Computers and Electrical Engineering 102 (2022) 108158

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

A tree classifier based network intrusion detection model for Internet of Medical Things☆

Karan Gupta [a], Deepak Kumar Sharma [b,*], Koyel Datta Gupta [c], Anil Kumar [d]

[a] Department of Information Technology, Netaji Subhas University of Technology, New Delhi, India
[b] Department of Information Technology, Indira Gandhi Delhi Technical University for Women, Delhi, India
[c] Department of Computer Science and Engineering, Maharaja Surajmal Institute of Technology, New Delhi, India
[d] Data Science Research Group, School of Computing, DIT University, Dehradun, India

ARTICLE INFO

Keywords:
Internet of Medical Things
Intrusion detection
Tree classifier
Security
Estimator

ABSTRACT

Healthcare is one of the key areas of prospect for the Internet of Things (IoT). To facilitate better medical services, enormous growth in the field of the Internet of Medical Things (IoMT) is observed recently. Despite the numerous benefits, the cyber threats on connected healthcare devices can compromise privacy and can also cause damage to the health of the concerned patient. The massive demand for IoMT devices with seamless and effective medical facilities for the large-scale population requires a robust secured model to ensure the privacy and safety of patients in this network. However, designing security models for IoMT networks is very challenging. An effort has been made in this work, to design a tree classifier-based network intrusion detection model for IoMT networks. The proposed system effectively reduces the dimension of the input data to speed up the anomaly detection procedure while maintaining a very high accuracy of 94.23%.

1. Introduction

Revolutionary progress of IoT in different fields including healthcare has made the dispense of services a lot more effective and expedient. Remote patient monitoring systems using sensors have made the medical treatment convenient and precise without requiring the presence of medical practitioners at the site [1,2]. The IoMT technology has empowered the medical community to provide several facilities like early disease diagnosis, increased productivity, remote monitoring of health parameters to a large number of population. The early detection of diseases using IoMT devices can save several lives and even reduce the medical costs also. The patient information collected through the medical devices can also be saved in the IoMT cloud database for future use. However, cyber-attacks are posing serious threats to digital healthcare monitoring and record keeping.

The security breach in IoMT devices and networks can cause interference in disease diagnosis, delayed communication and also cause loss of patients' private information. The security threats related to IoMT devices and connections are escalating owing to the expertise of the hackers. Manipulation of medical devices and unauthorized access of health records through internet can be caused by diverse malicious malwares. The mode and nature of such attacks are changing rapidly and the existing procedures are not adequate for malware identification and respective analysis. Most of the cyber attacks result in Distributed denial of service

☆ This paper is for special section VSI-aihc. Reviews were processed by Guest Editor Dr. Deepak Gupta and recommended for publication.

* Corresponding author.
E-mail addresses: karankg1997@gmail.com (K. Gupta), dk.sharma1982@yahoo.com (D.K. Sharma), koyel.dg@msit.in (K. Datta Gupta), dahiyaanil@yahoo.com (A. Kumar).

https://doi.org/10.1016/j.compeleceng.2022.108158
Received 31 January 2022; Received in revised form 30 May 2022; Accepted 2 June 2022
Available online 17 June 2022

(DDoS), denying access of devices or data to legitimate users. Hence, it is extremely important to identify any kind of unauthorized invasion in the IoMT system.

Several security mechanisms have been developed in past to protect IoT system including authentication, encryption and intrusion detection [3–5]. An intrusion detection system (IDS) [6] can identify an attack by examining system variables and possible attack signs or by detecting deviation from normal behaviour. Malicious activities can steal or modify health data either by intercepting the information during transmission in wireless medium or by accessing the cloud servers in an unauthorized manner. Attackers may also intervene with the diagnosis procedure. All these security breaches have a deep and long-lasting impact on the IoMT environment. This requires designing of a time efficient accurate IDS for the IoMT system.

In recent times researchers have been focusing specifically on Machine Learning (ML) techniques for detecting intrusions judiciously. In spite of several existing data analysis and statistical techniques for IDS, identifying new malicious activities is difficult. However, ML algorithms has the ability to recognize the dynamic changes in these activities' signatures. Machine Learning based models can be deployed in IDS for determining both network level and host level intrusions. Models equipped with ML can even identify unforeseen activities and classify the already detected malicious activities. In this paper, a novel lightweight model is proposed for effectively identifying intrusions in IoMT systems using anomaly detection. The model consists of a dimension reduction phase followed by an anomaly detection successively followed by a classification process. The main contributions of this work are mentioned below:

1. The input dataset is balanced by data augmentation to ensure sufficient availability of data for both normal and anomalous category.
2. IoMT datasets consists of vast amount of data with many features. Some features might not have much impact on the detection mechanism. To ensure that the model is able to determine the outcome within a short span of time, dimension reduction is applied.
3. A model based on random forest technique combined with Grid Search CV is followed by best estimator built, to determine anomalous data.
4. In addition, the model is evaluated with deranged datasets having large number of normal data or large number of anomalous data.

