuracy: 92.37%,  Recall: 89% | HRFLM,  Accuracy: 88.4%,  Precision: 90.1%,  Recall: 92.8%,  F-measure: 90% | MLP,  Accuracy: 87.28%,  Precision: 88.70%,  Recall: 84.85%,  F1-score: 86.71% | SVE,  Accuracy (Cleveland): 93.44%, (IEEE): 95% | Extra Trees,  Accuracy: 98.15%,  Precision: 97.48%,  Recall: 98.72%,  F1-score: 98.10% | RF+XGB,  Accuracy: 92.4%,  Precision: 92.32%,  Recall: 98.2%,  F1-score: 94.78% |
| *Model Validation* | - | - | - | - | - | Validated by *CC* after trained by *SP* |
| *Entities and Responsibilities of Data* | Single authority | Single authority | Single authority | 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 |

# **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.

## **Incomplete Protection through Categorization**

*Threat: While categorizing numeric data before training helps to reduce granularity and obscure exact values, it does not provide complete protection against data inference. An attacker with domain knowledge or access to external datasets might analyze the distribution of categories and correlate them with real-world patterns to infer sensitive information.*

*Solution:* As described in Section [4.3.3](#_Individual_stages), encryption is applied after categorization to prevent any unauthorized interpretation of these categorized values. By encrypting each category using the ElGamal scheme, the final data becomes indistinguishable and unlinkable, even if the original value ranges are known.

## **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](#_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 significantly reduces 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.

## **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](#_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](#_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.

## **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](#_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.

## **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 metric calculated by *CC* with that provided by the *SP* to ensure accuracy and integrity.

## **Corruption Risk**

*Threat: Corruption of CC, CRs, or SP may result in 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](#_Individual_stages) and verifies the models and data in Section [4.3.8](#_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.

## **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. Additionally, if the system allowed the SU to directly encrypt input using ElGamal, the SYS would have to share both its public key and the ephemeral keys used for encryption. If the SU were malicious or shared the ephemeral key with an attacker, and the attacker obtained encrypted data—either from other SUs or from the medical datasets used for training—the original values could be recovered using the formula m*=*c*2/*hk mod p, as described in Section* [*3.1*](#_ElGamal_Cryptosystem)*, where h is ga (the public key component). This is possible using known values such as p, g, h, c*₁*, c*₂*, and k, which expose sensitive health data and undermine the entire privacy-preserving mechanism.*

*Solution:* As outlined in Section [4.3.10](#_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.

## **Data Tampering and Forgery**

*Threat: There’s 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](#_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.

# **Limitations**

While the proposed system successfully protects sensitive data and supports a range of ML models, a few limitations were observed.

The use of a static ephemeral key for each feature means that the same encryption key is applied across all data points within that feature. Although this simplifies encryption and maintains consistency, it could potentially allow attackers to detect patterns from repeated use of the same key.

Additionally, the encryption and decryption operations—especially using ElGamal—introduce computational overhead. This may limit scalability or pose challenges in real-time or resource-constrained environments.

Despite these limitations, the overall model performance, particularly the F1-score, remains very close to that achieved on unencrypted data, demonstrating a good balance between data security and model effectiveness.

# **C****onclusion**

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, categorization, encryption, preprocessing, and model training with the use of DT, XGB, and RF 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. The distribution transformed by the DT gave it a fair average mean F1-score of 86.07%, while the XGB showed a mean F1-score of 89.41%. Even though both models performed well, their results were generally lower compared to those given by the models based on RF, which achieved an F1-score of 90.66%. Among all the RF-based models, the RF with XGB has shown the best performance with an average F1-score value of 94.78%, which exceeds the performances of all other configurations. For RF only, the mean F1-score is 94.56%, while that for both RF with iForest and RF with iForest and XGB is 94.22%. RF together with PCA had the lowest mean F1-score of 93.85%. Such results provide proof that RF with XGB is the most powerful, taking the benefits of both deep tree-based modeling and gradient boosting.

By addressing the challenges of data privacy, model accuracy, 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.

