Jahangir et al. BMC Cardiovascular Disorders (2025) 25:260
https://doi.org/10.1186/s12872-025-04678-9

BMC Cardiovascular Disorders

RESEARCH

Open Access

ECG-based heart arrhythmia classification using feature engineering and a hybrid stacked machine learning

Raiyan Jahangir[1], Muhammad Nazrul Islam[2*], Md. Shofiqul Islam[3] and Md. Motaharul Islam[4]

Abstract

A heart arrhythmia refers to a set of conditions characterized by irregular heart- beats, with an increasing mortality rate in recent years. Regular monitoring is essential for effective management, as early detection and timely treatment greatly improve survival outcomes. The electrocardiogram (ECG) remains the standard method for detecting arrhythmias, traditionally analyzed by cardiolo- gists and clinical experts. However, the incorporation of automated technology and computer-assisted systems offers substantial support in the accurate diagno- sis of heart arrhythmias. This research focused on developing a hybrid model with stack classifiers, which are state-of-the-art ensemble machine-learning techniques to accurately classify heart arrhythmias from ECG signals, eliminating the need for extensive human intervention. Other conventional machine-learning, bagging, and boosting ensemble algorithms were also explored along with the proposed stack classifiers. The classifiers were trained with a different number of features (50, 65, 80, 95) selected by feature engineering techniques (PCA, Chi-Square, RFE) from a dataset as the most important ones. As an outcome, the stack clas- sifier with XGBoost as the meta-classifier, trained with 65 important features determined by the Principal Component Analysis (PCA) technique, achieved the best performance among all the models. The proposed classifier achieved a perfor- mance of 99.58% accuracy, 99.57% precision, 99.58% recall, and 99.57% f1-score and can be promising for arrhythmia diagnosis.

Keywords ECG, Heart arrhythmia, Machine learning, Stack classifier

Background

A heartbeat is a periodic relaxation and contraction of the heart muscle that drives blood through the circulatory system [1, 2]. In a healthy heart, impulses follow a regu- lar and coordinated pattern, often referred to as a sinus rhythm [3]. Heart arrhythmia is a common cardiac condition that describes any abnormal heart rhythm. It occurs when the electrical impulses that regulate the heartbeat go awry, causing the heart to beat quickly, slowly, or irregularly. Arrhythmia can happen independently or with other cardiovascular conditions [4]. Although some arrhythmias are not dangerous, some have the potential to cause abrupt cardiac arrest, heart failure, stroke, and other cardiovascular diseases (CVDs). An estimated 17.7 million people died because of CVDs in 2017, which accounts for 31% of all deaths [5]. An essential tool in diagnos- ing arrhythmia is the electrocardiogram (ECG) besides other biointegrated wearable and implantable optoelectronic devices [6–9]. ECG is a crucial medical equipment that captures the heart's

*Correspondence:
Muhammad Nazrul Islam
nazrul@cse.mist.ac.bd
[1] Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Tejgaon, Dhaka 1208, Bangladesh
[2] Department of Computer Science and Engineering, Military Institute of Science and Technology, Mirpur Cantonment, Dhaka 1216, Bangladesh
[3] Institute for Intelligent Systems Research and Innovation (ISSRI), Deakin University, 75 Pigdons Rd, Warun Ponds, Victoria 3216, Australia
[4] Department of Computer Science and Engineering, United International University (UIU), Madani Avenue, Badda, Dhaka 1212, Bangladesh

excitability, transmission, and recovery [10]. The result of an ECG is a signal representation corresponding to the heart's electrical activity. Physi- cians inspect the pattern of the signals to identify any arrhythmias. With the advent of artificial intelligence and machine learning [11], researchers have been trying their best to incorporate machine learning in classifying arrhythmia in ECG signals.

ML techniques have been used in multidisciplinary fields for prediction purposes that include health informatics [12, 13], disaster forecasting [14], agriculture [15], monitoring systems [16], and so on. Similarly, several ML [17] and Deep learning (DL) [18, 19] techniques have been applied to classify heart arrhythmia. However, there is always room for improvement. Initially, ML algorithms were used to carry out such classification tasks. Melgani et al. [20] demonstrated the SVM algorithm's capacity to generalize the classification of ECG beats. They used Particle Swarm Opti- mization (PSO) to boost the SVM classifier's performance in terms of generalization (accuracy =89.72%). Kumar et al. [21] described a beat-to-beat interval-based ECG classification approach for arrhythmic beats. The beat-to-beat intervals were extracted from the ECG signals and converted into Discrete Cosine Transform (DCT) as part of the methodology. Then, the transformed beats were classified using the Random For- est algorithm (accuracy =92.16%). Park et al. [22] created a system that uses features like P wave and QRS complex for detecting heartbeats and the k-nearest neighbor (KNN) algorithm for classifying them (97.22% sensitivity and 97.4% specificity for heartbeat detection, 97.1% sensitivity and 96.9% specificity for classification). Ardeti et al. [23] utilized an improved filtering method to identify the extreme outliers of the signal for ranking features. A heterogeneous classification model based on an Opti- mized Random Forest (ORF) was also presented to increase the true positive of the ECG data. The majority voting technique was used to classify each type of heartbeat (accuracy =96.17%).

Eventually, deep learning techniques have evolved, and studies now focus more on these newer techniques. Ubeyli [24] integrated recurrent neural networks (RNN) and eigenvector techniques to extract features and classify ECG beats based on the extracted features. Guler and Ubeyli utilized feedforward neural Networks (FFNN) [25] to classify ECG beats (accuracy =96.94%). Li et al. [26]. suggested a general model based on ResNet to achieve the automated classification of regular rhythm. The 12- lead ECG signal was cut into a two-dimensional plane and rendered like a grayscale image. The intrinsic features of the two-dimensional ECG were extracted using DSE- ResNet. Furthermore, the DSE-ResNet's hyper-parameters were optimized using an orthogonal experiment approach, and classification performance was increased using a multi-model voting strategy (test f1-score =81.7%). For automatic arrhythmia clas- sification, Ramkumar et al. [27] proposed a combination of autoencoder (AE) and Bi-LSTM. An encoder in the AE-biLSTM approach extracts higher-level features. The decoder output reconstructs ECG signals using bi-LSTM, and heartbeats are finally categorized (accuracy =97.15%). Madan et al. [28] suggested a deep learning technique that combined 2D Convolutional Neural Network (2D-CNN) and Long Short Term Memory (LSTM) to automate the detection and classification process. 2D Scalogram pictures were created from 1D ECG data for noise reduction and feature extraction. After obtaining experimental data, the proposed model was designed, which got 98.7% accuracy.

The stacking ensemble method, or the stack classifier, is a noteworthy state-of-the- art process that integrates the predictions of more than one base model to arrive at the final prediction. It is an ensemble technique that intends to acquire the capabilities of different models and improve the final performance [29]. In a stack classifier, the meta- learner model is used to aggregate the output of various base models that have been trained on the same dataset. The meta-learner learns to provide the final predictions using the underlying models' predictions as input [30]. This architecture sets stack classifiers apart from single models or traditional ensemble methods like bagging and boosting. Though the stack classifier is found to perform better than other individual techniques in predictive tasks, this state-of-the-art approach is still uncommon in classifying heart arrhythmia. Again, only a few studies explored the optimal number of features required to properly classify heart arrhythmia applying numerous feature engineering techniques.

Therefore, the objectives of this research include exploring the performances of conventional ML and ensemble techniques for classifying heart arrhythmia from ECG signals and proposing a stacking classifier that employs an optimal number of features for classifying heart arrhythmia more effectively. Therefore, the key contributions of this research are as follows:

- A stack classifier that is trained and tested with five conventional ML models (Sup- port Vector Machine (SVM), K-Nearest Neighbours (KNN), Logistic Regression (LR), Decision Tree (DT), Multi-Layer Perceptron (MLP)) as weak learners and one model as meta learner to classify the dataset. The meta learner has been selected from the five different kinds of bagging or boosting classifiers, namely Random Forest (RF), Adaboost (AB), Gradient Boosting (GB), eXtreme Gradient Boost- ing (XGB), and Categorical

Boosting (CB). Each developed stack classifier is then evaluated to determine the best-performing classifier.
- For determining the number of optimal features, three feature engineering tech- niques, namely the Chi-Square Test, Principal Component Analysis, and Recursive Feature Elimination, have been applied. Each of the techniques selects different sets of features from the dataset by their method of feature prioritization. These features were then used by the ML methods to classify heart arrhythmia.
- To validate the performance of the proposed stack classifiers, each of the conven-tional, bagging, and boosting ML models were trained separately using the same dataset. A comparison was carried out using different performance parameters.