The remaining part of this paper is organized as follows. Section 2 comprises of related work followed by proposed work in Section 3. Experimentation and Results are provided in Section 4, and discussions in Section 5. Finally, Section 6 concludes the work.

2. Related work

In recent work throughout the years, numerous approaches have been proposed for building an Intrusion Detection system, and the following are some of the past work which has been done on IoMT Networks. Rani [7], proposed a cloud-based healthcare system, where authorization was given to only valid users. The system uses Support Vector Machine based model to determine a patient's condition and expected disease. This system uses ML-based approach for data mining and discoveries. This work not only gives an objective about malicious activities but also introduced an approach to merge the network flow metric with bio-metric data. Chakraborty [8] proposes a healthcare system design using blockchain technology. Blockchain technology is known to assure security, but the authors have not investigated the framework or tested it to present any benchmark result. Hady [9], proposed a new healthcare model for study in the area of Internet of Medical Things. The authors have introduced a wustl-ehms-2020 dataset using Enhanced Healthcare Monitoring System (EHMS) testbed. The proposed dataset was built on man-in-the-middle-attacks. The dataset combines the network flow metrics with patient biometrics. The authors in [10,11] uses K-Nearest Neighbour method as a base for cyber security research. Rao [10], used an Indexed Partial Distance search k-Nearest Neighbour(IKPDS) to test different types of attacks, which results in 99.6% accuracy. Shapoorifard [11] mainly focus on reducing the false alarm rate and show the accuracy of 85.2%. The above-discussed research uses an enhanced version of KDD Dataset but the dataset is being used since 1999 and the malicious attack signatures used are old. Further, Some recent studies have been done and compared with the current ones for providing a better approach in IoMT Security research and are mentioned in Table 1.

3. Proposed work

3.1. Motivation

The massive demand for IoMT networks to provide seamless effective medical facilities to a large scale population requires a robust secured model to ensure the privacy and safety of patients. However, designing security models for IoT networks is very challenging. An IoMT network typically involves sensor enabled medical devices, intermediate fog nodes, and cloud-based data servers which are connected via a wireless network. The indiscreet connection and dynamic nature of the network make it more vulnerable to attackers. Developing a security model for IoMT also requires the handling of huge data in a distributive manner. To address these challenges, a lightweight anomaly detection model is required for the IoMT network. This anomaly detection model can be used to protect the sensor data from cyber-attacks which can preserve the privacy and safety of the patients in real-time. As new IoMT Sensors are being introduced in wired or wireless mode, these models can be used to make these networks more robust from attacks using both network and biometric data. To represent the above challenge a new model has been introduced using both the metric data i.e., network and biometric.

Table 1
Related work summary.

Author	Dataset	Method	Results	limitations
Hady [9]	wustl-ehms- 2020	Used four different methods RF, KNN, SVM and ANN. ANN performed better on the dataset.	Training Accuracy = 88.75% Testing Accuracy = 90.42% AUC = 93.42	Requires hyperparameter tuning, prediction time is high, feature scaling was not performed.
Sarhan [12]	UNSW- BN 15 BoT-Iot ToN-IoT CSE-CIC- IDS2018	Used the standard dataset, generalize them into universal feature dataset, using n Probe Tool. Used extra tree classifier to bench mark the dataset. Derived NF-UQ-NIDS-V2 and Bench marked it with NF-UQ-NIDS-V1 with 43 Feature in Common Netflow.	Weighted Average = 96.93% F1 Score= 0.97 Prediction Time(μs) = 25.67	False Acceptance Rate (FAR) is higher. Takes more time in prediction. No other Algorithms have been considered for classification.
Sarhan [13]	UNSW- BN 15 BoT-Iot ToN-IoT CSE-CIC- IDS2018	Used the standard dataset, generalize them into universal feature dataset, using n Probe Tool. Used extra tree classifier to bench mark the dataset. Derived NF-UQ-NIDS-V1 and Bench marked it with 9 Feature in Common Netflow.	Weighted Average = 70.81% F1 Score = 0.79 Prediction Time(μs) = 14.67	Accuracy, FAR rate is higher. Takes more time in prediction. No other algorithms have been considered for classification.
Farhan [14]	CSE-CIC- IDS2018	Proposed a Deep Neural Network Model for classifying the attacks.	Accuracy = 90% Precision = 0.65 Recall = 0.59	No other Algorithms have been considered for classification, Dataset is not relevant for IoMT Purpose, Low precision and recall values.
Hussain [15]	CICIDS2017 CTU-13 IoT-23	Introduced the Universal Feature Set Concept. Common feature was selected using their frequency count. NB, KNN, RF, LR were used in Classification process.	Accuracy = 89.00% Precision = 81.64%	Dataset was not combined for universal classification process. Model was complex resulting in higher prediction time. Hyper-parameter tuning was not performed.
Kumar [16]	ToN-IoT	A Combination of ensemble learning and fog-cloud architecture-driven cyber-attack. The ensemble combines decision tree, naïve bayes , and random forest as first-level individual learners. Used IaaS on cloud side and Saas on fog side.	Accuracy = 96.35% Detection Rate = 99.98% FAR = 5.56	High FAR rate. Lack of Sensor Data. Not Lightweight.
Rathore [17]	Data obtained from Physionet	Used Supervised learning using Long Short-Term Memory model for preventing of stimulation strategies attacks on Deep brain simulators.	Multiple low Loss Value	Accuracy was not mentioned and the simulated attacks were not real.