# **Statements and Declarations**

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

# **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. doi: <https://doi.org/10.1002/ehf2.12005>
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. doi: <https://doi.org/10.1007/s12652-018-0724-4>
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. doi: <https://doi.org/10.1161/CIR.0b013e31820a55f5>
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. doi: <https://doi.org/10.1016/j.media.2012.02.005>
6. Ghwanmeh S, Mohammad A, Al-Ibrahim A. Innovative artificial neural networks-based decision support system for heart diseases diagnosis. doi: <https://doi.org/10.4236/jilsa.2013.53019>
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. doi: <https://doi.org/10.1109/ACCESS.2020.3001149>
8. Mohan S, Thirumalai C, Srivastava G. Effective heart disease prediction using hybrid machine learning techniques. IEEE access. 2019 Jun 19;7:81542-54. doi: <https://doi.org/10.1109/ACCESS.2019.2923707>
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. doi: [https://doi.org/10.1109/ACCESS.2022.3191669](https://doi.org/10.1109/ACCESS.2022.3191669%20)
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. doi: <https://doi.org/10.1109/CCWC57344.2023.10099294>
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. doi: <https://doi.org/10.1109/I2CT45611.2019.9033683>
12. Bhatt CM, Patel P, Ghetia T, Mazzeo PL. Effective heart disease prediction using machine learning techniques. Algorithms. 2023 Feb 6;16(2):88. doi: <https://doi.org/10.3390/a16020088>
13. Chandrasekhar N, Peddakrishna S. Enhancing heart disease prediction accuracy through machine learning techniques and optimization. Processes. 2023 Apr 14;11(4):1210. doi: <https://doi.org/10.3390/pr11041210>
14. Asif D, Bibi M, Arif MS, Mukheimer A. Enhancing heart disease prediction through ensemble learning techniques with hyperparameter optimization. Algorithms. 2023 Jun 20;16(6):308. doi: <https://doi.org/10.3390/a16060308>
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. doi: <https://doi.org/10.1109/ACCESS.2023.3289584>
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. doi: 10.5220/0002593600790084
18. 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. doi: <https://doi.org/10.1109/ICSMC.1998.725047>
19. Chen T, Guestrin C. Xgboost: A scalable tree boosting system. InProceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining 2016 Aug 13 (pp. 785-794). doi: <http://dx.doi.org/10.1145/2939672.2939785>
20. Pal M. Random forest classifier for remote sensing classification. International journal of remote sensing. 2005 Jan 1;26(1):217-22. doi: <https://doi.org/10.1080/01431160412331269698>
21. Wold S, Esbensen K, Geladi P. Principal component analysis. Chemometrics and intelligent laboratory systems. 1987 Aug 1;2(1-3):37-52. doi: <https://doi.org/10.1016/0169-7439(87)80084-9>
22. Liu FT, Ting KM, Zhou ZH. Isolation forest. In2008 eighth ieee international conference on data mining 2008 Dec 15 (pp. 413-422). IEEE. doi: <https://doi.org/10.1109/ICDM.2008.17>
23. 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). doi: 10.1007/978-3-319-23528-8 1
24. [dataset] Abhishek AVS. *Heart Disease Classification.* Kaggle; 2023. [online] Available at: <https://www.kaggle.com/datasets/abhishek14398/heart-disease-classification> [Accessed: August 2024].
25. [dataset] Fedesoriano. *Heart Failure Prediction Dataset.* Kaggle; 2021. [online] Available at: <https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction> [Accessed: August 2024].
26. [dataset] Dumlao J. *Cardiovascular\_Disease\_Dataset.* Kaggle; 2024. [online] Available at: <https://www.kaggle.com/datasets/jocelyndumlao/cardiovascular-disease-dataset> [Accessed: August 2024].
27. [dataset] Damarla R. *Heart Disease Prediction.* Kaggle; 2020. [online] Available at: <https://www.kaggle.com/datasets/rishidamarla/heart-disease-prediction> [Accessed: August 2024].
28. [dataset] Sony R. *UCI Heart Disease Data.* Kaggle; 2020. [online] Available at: <https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data> [Accessed: August 2024].
29. [dataset] Ineubytes. *Heart-Disease-Dataset.* Kaggle; 2023. [online] Available at: <https://www.kaggle.com/datasets/abhishek14398/heart-disease-classification> [Accessed: August 2024].
30. [dataset] Lapp D. *Heart Disease Dataset.* Kaggle; 2019. [online] Available at: <https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset> [Accessed: August 2024].
31. Esha IT. Heart-Disease-Prediction. GitHub; 2024. [online] Available at: <https://github.com/IsratTasnimEsha/Heart-Disease-Prediction> [Accessed: May 2025].