The remainder of this paper is divided as follows: Sect. 2 describes the methodology in detail. Section 3 presents the research results. Section 4 presents the discussions and concludes the paper.

Methodology

We divide our whole methodology into 5 phases, namely data collection, data pre- processing, feature engineering, model development, and performance analysis. The methodological overview of this systematic approach is depicted in Fig. 1. The whole workflow is described in the following subsections:

Data collection

At first, necessary data was collected. The dataset used was the MIT-BIH arrhythmia database [31] taken from PhysioNet [32]. The MIT-BIH dataset contained 2-channel ambulatory ECG recordings of 48 half-hour snippets utilized from 47 patients in Beth Israel Hospital. The participants included 25 men ranging from 32 to 89 years and 2 women ranging from 23 to 89 years. The participants had a mixed population of 60% inpatients and 40% outpatients. The recordings were digitized over a ten mV range at 360 samples per second per channel. More than two cardiologists independently annotated the records. Finally, around 110,000 annotations were obtained, each having a heartbeat.

The dataset was processed by Kachuee et al. [33]. They converted each of the annotations into a matrix form. Each row of the matrix represents one heartbeat and has 188 columns. The first 187 columns indicate the amplitude of the heartbeat at different time instances. The final column represents the class of the heartbeat. This dataset was used in this study to train the models. There were a total of 5 classes in the dataset.

The train and test data were already split in the dataset. There were 87,554 heart- beats in the train data, 72,471 of which were classified as “Normal” heartbeats (Fig. 2a). The remaining heartbeats belonged to one of the four classes of arrhythmia. 2,223 heartbeats were “Supraventricular heartbeats” [34] (Fig. 2b), 5,788 heart- beats were “Ventricular heartbeats” [35] (Fig. 2 (c)), 641 were “Fusion heartbeats” (Fig. 2d) and the rest, 6,431 heartbeats, did not fall into any of the other four classes, and so were considered as “Mixed heartbeats” (Fig. 2e). The test dataset had 21,892 heartbeats, with 18,118 normal, 556 supraventricular, 1,448 ventricular, 162 fusion, and 1,608 mixed heartbeats. Sample images of the five classes of heartbeats available in the dataset are shown in Fig. 2.

Data synthesis

This phase addressed several data-related challenges to enhance the training process. To begin with, the issue of class imbalance within the dataset was tackled. After this, gaussian

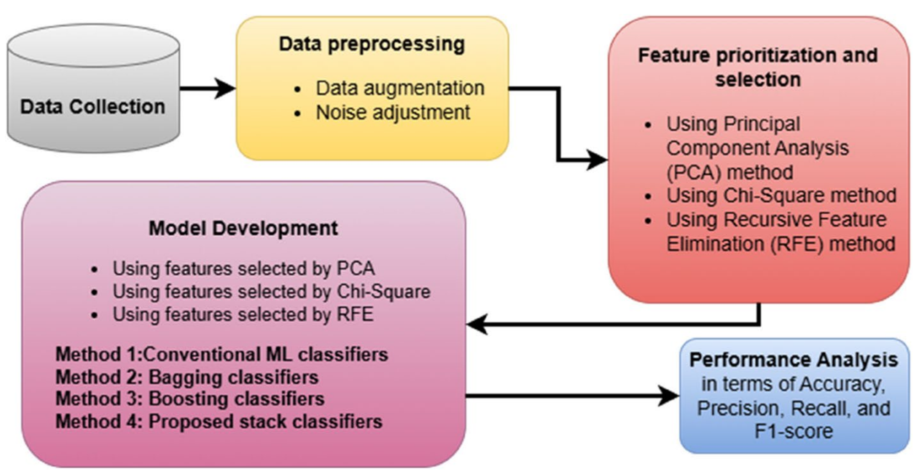

Fig. 1 The methodological overview

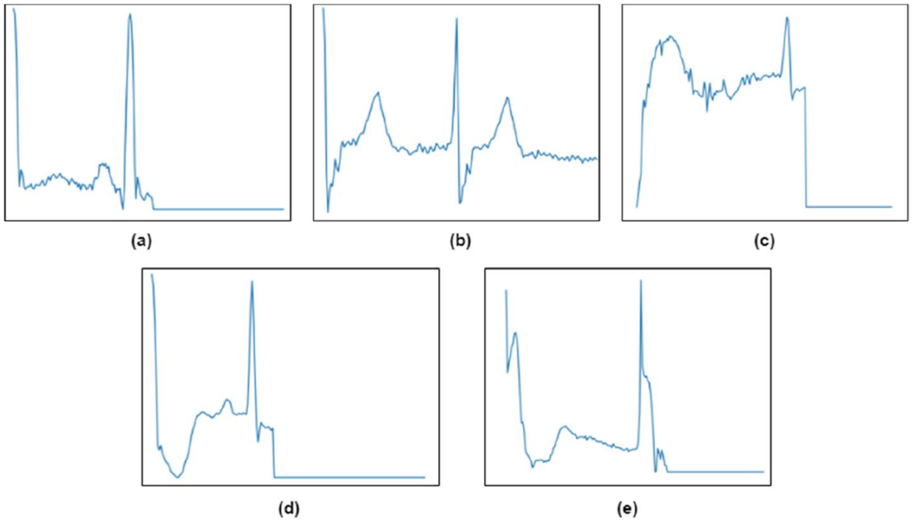

Fig. 2 Classes of Heartbeat: (**a**) Normal (**b**) Supraventricular (**c**) Ventricular (**d**) Fusion (**e**) Mixed

noise was added to the dataset to make the instances more robust to noises. Then, the number of important features was determined followed by feature engineering applying 3 different methods. An algorithm comprising the whole data preprocessing phase is given in Algorithm 1.

Class balancing

Considering that the "normal" class boasted the highest number of instances, totaling 72,471, the Synthetic Minority Oversampling Technique (SMOTE) method [36] was implemented. The SMOTE method generates

Algorithm 1 Algorithm for data preprocessing

Input :
- Raw dataset D
- Set of classes C
- Set of feature engineering methods FE

Output: A set of preprocessed datasets d

max $\leftarrow -\infty$;
foreach $i \in C$ **do**
  if $i_{number} > max$ **then**
    max $\leftarrow i$;
foreach $i \in C$ **do**
  if $i_{number} < max$ **then**
    Apply SMOTE on i;
  Processed Data $D' \leftarrow D' + i$;
foreach $x \in D'$ **do**
  $x_{noisy} \leftarrow x_{original} + N(0, 0.5)$;
  Processed Data $D'' \leftarrow D'' + x_{noisy}$;
Apply OLS to find the number of important features;
Based on the result, No of features $K = [50, 65, 80, 95]$;
Create a set of preprocessed dataset $d \leftarrow []$;
foreach $i \in FE$ **do**
  foreach $j \in K$ **do**
    $d_{ij} \leftarrow$ Apply technique i on data D'' for extracting j features;
    d.append(d_{ij});

some synthetic instances of the classes with a lower number of samples to reduce the class imbalance. The formula for the SMOTE method is shown in Eq. 1.

$$\times_{sample} = \times + \eta(\times_{random} - \times) \tag{1}$$

Here, x_{sample} refers to the generated samples of minority classes x. Whereas, x_{random} refers to a value chosen randomly from the nearest neighbors of x with $0 \leq x \leq 1$. This technique augments the instances in all classes to a consistent count of 72,500, resulting in a massive training dataset of 290,000 instances. The algorithm followed for oversampling with SMOTE is given in Algorithm 2.

$$\times_{noisy} = \times_{original} + N(0,\ 0.5) \tag{2}$$

x_{noisy} refers to the generated noisy samples from the original samples $x_{original}$ with the addition of the random variable N(0, 0.5). The random variable N was sampled using the Gaussian distribution of mean 0 and standard deviation 0.5. The impact of this noise addition on signal characteristics is visualized in Fig. 3.