3.2. Assumptions

The following assumptions have been made related to the system model:

1. The data contains both network and biometric information.
2. The cyber-attacks are initiated by the nodes placed outside a given network.
3. The model is trained and tested for Man-in-the-middle attacks (Data Spoofing and Alteration).

3.3. System design

The Framework proposed in this work Fig. 1 is a lightweight model designed for intrusion detection. The model uses a benchmarked Dataset [9] for the training model. The main goal is to achieve a model with less time in prediction and less FAR which is later explained in Sections 5.1, 5.2 and 5.4. The framework includes a data pre-processing phase which involves Analysis, Pre-processing and Feature Selection & Dimensional Reduction and an anomaly detection model. The anomaly detection model is based on Random Forest with Hyper-parameter tuning to achieve the best estimator which classifies for anomalies and normal samples.

3.4. Dataset description

The Dataset introduced by [9], is a combination of both network flow metric and bio-metric flow of the patients. The attack proposed in the dataset set is the Man in the Middle (MITM). The attack is described as when the attacker pretends to be a kind of router to get the packet first. Further two sub-types of MITM were considered by the researchers [9] as Spoofing attacks and Data Alteration. The dataset [9] has been built using a health monitoring sensor board that collects data from several healthcare sensors

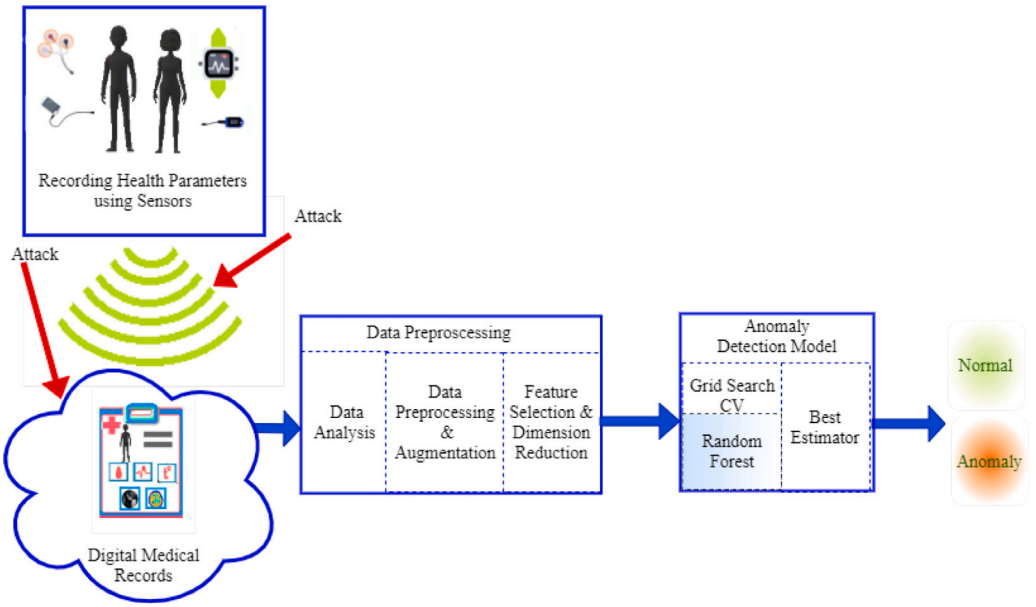

Fig. 1. Proposed framework.

Metric	Description	Metric Type	Metric Datatype	Metric	Description	Metric Type	Metric Datatype
Flgs	Flags Used in Network	Flow Metric	Object	Rate	Number of Packets per Second	Flow Metric	float64
SrcAddr	Source Address	Flow Metric	Object	SrcMac	Source Mac-address	Flow Metric	Object
DstAddr	Destination Address	Flow Metric	Object	DstMac	Destination Mac-address	Flow Metric	Object
Sport	Source Port	Flow Metric	int64	Packet_num	Packet Number	Flow Metric	int64
Dport	Desination Port	Flow Metric	int64	SIntPkt	Source Inter Packets	Flow Metric	float64
SrcBytes	Source Bytes	Flow Metric	float64	DIntPkt	Destination Inter Packets	Flow Metric	float64
DstBytes	Destination Bytes	Flow Metric	float64	SIntPktAct	Source Active Inter Packets	Flow Metric	float64
SrcLoad	Source Load	Flow Metric	float64	Dur	Duration	Flow Metric	float64
DstLoad	Destination Load	Flow Metric	float64	Trans	Aggregated Packet Counts	Flow Metric	int64
SrcGap	Source Missing Bytes	Flow Metric	float64	TotPkts	Total Packet Counts	Flow Metric	float64
DstGap	Destination Missing Bytes	Flow Metric	float64	TotBytes	Total Packet Bytes	Flow Metric	float64
DIntPktAct	Destination Active Inter Packets	Flow Metric	float64	Loss	Retransmitted or Dropped Packet	Flow Metric	int64
SrcJitter	Source Jitter	Flow Metric	float64	pLoss	Percentage of Retransmitted or Dropped Packet	Flow Metric	float64
DstJitter	Destination Jitter	Flow Metric	float64				
sMaxPktSz	Source Maximum Transmitted Packets	Flow Metric	int64	pSrcLoss	Percentage of Source Retransmitted or Dropped Packet	Flow Metric	float64
dMaxPktSz	Destination Maximum Transmitted Packets	Flow Metric	int64	pDstLoss	Percentage of Destination Retransmitted or Dropped	Flow Metric	float64
sMinPktSz	Source Minimum Transmitted Packets	Flow Metric	int64	Temp	Packet Temperature	Biometric	float64
dMinPktSz	Destination Minimum Transmitted Packets	Flow Metric	int64				
SpO2	Peripheral Oxygen Saturation	Biometric	int64	Load	Load	Flow Metric	float64

Fig. 2. Dataset description.

which are placed on a patient's body. The board was attached to a window-based system using the USB port. The software was developed using the C++ language. The system acts as a gateway where the transferring of data was done using TCP/IP protocol through Wi-Fi over the server. The testbed used Argus to collect all the network traffic flows and patient data between the gateway and server. It is open-source software that is used for monitoring the network flow traffic in real-time. The features used in dataset are mentioned in Fig. 2. Table 2 represents the number of samples in the dataset.

The number of samples in dataset are described in Table 2.

3.5. Data pre-processing

The Dataset was pre-processed before the Grid search process. The Pre-process Phase contains the Data Analysis, Data Pre-Processing and Augmentation, Feature Selection and Dimensional Reduction.

Table 2
No. of samples in the dataset [9].

Number of samples	Value
Normal samples	14,272
Attack samples	2,046
Total number of samples	16,318

3.5.1. Data analysis

The Data analysis part contains the study of features and their relationships. There are two parts of Data analysis which are mentioned below.

Count Analysis: It gives an idea about the data in feature such as total count, uniqueness, top value in feature and its frequency. Other than that, Count Analysis also provides the count for each value in the feature set which is to be explored. To build the relationship between the different features like number of attack samples which contains a particular flag number of the protocol in the network.

Variable analysis: It tells about the nature of the feature. The nature of the feature is related to datatype as object datatype will be equivalent to count analysis, but on the other hand in the int64 and float64 will give us some statistical observation about the feature. These observations include mean, standard deviation, min. value, max. value and three quartile values (1st, 2nd, and 3rd). If the same process is used continuously, these values play a significant role in building the model.

3.5.2. Data pre-processing and augmentation

This phase contains two steps namely Data Pre-Processing and Data Augmentation. In this phase, it is observed that there are some changes needed to be done in the dataset such as converting the related feature to relevant datatype, removal of irrelevant features which can create noise in classification and label encoding of the relevant feature. To scale the dataset the statistical feature was used such as quartile 1 and 3 in Robust scaler.

The main goal for robust scaling is to scale features using statistics that are robust to outliers. The scaler removes the median and scales the data according to the quartile range. Default range value is provided using 1st quartile and 3rd quartile.

$$X_{FeatureScaled} = \frac{X_{FeatureCurrentValue} - Q_1(X_{Feature})}{Q_3(X_{Feature})) - Q_1(X_{Feature})} \tag{1}$$

The first quartile (Q_1) is defined as the middle number between the smallest number (minimum) and the median of the data set. It is also known as the lower or 25th empirical quartile, as 25% of the data is below this point. The third quartile ($Q3$) is the middle value between the median and the highest value (maximum) of the data set. It is known as the upper or 75th empirical quartile, as 75% of the data lies below this point.