Likewise, for the test data, SMOTE harmonized the instance counts across all classes to achieve a uniform count of 20,000 instances, totaling 100,000 instances. The same Gaussian noise was added to this dataset with a mean distribution of 0 and a standard deviation of 0.5,

Algorithm 2 Algorithm for oversampling

Input :
- Minority class dataset $x_{minority}$
- N = Number of synthetic samples needed
- k = Number of k nearest neighbors

Output: Augmented dataset with synthetic samples

foreach $f_i \in x_{minority}$ **do**
  Find the k nearest neighbors of f_i in $x_{minority}$;
  Randomly select N neighbors from the k-nearest neighbors;
  foreach *selected neighbor* x_{random} **do**
    Generate a random number η in the range [0, 1];
    Compute the synthetic sample $x_{sample} = f_i + \eta(x_{random} - f_i)$;
    Add x_{sample} to the synthetic samples set;

ECG signals are susceptible to various types of noise, including interference from external electrical devices and signal degradation due to electrode distance [37]. Gaus- sian noise was introduced to the dataset to bolster the model's resilience against noise and enhance its ability to generalize effectively to unseen data. The equation to add Gaussian noise to the data is shown in Eq. 2.

ensuring consistency in noise robustness across the training and testing phases.

Feature engineering

Feature engineering is the process of selecting the features with the most impor- tant attributes and eliminating less important features from a dataset to increase the

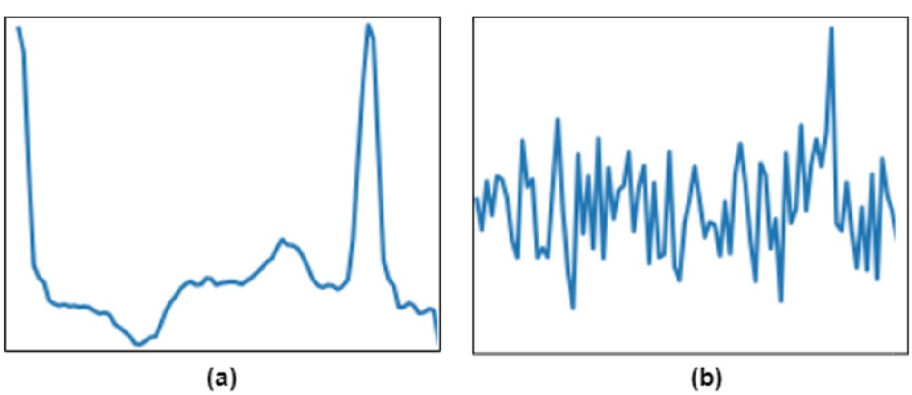

Fig. 3 ECG signal before and after adding Gaussian noise

predictive performance of machine learning methods [38]. It is a method of finding out the best subset of features necessary to train a prediction model with superior performance. The dataset used in this research has a total of 187 features for each data instance. Training ML models with this enormous number of features is time- consuming, difficult, and a likely chance of curse of dimensionality [39]. Again, not all the features are necessary to develop a proper model. Therefore, it is necessary to find the right number of features that, when used to train ML models, bring out the best performance in the model with fewer complexities.

At first, the Ordinary Least Square (OLS) regression method [40] was used to determine the number of dataset features that held significant importance. The random forest classifier determines the feature importance based on the pureness of its leaf nodes. The purity of the leaf nodes is 100% if all the nodes point to one class. Otherwise, it is impure. The feature that shows more purity, has more importance. It is noticeable from Fig. 4 that the cumulative variances in data become stable at approximately 80 features. After that, the variance change is negligible. This infers that approximately 80 along the total 187 features hold more importance in determin- ing a class of heartbeat. For a more rigorous approach, we decide on a fixed number of different feature size closer to 80 (50, 65, 80, and 95) to train the models. Now, the 3 feature engineering techniques were implemented on the dataset. For each tech- nique, the most significant 50, 65, 80, and 95 features were selected, and the dataset with the selected features was then used to train each of the models separately. The 3 techniques are briefly described as follows:

Chi-Square test

The Chi-Square Test (CST) is one of the most useful feature engineering techniques in the field of ML [41]. It carries out a statistical evaluation where deviation is calculated from the predicted distribution when the feature event is independent of the class value and feature priority is determined by observing the relationships between them [42]. The formula of CST is shown in Eq. 3. In the equation, the observed values are the total real observations that fit a particular feature i, and the expected values are the total observations that are expected to occur. The prioritized features are selected based on the best scores of χ^2. For selecting the k best features, the python SelectKBest function was applied with k= n, where n is the total number of features.

$$\times^2 = \sum_{i=1}^{n} \frac{(Observed\ Value_i - Expected\ Value_i)^2}{Expected\ Value_i} \tag{3}$$

Principal component analysis (PCA)

Principal Component Analysis (PCA) is a dimension reduction tool that prioritizes features by observing the correlation between characteristics to determine the most important features or components [43]. PCA maps the original n-dimensional con- structs into a k-dimensional construct where $k < n$ [44]. These k features

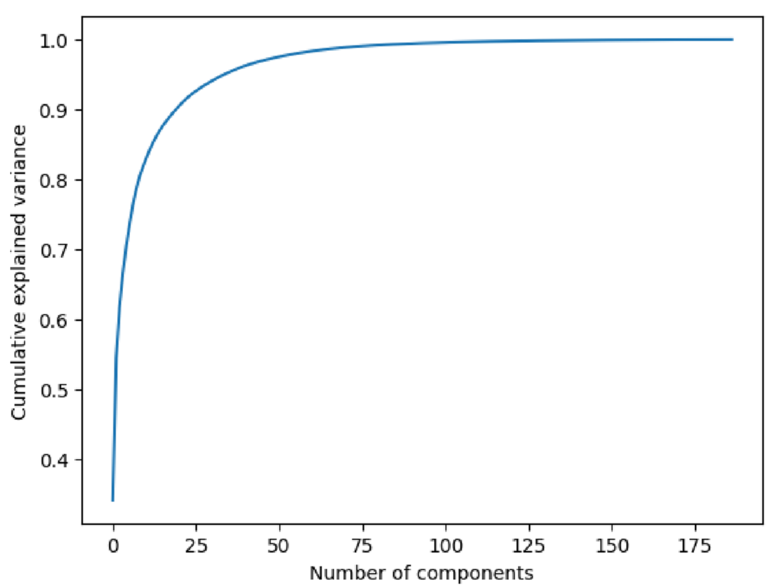

Fig. 4 Diagrammatic view of the number of features significant to the dataset

are new principal attributes that reduce the curse of dimensionality.

Recursive feature elimination (RFE)

Recursive Feature Elimination (RFE) is a wrapper technique used for removing fea- tures from training data by ranking them in the order of importance and eliminating the low-ranked features [45]. This is a recursive method that applies various ML models and determines feature importance at every iteration by removing the least important ones.

The pseudocode of feature engineering is shown in Algorithm 3.

Algorithm 3 Pseudocode for feature engineering

Input :
- n = Set of all initial features
- k = Number of features wanted
- FE = Feature Engineering Techniques (CST, PCI, RFE)

Output: R_k = Reduced feature set with k features

foreach $f_i \in n$ **do**
    relative significance of $f_i \leftarrow FE$;
    relativeValues[f_i] $\leftarrow$ relative importance of f_i;
$n \leftarrow$ sort(relativeValues[f_i]);
$R_k \leftarrow n$.select(k);
return R_k;

Development of models

In this phase, the models were developed with a Python tool called Scikit-learn [46] in the Kaggle platform. The development of models was carried out in 4 phases or techniques as seen in Fig. 1. The descriptions of the 4 techniques are given as follows:

Conventional ML model development

Technique 1 applies 5 conventional ML algorithms that have been extensively used in health informatics. The algorithms are SVM, KNN, Logistic Regression, Decision Tree, and MLP. These algorithms follow some fundamental structures that are used to carry out predictive tasks.