Other than that, two other scalers were used namely MinMax Scaler and Standard Scaler, they are explained as below.

Min–Max Scaler is also known as normalization and considered as the simplest scaling. The value range for this scaling is between 0 to 1. The feature value is obtained by subtracting the min value and divided by the max value minus min value.

$$X_{FeatureScaled} = \frac{X_{FeatureCurrentValue} - min(X_{Feature})}{max(X_{Feature}) - min(X_{Feature})} \tag{2}$$

Standard Scalar scaling technique uses mean and standard deviation to scale the feature. Standard scalar is not dependent upon any particular range. The main goal of this feature is to keep the mean equal to 0 and the standard deviation to 1 because it distributes all the feature values in the similar distribution.

$$X_{FeatureScaled} = \frac{X_{FeatureCurrentValue} - mean(X_{Feature})}{stdev(X_{Feature})} \tag{3}$$

In above Eqs. (1)–(3) $X_{FeatureScaled}$ represents a scaled feature value after scaling has been processed, $X_{FeatureCurrentValue}$ represents current feature value without any modifications, $X_{Feature}$ represents whole feature columns from the dataset, $Q_1()\&Q_3()$ represents a quartile function ranging from 0 to 100 but in the proposed work defaults were used i.e, 25 and 75, min() and max() represent minimum and maximum functions to retrieve the respective values from each feature of the dataset, mean() represents mean for each feature and stdev() represents standard deviation for the feature.

Data augmentation in data analysis is a technique used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model [18]. Oversampling in data analysis is a technique used to adjust the class distribution of a data set (i.e., the ratio between the different classes/categories represented). These terms are used both in statistical sampling, survey design methodology and in machine learning. There are a number of methods available to oversample a dataset used in a typical classification problem. The most common technique is known as SMOTE: Synthetic Minority Over-sampling Technique [19]. This technique was used to resample the dataset in the work. The below mentioned algorithm 1 in Section 3.8 was used to achieve the resampling of the dataset [9].

3.5.3. Feature selection and dimensional reduction

Feature selection is a technique of obtaining a subset of the features from the original feature set without any transformation on the dataset. On the other hand, Dimensional reduction is referred to same as Feature Selection but some transformation is applied to transform the higher dimension to the lower dimension in the dataset. A study of mean, standard deviation, 1st quartile, 3rd quartile and count analysis were considered to select various features from the original feature set of the dataset.

3.6. Anomaly detection model

In this phase, the model development is presented using Random Forest [20,21], GridSearchCV [22], and Best Estimator [21,22]. Here, other algorithms such as Logistic Regression [21,23], Decision Tree [24] and Extremely Randomized Tree [25] have also been considered. The various parts of this phase are described as below.

3.6.1. Grid search

In this section, a popular hyperparameter optimization technique called grid search is used, which helps in improvising the performance of classifier by finding optimal combination of hyperparameter values. Grid Search is a brute-force exhaustive search paradigm where a list of various values are defined for different hyperparameters to be initialized [22,24]. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set. After the training is completed on training data, the best estimator can be used which is result of best parameter from exhaustive search.

3.6.2. Logistic regression

It is a parametric classification algorithm which is very simple in use and implementation. It efficiently works on linear separable classes. It classifies both binary and multi-class problems using OvR(One-vs-Rest) Method [24]. Logistic Regression uses Sigmoid function for classification.

3.6.3. Decision tree

It is a non-parametric classification and regression algorithm and even multioutput tasks. These are very powerful algorithms so that this makes them capable of fitting the complex dataset. These trees are fundamentals for the random forest. To split the nodes, the use of most informative features is done like an objective function is defined as Informative Gain. Generally, binary decision tree is implemented to reduce the combinatorial search space [24]. There are three impurity measures or splitting criterion which are considered Gini Entropy (I_G), entropy (I_H) and classification error (I_E).

3.6.4. Random forest

A Random Forest [20] is known as ensemble of various Decision Trees, they are trained using bagging method. The intuition idea behind developing random forest was to get average multiple (deep) decision trees in which the individual suffer from high variance, to build more robust model having a better generalization performance and it is less susceptible to over-fitting.

3.6.5. Extra tree

An Extremely randomized Trees ensemble [25] is technique which trades more bias for a lower variance. It makes Extra-Trees much faster to train than regular forests, because finding the best possible threshold for each feature at every node is one of the most time-consuming tasks of growing a tree.

3.7. Proposed framework

The work aimed to design a lightweight intrusion detection model that deploys the feature scaling technique with the random forest model. The proposed model shown in Fig. 3 identifies an intrusion by classifying the input data as ‘intrusion’ and ‘normal’ data. The model is considered lightweight, since it uses feature selection and dimension reduction technique which reduces the number of features to be processed and also the overall detection time. To select the proper category of the input sample, hyperparameter tuning is used to attain a best estimator. The model can function with low resources and can be appropriately used both at the node and the network level of an IoMT system.

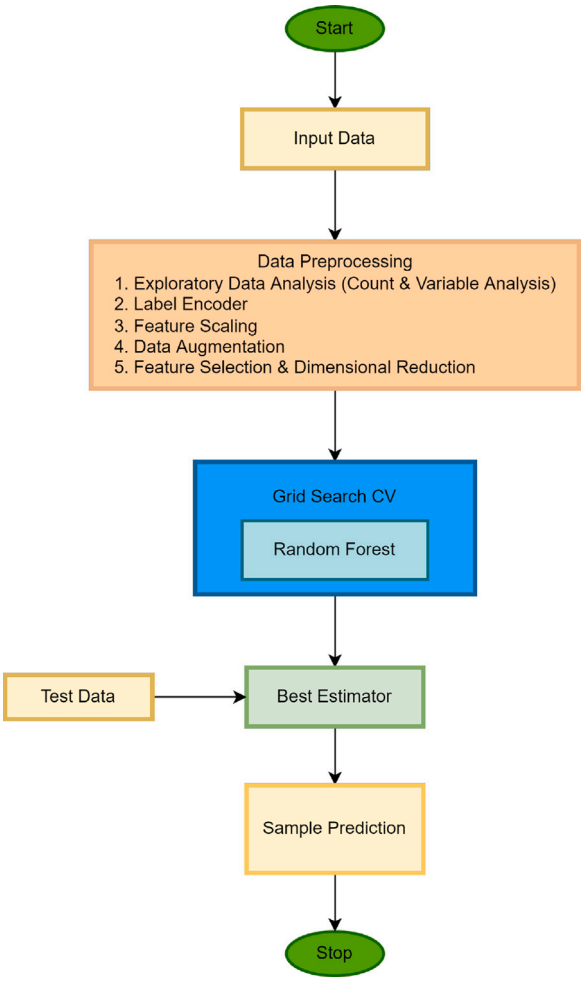

Fig. 3. Proposed model.

3.8. Algorithms

Algorithm 1 - Performs data augmentation in the dataset by making the number of samples equal to majority and minority class. The notations used in the algorithms are specified in Table 3.

Algorithm 1 Data Augmentation()

1: **Input: Dataset** DS
2: **Output:** DS'
3: **Begin**

i Initialization: $DS' = Data\_Augmentation$(DS)
ii Data_Augmentation(DS)

(a) DS_{temp} = call SMOTE(DS)
(b) **return** (DS_{temp})

4: **End**

Algorithm 2 - It Computes the ratio between majority and minority classes in the dataset. This ratio is passed in classification algorithms as a parameter which will be explained later.

Algorithm 2 Compute Class Weight()

1: **Input: Training Classes** TC
2: **Output: ClassWeightMap**
3: **Begin**
4: **Initialization:** $Classes\_Array$ = Number of Classes in the Array available.
5: **Call** $Compute\_Class\_Weight()$ function with parameter as **Balanced** ratio, $Classes\_Array$, $Training\_Classes$.
6: **Map** the weight ratio array with classes.
7: **return** (ClassWeightMap)

Algorithm 3- It perform a feature scaling using Eq. (1). Robust scaler function has two attributes scale_ and center_ which specifically returns an array of median values and quartile ranges for each feature.

Algorithm 3 Robust Scaler()

1: **Input: Training Dataset** TD
2: **Output:** TD'**,** $center\_value\_array$**,** $scale\_array_1$
3: **Begin**

i Initialization: TD', $center\_value\_array$, $scale\_array_1$ = Scaling(TD)
ii Scaling(TD)

(a) TD_{new} = call RobustScaler(TD) using Eq. (1).
(b) $center\_value\_array$ = call RobustScaler(TD).center_attribute
(c) $scale\_array_1$ = call RobustScaler(TD).scale_attribute
(d) ***return*** (TD_{new} , $center\_value\_array$, $scale\_array_1$)

4: **End**

Algorithm 4- It perform a feature scaling using Eq. (2). MinMax scaler function has two attributes data_min_ and data_max_ which specifically returns an array of minimum and maximum values for each feature.

Algorithm 4 MinMax Scaler()

1: **Input: Training Dataset** TD
2: **Output:** TD'**,** min_value_array**,** max_value_array
3: **Begin**

i Initialization: TD', min_value_array, max_value_array = Scaling(TD)
ii Scaling(TD)

(a) TD_{new} = call MinMaxScaler(TD) using Eq. (2).
(b) min_value_array = call MinMaxScaler(TD).data_min_attribute
(c) max_value_array = call MinMaxScaler(TD).data_max_attribute
(d) ***return*** (TD_{new} , min_value_array, max_value_array)

4: **End**

Algorithm 5- It perform a feature scaling using Eq. (3). Standard scaler function has two attributes mean_ and scale_ which specifically returns an array of mean and standard deviation values for each feature.