Bagging classifier development

Technique 2 develops a bagging classifier for prediction. A bagging classifier is an ensemble technique that integrates more than one base model on a random subset of the dataset with equal weights provided to each model and decides on a final result based on the individual predictions [47]. The bagging classifier used in this research is the random forest classifier. The Random Forest aggregates the results of several decision trees and reaches the final decision.

Boosting classifier development

Technique 3 develops a boosting classifier for prediction. A boosting classifier is an ensemble technique that combines a group of weak learners into a strong learner by reducing the error of the weak learners [48]. In this research, 4 boosting classifiers were developed namely Adaboost (AB), Gradient Boosting (GB), eXtreme Gradient Boosting (XGB), and Categorical Boosting (CB).

Proposed stack classifiers development

Ensemble learning uses many classifiers to obtain better forecasting accuracy than a single classifier; where the method known as stacking ensemble learning combines multiple weak classifiers using a meta-classifier. In this method, each of the classifiers in the first level receives the data samples as input. If the dataset has a dimension of r x c, then each classifier in the first level receives data of r x c dimensions. Then, each classifier provides its predictions. These predictions of the first level, along with the true values, are used as features in the classifier in the second level. If there are n classifiers in the first level, then the classifier in the second level will receive a dataset of r x (n +1). Lastly, the prediction of the final classifier is considered as the final result [49]. An illustration of the stack classifier mechanism is given in Fig. 5.

The proposed method is a multi-layered stack architecture where the dataset is preprocessed and then sent to base learners at level 0. In level 0, the 5 conventional ML algorithms have been kept which were used in technique 1. They are SVM, KNN, Logistic Regression, Decision Tree, and MLP. Each base model learns from the dataset independently applying its prediction method. Each base

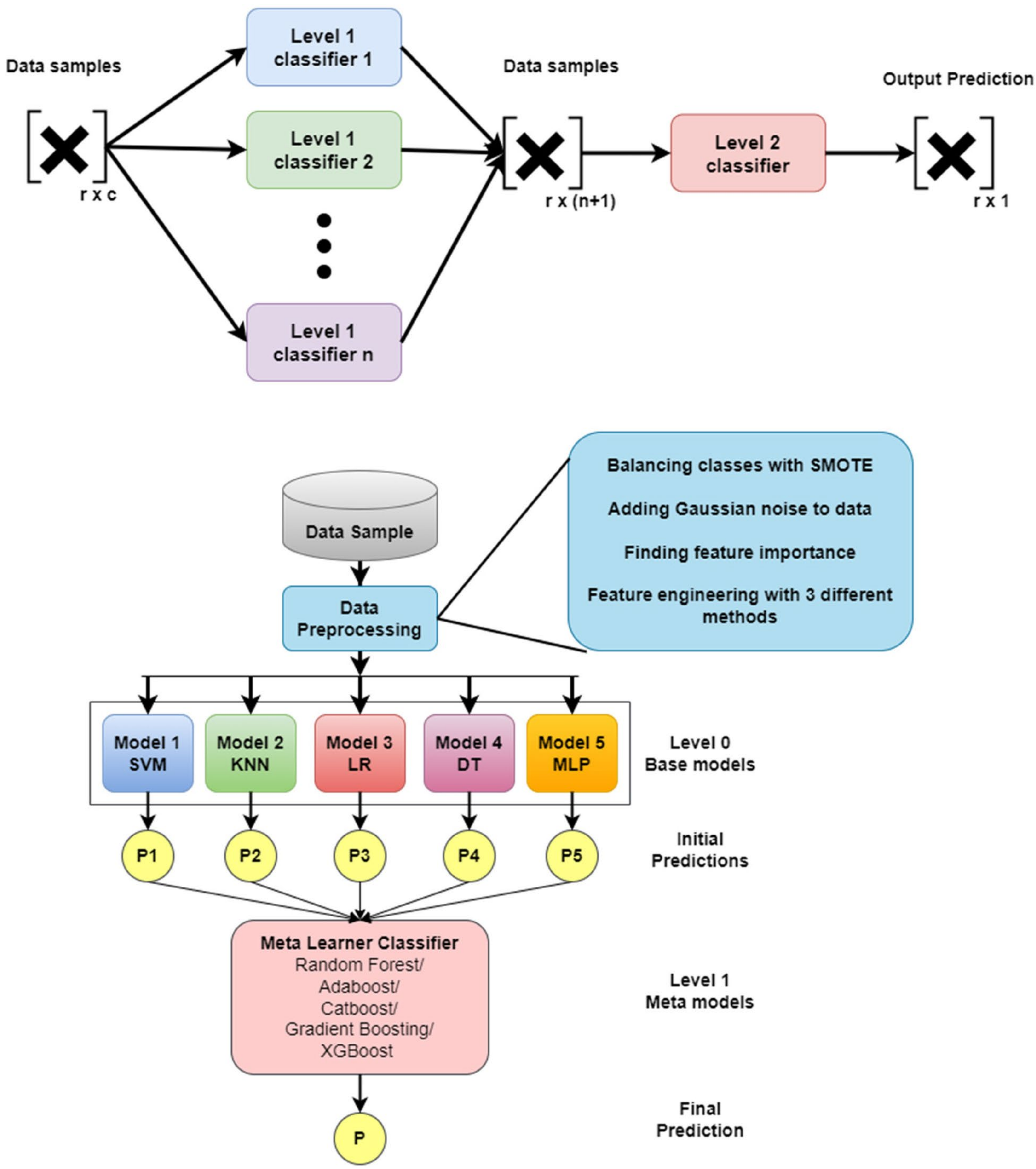

Fig. 5 Mechanism of stack classifier and the architecture of the proposed classifier

model predicts outputs which are denoted by P1, P2, P3, P4, and P5 in Fig. 5. After this, the level 1 model receives the output of these base models as their features and gives the final output. 5 different algorithms were tried as the level 1 model while keeping the same base models at level 0. This resulted in 5 different types of the proposed classifier. The models are the bagging and boosting models used in techniques 2 and 3, respectively. A conceptual view of the proposed stack ensemble classifier has been given in Fig. 5.

```
Input :
  • Processed dataset D
  • A set of ML algorithms B
  • A set of ensemble ML algorithms E
Output: A set of developed stack classifiers M
B ← [SVM, KNN, LR, MLP, DT];
E ← [RF, AB, GB, CB, XGB];
M ← [];
foreach i ∈ E do
    Make a new stack classifier S;
    foreach j ∈ B do
        Add B[j] in the level 0 of S;
    Add E[i] in the level 1 of S;
    Trained Model S' ← model_train(S, D);
    M.append(S');
```

Algorithm 4 Development of proposed stack classifier

Results

The performances of the ML models trained with different sets of features were mea- sured in terms of accuracy, precision, recall, and f1-score. The results of this rigorous evaluation are shown in Tables 1, 2, 3 and 4, where Table 1 shows the accuracy of the developed models, Table 2 shows the precision of the developed models, Table 3 shows the recall of the developed models, and Table 4 shows the f1-score of the developed models. The best performance among the models with the optimal future set has been highlighted in all the tables.

It is noticeable from Tables 1, 2, 3 and 4 that the performances of the proposed stack classi- fiers outperform other conventional techniques. Among the proposed stack classifiers, the stack classifier with the XGBoost algorithm as the meta-classifier achieved the best performance among all the other models in all 4 performance parameters with the dataset of 65 features selected by the PCA feature engineering technique. It achieved a remarkable accuracy, precision, recall, and f1-score of 99.58%, 99.578%, 99.58%, and 99.579%, respectively. Thus, it is proved that, among 187 features, 65 is the optimal number of features required to train the ML models. It is also evident that, the pro- posed stack classifier with XGBoost as the meta-classifier performs the best with the given dataset.

The f1-score of each model in predicting every class where the models were trained with 65 features extracted by the 3 different feature engineering techniques are shown in Table 5. The reason for showing the f1-score is because the f1-score is the harmonic mean of precision and recall, the other two evaluation metrics. On the other hand, accuracy alone is not a reliable evaluation metric since accuracy can be misleading sometimes [50]. The f1-score is shown for models trained with the 65 most important features since the best-performing model was obtained when the models were trained with the 65 most significant features. It is noticeable from Table 5 that most of the classes have better f1-score when the features were extracted by the PCA technique. The best f1-score per class per model is provided in bold font. From this, we can conclude with the given dataset, the PCA technique is the best in extracting the 65 most useful features for training the models.