Algorithm 5 Standard Scaler()

1: **Input: Training Dataset** TD
2: **Output:** TD'**,** $mean\_value\_array$**,** $scale\_array_2$
3: **Begin**

i Initialization: TD', $mean\_value\_array$, $scale\_array_2$ = Scaling(TD)
ii Scaling(TD)

(a) TD_{new} = call StandardScaler(TD) using Eq. (3).
(b) $mean\_value\_array$ = call StandardScaler(TD).mean_attribute
(c) $scale\_array_2$ = call StandardScaler(TD).scale_attribute
(d) ***return*** (TD_{new} , $mean\_value\_array$, $scale\_array_2$)

4: **End**

Algorithm 6- This Algorithm uses Logistic regression. The *Param_grid* contains C parameter which is inverse of regularization strength. Scoring method and CV for algorithms 6, 7, 8 and 9 were accuracy and 10 folds respectively.

Algorithm 6 GridSearchCV + LogisticRegression()

1: **Input: Training Features and Classes**
2: **Output: Hyper-tuned Logistic Regression Classifier**
3: **Begin**
4: **Training**

i Initialize the *Param_grid* for Logistic regression.
ii Call the GridSearchCV estimator (with class weight if available) and input the *Param_grid*.
iii Fit the grid estimator with scaled training features and labelled classes.
iv Retrieve the best parameters for the classifier.
v Initialize the classifier using GridSearchCV Best Estimator as the input.
vi Fit the classifier with scaled training features and labelled classes.
vii Obtain the classifier.

5: **Testing**

i Predict class of each testing feature until all the features are processed and store prediction time for each prediction.
ii Evaluate the classifier using Evaluation Metrics such as Accuracy, Detection Rate, False Acceptance Rate, AUC Score, F1-Score, Precision and Prediction Time.

6: **End**

Algorithm 7, 8 and 9- These Algorithm uses Decision Tree, Random Forest and Extra Tree. The below parameters were used in random forest for getting better results and The *Param_grid* contains following ranges also:

1. Criterion - It is function to measure the quality of split i.e, Gini and Entropy.
2. min_samples_split - It is minimum number of samples required to split an internal node. Minimum Sample Split ranges from 10 to 100 at an interval of 10.
3. n_estimators(only for algorithms 8 and 9) - The number of trees to build the forest that ranges from 10 to 30 at an interval of 2.

Algorithm 7 GridSearchCV + DecisionTree()

1: **Input: Training Features and Classes**
2: **Output: Hyper-tuned Decision Tree Classifier**
3: **Begin**
4: **Training**

i Initialize the *Param_grid* for Decision Tree.
ii Call the GridSearchCV estimator (with class weight if available) and input the *Param_grid*.
iii Fit the grid estimator with scaled training features and labelled classes.
iv Retrieve the best parameters for the classifier.
v Initialize the classifier using GridSearchCV Best Estimator as the input.
vi Fit the classifier with scaled training features and labelled classes.
vii Obtain the classifier.

5: **Testing**

i Predict class of each testing feature until all the features are processed and store prediction time for each prediction.
ii Evaluate the classifier using Evaluation Metrics Accuracy, Detection Rate, False Acceptance Rate, AUC Score, F1-Score, Precision and Prediction Time.

6: **End**

Algorithm 8 GridSearchCV + RandomForest()

1: **Input: Training Features and Classes**
2: **Output: Hyper-tuned Random Forest Classifier**
3: **Begin**
4: **Training**

i Initialize the *Param_grid* for Random Forest.
ii Call the GridSearchCV estimator (with class weight if available) and input the *Param_grid*.
iii Fit the grid estimator with scaled training features and labelled classes.
iv Retrieve the best parameters for the classifier.
v Initialize the classifier using GridSearchCV Best Estimator as the input.
vi Fit the classifier with scaled training features and labelled classes.
vii Obtain the classifier.

5: **Testing**

i Predict class of each testing feature until all the features are processed and store prediction time for each prediction.
ii Evaluate the classifier using Evaluation Metrics Accuracy, Detection Rate, False Acceptance Rate, AUC Score, F1-Score, Precision and Prediction Time.