Discussion

Three types of ensemble techniques with several classifiers were explored, trained, and tested along with conventional ML algorithms to classify heart arrhythmia from

Table 1 Comparison of accuracy of ML models utilizing different numbers of features

Type	Models	Performance with 50 Features			Performance with 65 Features			Performance with 80 Features			Performance with 95 Features		
		Chi-Square	PCA	RFE	Chi-Square	PCA	RFE	Chi-Square	PCA	RFE	Chi-Square	PCA	RFE
Conventional	SVM	86.52	93.727273	89.32	87.71	94.072727	89.23	88.39	94.209091	89.75	88.64	94.309091	89.69
	KNN	87.34	97.818182	90.41	87.92	97.809091	90.6	88.33	97.845455	90.1	88.72	97.8	89.61
	Logistic Regression	68.89	73.918182	74.15	70.39	76.090909	74.29	71.09	76.6	75.53	71.76	77.372727	75.04
	Decision Tree	79.52	98.109091	82.78	81.13	97.790909	83.7	80.91	97.881818	83.19	81.8	97.809091	83.06
	MLP	81.98	91.418182	85.79	82.13	92.254545	84.39	82.71	93.154545	84.55	84.91	92.227273	83.91
Bagging	Random Forest	83.61	99.490909	86.69	84.51	99.472727	86.73	85.19	99.509091	86.39	85.29	99.563636	86.7
Boosting	Adaboost	67.54	68.9	68.66	71.99	69.736364	67.38	70.8	70.663636	68.89	70.85	70.763636	67.36
	Gradient Boosting	84.67	89.154545	87.06	85.74	88.945455	87.26	86.61	88.781818	86.91	86.73	88.8	87.05
	XGBoost	88.12	99.145455	90.08	88.94	99.118182	90.39	89.96	99.2	90.5	89.75	99.290909	90.48
	Catboost	89.68	98.4	91.32	90.75	98.445455	91.75	90.78	98.527273	91.36	90.39	98.509091	91.45
Proposed Stack Classifier	Meta Learner Adaboost	69.1	97.127273	83.62	81.95	97.1	71.31	67.44	96.318182	70.96	84.45	86	57.76
	Meta Learner Random Forest	79.42	99.418182	83.64	81.13	99.509091	83.81	80.81	99.4	82.98	81.9	99.518182	82.97
	Meta Learner XGBoost	79.57	99.536364	83.72	81.67	**99.581818**	82.7	81.41	99.5	83.02	82.45	99.545455	83.78
	Meta Learner Catboost	79.89	99.481818	83.73	82.24	99.536364	85.04	82.11	82.65	84.01	82.65	99.572727	84.63
	Meta Learner Gradient Boosting	79.67	99.245455	82.82	82.68	99.390909	83.2	81.45	99.318182	83.53	82.41	99.354545	83.95

Table 2 Comparison of precision of ML models utilizing different numbers of features

Type	Models	Performance with 50 Features			Performance with 65 Features			Performance with 80 Features			Performance with 95 Features		
		Chi-Square	PCA	RFE	Chi-Square	PCA	RFE	Chi-Square	PCA	RFE	Chi-Square	PCA	RFE
	SVM	86.94485	93.827373	89.783716	88.031384	94.158568	89.702257	88.699315	94.309675	90.28322	88.894475	94.407348	90.416207
	KNN	88.220718	97.871204	91.159457	88.745439	97.859004	91.312866	89.160959	97.892947	90.940123	89.434908	97.844466	90.537157
Conventional	Logistic Regression	68.35774	73.920882	74.154286	69.921345	76.162387	74.41881	70.856974	76.691972	75.92459	71.615865	77.643318	75.415093
	Decision Tree	84.47964	98.115206	86.490791	85.043538	97.802999	86.846444	84.953147	97.892351	86.627764	85.114829	97.813428	86.502323
	MLP	82.075598	91.459967	86.282247	82.32344	92.29576	84.477325	83.136763	93.180309	84.959322	85.077082	92.276122	84.301534
Bagging	Random Forest	88.532088	99.491581	90.210903	88.890848	99.473322	90.181189	89.330666	99.509683	90.040169	89.412383	99.563449	90.073922
	Adaboost	66.246866	68.413491	68.02799	70.755199	69.26663	66.597251	69.67253	70.177415	68.378933	69.68451	70.368051	66.574322
Boosting	Gradient Boosting	84.945001	89.367656	87.519153	86.110471	89.23484	87.710009	86.970485	89.03178	87.584672	87.212089	89.054753	87.618653
	XGBoost	89.886282	99.145929	91.696595	90.416543	99.120371	92.118164	91.281987	99.203167	92.038688	91.179365	99.293223	92.062012
	Catboost	90.549455	98.409498	92.274737	91.511813	98.454101	92.597914	91.6794	98.530935	92.397249	91.240803	98.51182	92.39095
	Meta Learner Adaboost	81.785852	97.310323	87.689143	87.103423	97.148414	65.82709	74.216292	96.450874	64.838822	87.042139	87.39453	49.175009
Proposed Stack Classifier	Meta Learner Random Forest	87.026433	99.417122	88.690556	86.972746	99.507156	88.618117	87.114547	99.397555	88.068773	87.492799	99.516615	88.425236
	Meta Learner XGBoost	86.542757	99.535165	88.459659	87.090268	**99.578936**	87.631411	87.177593	99.498847	88.098817	87.344629	99.544351	88.429219
	Meta Learner Catboost	87.012061	99.480756	88.451724	87.42107	99.533393	89.377497	87.640441	87.676867	88.569579	87.676867	99.570682	88.784529
	Meta Learner Gradient Boosting	86.631502	99.243626	88.036039	87.545036	99.38699	88.22027	87.073849	99.31472	88.227192	87.497407	99.353075	88.511089

Table 3 Comparison of recall of ML models utilizing different numbers of features

Type	Models	Performance with 50 Features			Performance with 65 Features			Performance with 80 Features			Performance with 95 Features		
		Chi-Square	PCA	RFE	Chi-Square	PCA	RFE	Chi-Square	PCA	RFE	Chi-Square	PCA	RFE
	SVM	86.52	93.762221	89.32	87.71	94.106501	89.23	88.39	94.238204	89.75	88.64	94.339249	89.69
	KNN	87.34	97.812095	90.41	87.92	97.8021	90.6	88.33	97.838372	90.1	88.72	97.792078	89.61
Conventional	Logistic Regression	68.89	73.938747	74.15	70.39	76.129881	74.29	71.09	76.640621	75.53	71.76	77.418915	75.04
	Decision Tree	79.52	98.114136	82.78	81.13	97.796906	83.7	80.91	97.887744	83.19	81.8	97.814257	83.06
	MLP	81.98	91.457907	85.79	82.13	92.275064	84.39	82.71	93.17639	84.55	84.91	92.255277	83.91
Bagging	Random Forest	83.61	99.487168	86.69	84.51	99.46915	86.73	85.19	99.505436	86.39	85.29	99.559822	86.7
	Adaboost	67.54	68.831092	68.66	71.99	69.69659	67.38	70.8	70.603963	68.89	70.85	70.737285	67.36
Boosting	Gradient Boosting	84.67	89.179487	87.06	85.74	88.97516	87.26	86.61	88.812292	86.91	86.73	88.830666	87.05
	XGBoost	88.12	99.144949	90.08	88.94	99.117907	90.39	89.96	99.200128	90.5	89.75	99.290137	90.48
	Catboost	89.68	98.395487	91.32	90.75	98.442539	91.75	90.78	98.524722	91.36	90.39	98.502701	91.45
	Meta Learner Adaboost	69.1	97.116066	83.62	81.95	97.071649	71.31	67.44	96.353562	70.96	84.45	85.726374	57.76
Proposed Stack Classifier	Meta Learner Random Forest	79.42	99.416317	83.64	81.13	99.507227	83.81	80.81	99.398341	82.98	81.9	99.513297	82.97
	Meta Learner XGBoost	79.57	99.53388	83.72	81.67	**99.580174**	82.7	81.41	99.500117	83.02	82.45	99.544735	83.78
	Meta Learner Catboost	79.89	99.47916	83.73	82.24	99.534381	85.04	82.11	82.65	84.01	82.65	99.569585	84.63
	Meta Learner Gradient Boosting	79.67	99.24448	82.82	82.68	99.389632	83.2	81.45	99.317228	83.53	82.41	99.354305	83.95

Table 4 Comparison of f1-score of ML models utilizing different numbers of features

Type	Models	Performance with 50 Features			Performance with 65 Features			Performance with 80 Features			Performance with 95 Features		
		Chi-Square	PCA	RFE	Chi-Square	PCA	RFE	Chi-Square	PCA	RFE	Chi-Square	PCA	RFE
	SVM	86.556822	93.7727	89.368473	87.696523	94.115927	89.283428	88.393808	94.254474	89.815408	88.661491	94.354261	89.789576
	KNN	87.353429	97.808338	90.425689	87.929686	97.799598	90.614523	88.343102	97.836295	90.133428	88.718124	97.789152	89.648468
Conventional	Logistic Regression	68.2951	73.904847	73.841013	69.950198	76.097522	74.051543	70.758823	76.616605	75.517921	71.474733	77.429435	75.06395
	Decision Tree	79.816924	98.083679	83.054123	81.436911	97.755537	83.938797	81.025421	97.848795	83.467625	82.015456	97.772576	83.350057
	MLP	81.967865	91.450808	85.83568	82.115157	92.277215	84.344932	82.78195	93.173759	84.635942	84.963844	92.258921	83.990735
Bagging	Random Forest	83.912881	99.488969	86.969604	84.807135	99.470771	86.984807	85.499267	99.507199	86.646247	85.631305	99.561437	86.918313
	Adaboost	66.535627	68.564091	68.032018	71.113155	69.398873	66.620706	69.784861	70.207459	68.39929	69.788981	70.491034	66.533863
Boosting	Gradient Boosting	84.712854	89.218927	87.139663	85.794532	89.037274	87.333499	86.695533	88.86352	87.000165	86.844668	88.886465	87.126998
	XGBoost	88.31604	99.14223	90.198261	89.067378	99.115116	90.529592	90.084114	99.197932	90.604687	89.888732	99.288985	90.601472
	Catboost	89.771967	98.395892	91.40443	90.828578	98.443625	91.825503	90.886251	98.523118	91.446433	90.45234	98.501311	91.520324
	Meta Learner Adaboost	66.259593	97.139316	83.491433	82.418851	97.068497	65.822072	61.841142	96.305269	65.388194	84.516743	85.696298	47.314263
Proposed Stack Classifier	Meta Learner Random Forest	79.739615	99.416084	83.817597	81.313482	99.506713	83.939424	80.850903	99.396677	83.110908	82.018304	99.51469	83.198261
	Meta Learner XGBoost	79.671042	99.533896	83.769362	81.769336	**99.579102**	82.612813	81.373398	99.498307	83.09315	82.432804	99.543651	83.903346
	Meta Learner Catboost	80.124826	99.479212	83.754863	82.382379	99.53323	85.194969	82.155406	82.71726	84.140731	82.71726	99.569837	84.765889
	Meta Learner Gradient Boosting	79.726252	99.24296	82.792387	82.831668	99.387904	83.183019	81.353922	99.315578	83.5622	82.440377	99.353212	83.974143

Table 5 F1-score of each model per class feature-engineered with 3 methods and 65 features

Model	FE Method	N	V	S	F	M
SVM	CHI	81.65	83.04	88.56	87.6	97.63
	PCA	**88.65**	**91.64**	**96.04**	**95.48**	**98.77**
	RFE	81.66	85.6	90.99	90.53	97.64
KNN	CHI	83.66	85.74	87.57	85.24	97.42
	PCA	**95.61**	**97.35**	**97.78**	**99.16**	**99.1**
	RFE	86.51	89.93	91.16	87.58	97.9
LR	CHI	51.41	67.81	62.2	80.63	87.7
	PCA	**59.56**	**73.63**	**72.59**	**83.49**	91.23
	RFE	58.53	68.63	71.09	80.58	**91.43**
DT	CHI	73.18	79.97	81.88	77	95.16
	PCA	**94.63**	**98.53**	**97.08**	**99.67**	**98.87**
	RFE	75.64	80.75	86.74	81.26	95.3
MLP	CHI	70.63	77.62	82.98	85.73	93.61
	PCA	**85.7**	**90.02**	**92.8**	**95.56**	**97.3**
	RFE	75.45	79.23	85.99	86.85	94.21
RF	CHI	75.99	81.41	89.19	80.12	97.33
	PCA	**98.76**	**99.69**	**99.36**	**99.84**	**99.7**
	RFE	78.61	84.29	91.41	83.05	97.55
AB	CHI	46.07	64.83	**70.85**	**84.08**	**89.73**
	PCA	**51.1**	**68.26**	59.73	83.03	84.87
	RFE	43.46	63.68	64.16	78.62	83.19
GB	CHI	76.16	82.72	86.96	87.71	95.43
	PCA	**80.14**	**86.86**	**88.95**	**93.22**	**96.03**
	RFE	79.61	84.51	88.31	88.24	96
XGB	CHI	82.88	86.22	89.9	88.35	97.98
	PCA	**97.95**	**99.16**	**99.04**	**99.82**	**99.61**
	RFE	83.91	87.32	93.29	89.49	98.64
CB	CHI	85.15	87.79	91.78	91.18	98.24
	PCA	**96.88**	**98.36**	**98.35**	**99.19**	**99.44**
	RFE	86.58	90	92.81	91.36	98.38
RF-Stack	CHI	72.61	78.85	85.1	72.47	97.54
	PCA	**98.85**	**99.58**	**99.25**	**99.98**	**99.88**
	RFE	75.61	80.42	88.6	77.19	97.88
AB-Stack	CHI	72.99	79.03	83.99	79.09	97
	PCA	**96.31**	**97.59**	**95.5**	**99.93**	96.01
	RFE	59.92	80.54	91.15	0	**97.5**
GB-Stack	CHI	74.66	80.34	86.08	75.33	97.75
	PCA	**98.62**	**99.67**	**99.04**	**99.8**	**99.81**
	RFE	75.35	80.18	88.47	74.13	97.78
XGB-Stack	CHI	73.57	79.51	85.41	72.58	97.78
	PCA	**98.98**	**99.78**	**99.36**	**99.96**	**99.81**
	RFE	74.6	80.58	88.24	73.84	98.21
CB-Stack	CHI	74.59	79	85.55	75.36	97.41
	PCA	**98.89**	**99.71**	**99.25**	**99.96**	**99.86**
	RFE	76.53	79.87	90.61	80.65	98.32

ECG signals in this study. Most previous works utilized only the conventional algorithms and very few studies focused on the stack classifier, a state-of-the-art technology. Therefore, the proposed technique based on a stack classifier where conventional, bag- ging, and boosting models were all integrated to achieve a better prediction is a novel contribution in this domain.

Again, most of the previous studies did not utilize feature engineering techniques to reduce the number of features and determine the optimal number of features. There- fore, in this research, 3 different feature engineering techniques (Chi-square, PCA, and RFE) were applied to determine the optimal number of features necessary to train ML models with a satisfactory performance avoiding any complexities like curse of dimensionality, training time, memory requirements and so on.

Finally, the performances of each of the developed models were evaluated based on accuracy, precision, recall, and f1-score to validate the efficacy and effectiveness of the developed models. It was found that the stack classifier that was developed using XGBoost as the meta-classifier and trained with the dataset consisting of 65 features selected by the PCA method outperformed not only all other models but also the previous works carried out with the same dataset. A performance comparison with the previous works is given in Table 6.

Novelty of the study

The proposed research presents a novel and advanced approach to heart arrhythmia diagnosis by developing a sophisticated stack classifier system that leverages cutting- edge ensemble machine-learning techniques. The model is designed with XGBoost as the meta-classifier, a robust and highly effective algorithm known for its strong perfor- mance in classification tasks. This approach is further enhanced by the incorporation of advanced feature engineering techniques, including Principal Component Analysis (PCA), to extract and refine critical features from electrocardiogram (ECG) signals. One of the primary innovations of this research is the automation of the heart arrhythmia diagnosis process, significantly reducing the need for human intervention. Traditionally, the analysis of ECG data has relied heavily on cardiologists and clinical experts, which can be time-consuming and prone to human error. By utilizing this automated system, the research addresses these limitations, offering a faster, more accurate, and reliable alternative for detecting arrhythmias.

The ensemble machine-learning techniques employed in this study, particularly the use of XGBoost, offer substantial improvements over conventional machine-learning methods. XGBoost's ability to handle large datasets, its superior speed, and its high predictive power make it an ideal choice for this application. Moreover, the integration of PCA allows for the selection of 65 optimal features from the ECG data, ensuring that the classifier is trained with the most relevant information, thus enhancing its performance.

The results of this study are highly promising, with the stack classifier achieving exceptional performance metrics: 99.58% accuracy, 99.57% precision, 99.58% recall, and 99.57% F1-score. These results not only demonstrate the effectiveness of the pro- posed model but also its potential to revolutionize the field of medical diagnostics, particularly in the area of arrhythmia detection. By outperforming other conventional machine-learning and ensemble algorithms, the proposed stack classifier sets a new benchmark for accuracy and reliability in this domain.

This research provides a significant contribution to the field of automated medical diagnostics. The novel stack classifier system, combining XGBoost with PCA-driven feature engineering, offers a powerful tool for the accurate and timely diagnosis of heart arrhythmias. This advancement has the potential to greatly improve patient outcomes by facilitating early detection and treatment, ultimately reducing the mor- tality rate associated with heart arrhythmias. The system's ability to operate with minimal human intervention also makes it highly scalable and adaptable for use in various clinical settings, further enhancing its practical application in healthcare.

Table 6 Comparative analysis with existing ML approaches

Authors	Modality	Models used	Performance Results	Dataset used
Melgani et al. [20]	Classification of Electrocardiogram Signals	SVM, PSO	Overall accuracy of 89.72%	MIT-BIH dataset
Kumar et al. [21]	Classification of Electrocardiogram Signals	Random Forest	92.16% accuracy	MIT-BIH dataset
Park et al. [22]	Arrhythmia detection and classification	KNN	97.22% sensitivity and 97.4% specificity for heartbeat detection. 97.1% sensitivity and 96.9% specificity for classification.	MIT-BIH dataset
Ubeyli et al. [24]	Classification of Electrocardiogram Signals	RNN, Eigenvector methods	98.06% accuracy	MIT-BIH dataset
Guler et al. [25]	Classification of Electrocardiogram Signals	FFNN	96.94% accuracy	MIT-BIH dataset
Ardeti et al. [23]	Outlier Detection and Feature Ranking in ECG Beats	Hybrid Ensemble Method	95.81% accuracy on ensemble deep learning accuracy of 95.81% and 94.47% accuracy on ensemble SVM.	MIT-BIH dataset
Li et al. [26]	Classification of ECG signals from 2D data	DSE-RESNET	81.7% f1-score	CPSC2018 dataset
Ramkumar et al. [27]	Classification of ECG signals	Hybrid model of Auto Encoder and Bi-LSTM	98.33% positive predictive value, 97.15% accuracy, 96.22% specificity, 99.43% sensitivity	New N beat, AFIB beat
Madan et al. [28]	ECG Arrhythmia Classification	CNN-LSTM Ensemble	98.7% accuracy	MIT-BIH dataset
This research	Classification of Arrhythmia from ECG signals by adopting the best model	Proposed Stack Classifiers	99.58% Accuracy, 99.57% Precision, 99.58% Recall, 99.57% F1-score	MIT-BIH dataset

Limitations and future works

This research has certain limitations. Firstly, the classification methods based on image or signal processing techniques were not explored. Secondly, this study's classification of heart arrhythmia was based solely on 2D data. Thirdly, Transfer learning models, such as pre-trained CNN models, were not investigated due to their reliance on image data. Fourthly, explainable AI, which explains and justifies the AI system predictions, was not explored in this research. Hence, future works may focus on (a) employing image and signal processing techniques with larger sample sizes for the detection of heart arrhythmia from ECG signals, (b) categorizing heart arrhythmia using image data, (c) training and evaluating untested models, and their performances, (d) incorporating explainable AI with the current research, and (e) conducting a more detailed analysis of the complexities of the models based on time and memory.

Conclusion

Machine learning plays a pivotal role in the precise and timely diagnosis of heart abnormalities, especially in detecting arrhythmias. Its capacity to continuously analyze electrocardiogram (ECG) data allows for the early identification of patterns indicative of arrhythmias, enabling swift intervention. Healthcare professionals may incorporate these machine learning models into their daily practice, enhancing patient care through real-time monitoring and early warning systems.

The need for effective and unbiased analysis of large-scale medical data drives the growing interest in ECG-based cardiac arrhythmia analysis for heart-related studies. Early recognition of heart problems is crucial for prompt treatment and reduced mor- tality rates. However, manual diagnosis of heart conditions is time-consuming and requires expert operators due to the intricacies of the heart's functions. Thus, the methodology described in this article can be a benchmark for accurate and precise heart arrhythmia classification from ECG signals. The high performances achieved by the proposed methodology demonstrate the validity of this study.

Authors' contributions

R. J. and M.N.I conceptualize the research. R.J. and M.M.I. review the related studies. R. J. and M.S.I. contribute in methodology. Model development and Results analysis is carried out by R.J.; R.J., M.S.I. and M.M.I. contribute in writing and editing the article. The research was supervised by the M.N.I. All authors reviewed the manuscript.

Funding

Not applicable.

Data availability

The open source MIT-BIH arrhythmia database is used in this article, which is available at https://doi.org/10.13026/C2F305.

Declarations

Ethics approval and consent to participate

We confirm that ethical approval was applied for conducting this research. No human data, human tissue, or any clinical data were collected for this study. Therefore, there were no requirements to have formal approval.

Consent for publication

Not applicable.

Competing interests

The authors declare no competing interests.

Received: 25 November 2024 Accepted: 17 March 2025
Published online: 07 April 2025

References

1. Iaizzo PA. General features of the cardiovascular system. Handbook of Cardiac Anatomy, Physiology, and Devices. 2015; pp. 3–12.
2. Qiu W, Quan C, Yu Y, Kara E, Qian K, Hu B, et al. Federated abnormal heart sound detection with weak to no labels. Cyborg and Bionic Systems. 2024;5:0152.
3. Bahnson TD, Grant AO. To be or not to be in normal sinus rhythm: what do we really know? Ann Intern Med. 2004;141(9):727–9.
4. Cox JL. Surgery for cardiac arrhythmias. Curr Probl Cardiol. 1983;8(4):3–60.
5. Salem M, Taheri S, Yuan JS. ECG arrhythmia classification using transfer learn- ing from 2-dimensional deep CNN features. In: 2018 IEEE biomedical circuits and systems conference (BioCAS). Ieee; 2018. pp. 1–4
6. Li C, Bian Y, Zhao Z, Liu Y, Guo Y. Advances in biointegrated wearable and implantable optoelectronic devices for cardiac healthcare. Cyborg and Bionic Systems. 2024;5:0172.
7. Zhang Z, Wu K, Wu Z, Xiao Y, Wang Y, Lin Q, et al. A case of pioneering subcu- taneous implantable cardioverter defibrillator intervention in Timothy syndrome. BMC Pediatr. 2024;24(1):729.
8. Bing P, Liu W, Zhai Z, Li J, Guo Z, Xiang Y, et al. A novel approach for denoising electrocardiogram signals to detect cardiovascular diseases using an efficient hybrid scheme. Frontiers in Cardiovascular Medicine. 2024;11:1277123.
9. Jahangir R, Mohim NS, Mumu AA, Naim M, Ashraf A, Syed MA, Development of a Smart Infant Monitoring System for Working Mothers. In, et al. IEEE 11th Region 10 Humanitarian Technology Conference (R10-HTC). IEEE. 2023;2023:37–42.
10. Sundnes J, Lines GT, Cai X, Nielsen BF, Mardal KA, Tveito A. Computing the electrical activity in the heart. vol. 1. Springer Science & Business Media; 2007.
11. Jahangir R, Mohim NS, Khan NI, Akhtaruzzaman M, Proposing IMN, Architectures NRNN, for Infant Cry Detection in Domestic Context. In,. IEEE 11th Region 10 Humanitarian Technology Conference (R10-HTC). IEEE. 2023;2023:7–12.
12. Jahangir R, Sakib T, Haque R, Kamal M. A Performance Analysis of Brain Tumor Classification from MRI Images using Vision Transformers and CNN-based Clas- sifiers. In: 2023 26th International Conference on Computer and Information Technology (ICCIT). IEEE; 2023. pp. 1–6.
13. Jahangir R, Sakib T, Juboraj MFUA, Feroz SB, Sharar MMI. Brain Tumor Clas- sification on MRI Images with Big Transfer and Vision Transformer: Comparative Study. In: 2023 IEEE 9th International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE). IEEE; 2023. pp. 46–51.
14. Rakin FI, Ahmmed T, Kabir R, Hasan MS, Ramadan STY, Sakib T, et al. Predic- tive Analytics for Floods in Bangladesh: A Comparative Exploration of Machine Learning and Deep Learning Classifiers. In: 2023 26th International Conference on Computer and Information Technology (ICCIT). IEEE; 2023. pp. 1–6.
15. Jahangir R, Sakib T, Baki R, Hossain MM. A Comparative Analysis of Potato Leaf Disease Classification with Big Transfer (BiT) and Vision Transformer (ViT) Models. In: 2023 IEEE 9th International Women in Engineering (WIE) Confer- ence on Electrical and Computer Engineering (WIECON-ECE). IEEE; 2023. pp. 58–63.

16. Jahangir R, Juboraj MFUA, Islam MT, Hossain MM, Khandaker NA, Sharar MMI. A Conceptual Framework of an Automated Mosquito Control in Drainage Systems for Combating Dengue in Bangladesh. In: 2023 26th International Conference on Computer and Information Technology (ICCIT). IEEE; 2023. pp. 1–6.
17. Nasiri JA, Naghibzadeh M, Yazdi HS, Naghibzadeh BECG, arrhythmia classification with support vector machines and genetic algorithm. In,. Third UKSim European Symposium on Computer Modeling and Simulation. IEEE. 2009;2009:187–92.
18. Islam MS, Islam MN, Hashim N, Rashid M, Bari BS, Al FF. New hybrid deep learning approach using BiGRU-BiLSTM and multilayered dilated CNN to detect arrhythmia. IEEE Access. 2022;10:58081–96.
19. Islam MS, Hasan KF, Sultana S, Uddin S, Quinn JM, Moni MA, et al. HARDC: A novel ECG-based heartbeat classification method to detect arrhythmia using hierarchical attention based dual structured RNN with dilated CNN. Neural Netw. 2023;162:271–87.
20. Melgani F, Bazi Y. Classification of electrocardiogram signals with support vector machines and particle swarm optimization. IEEE Trans Inf Technol Biomed. 2008;12(5):667–77.
21. Kumar RG, Kumaraswamy Y, et al. Investigating cardiac arrhythmia in ECG using random forest classification. Int J Comput Appl. 2012;37(4):31–4.
22. Park J, Lee K, Kang K. Arrhythmia detection from heartbeat using k-nearest neighbor classifier. In: 2013 IEEE International Conference on Bioinformatics and Biomedicine. IEEE; 2013. p. 15–22.
23. Ardeti VA, Kolluru VR, Varghese GT, Patjoshi RK. An Outlier Detection and Feature Ranking based Ensemble Learning for ECG Analysis. Int J Adv Comp Sci Appl. 2022;13(6):727–37.
24. Übeyli ED. Combining recurrent neural networks with eigenvector methods for classification of ECG beats. Dig Sig Process. 2009;19(2):320–9.
25. Güler I, Übeylı ED. ECG beat classifier designed by combined neural network model. Pattern recognition. 2005;38(2):199–208.
26. Li J, Pang SP, Xu F, Ji P, Zhou S, Shu M. Two-dimensional ECG-based cardiac arrhythmia classification using DSE-ResNet. Scientific Reports. 2022;12(1):14485.
27. Ramkumar M, Kumar RS, Manjunathan A, Mathankumar M, Pauliah J. Auto- encoder and bidirectional long short-term memory based automated arrhythmia classification for ECG signal. Biomed Signal Process Control. 2022;77: 103826.
28. Madan P, Singh V, Singh DP, Diwakar M, Pant B, Kishor A. A hybrid deep learning approach for ECG-based arrhythmia classification. Bioengineering. 2022;9(4):152.
29. Ganaie MA, Hu M, Malik A, Tanveer M, Suganthan P. Ensemble deep learning: A review. Eng Appl Artif Intell. 2022;115: 105151.
30. Alfred R, Obit JH. The roles of machine learning methods in limiting the spread of deadly diseases: A systematic review. Heliyon. 2021;7(6):e07371.
31. Moody GB, Mark RG. The impact of the MIT-BIH arrhythmia database. IEEE Eng Med Biol Mag. 2001;20(3):45–50.
32. Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, et al. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. Circul. 2000;101(23):e215–20.
33. Kachuee M, Fazeli S, Sarrafzadeh M, Ecg heartbeat classification: A deep trans- ferable representation. In,. IEEE international conference on healthcare informatics (ICHI). IEEE. 2018;2018:443–4.
34. Hebbar KA, Hueston WJ. Management of common arrhythmias: Part I. Supraventricular arrhythmias American family physician. 2002;65(12):2479.
35. John RM, Tedrow UB, Koplan BA, Albert CM, Epstein LM, Sweeney MO, et al. Ventricular arrhythmias and sudden cardiac death. The Lancet. 2012;380(9852):1520–9.
36. Blagus R, Lusa L. SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics. 2013;14:1–16.
37. Rahman S, Karmakar C, Natgunanathan I, Yearwood J, Palaniswami M. Robust- ness of electrocardiogram signal quality indices. J R Soc Interface. 2022;19(189):20220012.
38. Turner CR, Fuggetta A, Lavazza L, Wolf AL. A conceptual basis for feature engineering. J Syst Softw. 1999;49(1):3–15.
39. Verleysen M, Franc¸ois D. The curse of dimensionality in data mining and time series prediction. In: International work-conference on artificial neural networks. Springer; 2005. pp. 758–770.
40. Craven B, Islam SM. Ordinary least-squares regression. The SAGE dictionary of quantitative management research. 2011; pp. 224–228.
41. Tallarida RJ, Murray RB, Tallarida RJ, Murray RB. Chi-square test. Manual of pharmacologic calculations: with computer programs. 1987; pp. 140–142.
42. Thaseen IS, Kumar CA. Intrusion detection model using fusion of chi-square feature selection and multi class SVM. Journal of King Saud University-Computer and Information Sciences. 2017;29(4):462–72.
43. Abdi H, Williams LJ. Principal component analysis. Wiley interdisciplinary reviews: computational statistics. 2010;2(4):433–59.
44. Zhao H, Zheng J, Xu J, Deng W. Fault diagnosis method based on principal com- ponent analysis and broad learning system. IEEE Access. 2019;7:99263–72.
45. Chen Xw, Jeong JC. Enhanced recursive feature elimination. In: Sixth inter- national conference on machine learning and applications (ICMLA 2007). IEEE; 2007. pp. 429–435.
46. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
47. Yaman E, Subasi A, et al. Comparison of bagging and boosting ensemble machine learning methods for automated EMG signal classification. BioMed Res Int. 2019;2019:9152506.
48. Chen W, Lei X, Chakrabortty R, Pal SC, Sahana M, Janizadeh S. Evaluation of different boosting ensemble machine learning models and novel deep learn- ing and boosting framework for head-cut gully erosion susceptibility. J Environ Manage. 2021;284: 112015.
49. Oyewola DO, Dada EG, Ndunagu JN. A novel hybrid walk-forward ensemble optimization for time series cryptocurrency prediction. Heliyon. 2022;8(11):e11862.
50. Jahangir R. CNN-SCNet: A CNN net-based deep learning framework for infant cry detection in household setting. Engineering Reports. 2023;pp. e12786.

Publisher's Note