6: **End**

Algorithm 9 GridSearchCV + ExtraTreeClassifier()

1: **Input: Training Features and Classes**
2: **Output: Hyper-tuned Extra Tree Classifier**
3: **Begin**
4: **Training**

i Initialize the *Param_grid* for Extra Tree.
ii Call the GridSearchCV estimator (with class weight if available) and input the *Param_grid*.
iii Fit the grid estimator with scaled training features and labelled classes.
iv Retrieve the best parameters for the classifier.
v Initialize the classifier using GridSearchCV Best Estimator as the input.
vi Fit the classifier with scaled training features and labelled classes.
vii Obtain the classifier.

5: **Testing**

i Predict class of each testing feature until all the features are processed and store prediction time for each prediction.
ii Evaluate the classifier using Evaluation Metrics Accuracy, Detection Rate, False Acceptance Rate, AUC Score, F1-Score, Precision and Prediction Time.

6: **End**

4. Experimentation and results

To evaluate the proposed model, rigorous experimentation has been performed using the dataset [9].

4.1. Data augmentation and class weight ratio

After the Data Pre-processing, Class weight ratio algorithm 2 has been used on original dataset to adjust the class importance in classification. In addition, algorithm 1 has also been used to balance the dataset. The description of the number of samples and their class weight ratio in presented in Tables 4 and 5. The class weight ratio is a parameter in the classifier which will make an assumption of a balanced classification by assigning a ratio to the minority class in the process. To obtain these values, algorithms 1 and 2 are applied on the dataset.

Scaled dataset is for the classification algorithms under hyper-parameter tuning. The scaled feature dataset are Original Dataset and Data Augmented Dataset. Before that, only relevant features from the dataset i.e., 34 out of 44 are considered using variable analysis. All the bio-metric features in the classification process have been used.

4.2. Classification algorithms and hyper-parameter tuning

The algorithms 6, 7, 8 and 9 in Section 3.8 are executed with the original dataset, augmented dataset and variation of original dataset with class weight ratio. However, the class weight ratio is not calculated for augmented dataset as augmented dataset is already balanced.

Table 3
Notations used for the essential parameters.

Symbol	Details.
DS	Original Dataset.
DS'	Augmented Dataset.
DS_{temp}	Store the augmented dataset.
TD	Training Dataset.
TD'	Scaled Training Dataset.
TD_{new}	Store the scaled dataset.
$center\_value\_array$	Array to Store/Return quartile ranges for each feature in training dataset.
$scale\_array_1$	Array to Store/Return median value for each feature in training dataset.
min_value_array	Array to Store/Return minimum value for each feature in training dataset.
max_value_array	Array to Store/Return maximum value for each feature in training dataset.
$mean\_value\_array$	Array to Store/Return mean value for each feature in training dataset.
$scale\_array_2$	Array to Store/Return standard deviation for each feature in training dataset.
Scoring	Strategy to evaluate the performance of the cross-validated model on the test set.
$param\_grid$	Parameter list for the classifiers.
CV	Cross-Validation splitting strategy

Table 4
Number of samples with and without data augmentation.

Number of samples	Without augmentation	With augmentation
Training samples	13,054	22,383
Testing samples	3,264	5,709
Total samples	16,138	28,544

Table 5
Class weight ratio.

Types of samples	Weight ratio
Normal samples	0.57134104
Attack samples	4.00429448

The result of each test is given in terms of accuracy (% of correctly classified instances). The evaluation metrics is been calculated using following measurements.

True Positives (TP)- Correctly classified as Attack Sample.

True Negatives (TN)- Correctly classified as Normal Sample.

False Positive (FP)- Classified a Normal data as Attack Sample.

False Negatives (FN)- Classified as Normal Sample but actually it is an Attack Sample.

Accuracy is calculated from True Positives(TP) and True Negatives(TN) for binary classification using Eq. (4):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{4}$$

In order to measure performance, evaluation metrics like Precision, Recall/Detection Rate, FAR, F1-score and AUC are calculated in terms of positives and negatives :

Precision – It represents the number of correct positive prediction in total number of positive predictions and defined by Eq. (5):

$$Precision = \frac{TP}{TP + FP} \times 100 \tag{5}$$

Recall/Detection Rate – It represents the number of correct Positive prediction in total number of positive observations and defined by Eq. (6):

$$Recall/DR = \frac{TP}{TP + FN} \times 100 \tag{6}$$

False Acceptance Rate/False Positive Rate (FAR/FPR) – It represents number of False Positive Prediction in total number of False attempts and defined by Eq. (7):

$$FAR/FPR = \frac{FP}{TN + FP} \tag{7}$$

False Negative Rate (FNR) – It represents number of False Negative Prediction in total number of True Attempts and defined by Eq. (8):

$$FNR = \frac{FN}{TN + FN} \tag{8}$$

Table 6
Comparison with existing model.

Evaluation metrics	Hady [9]	Proposed model
10-Fold Training Accuracy	88.75%	92.85%
10-Fold Testing Accuracy	90.04%	94.23%
AUC Scores	93.42	90.68

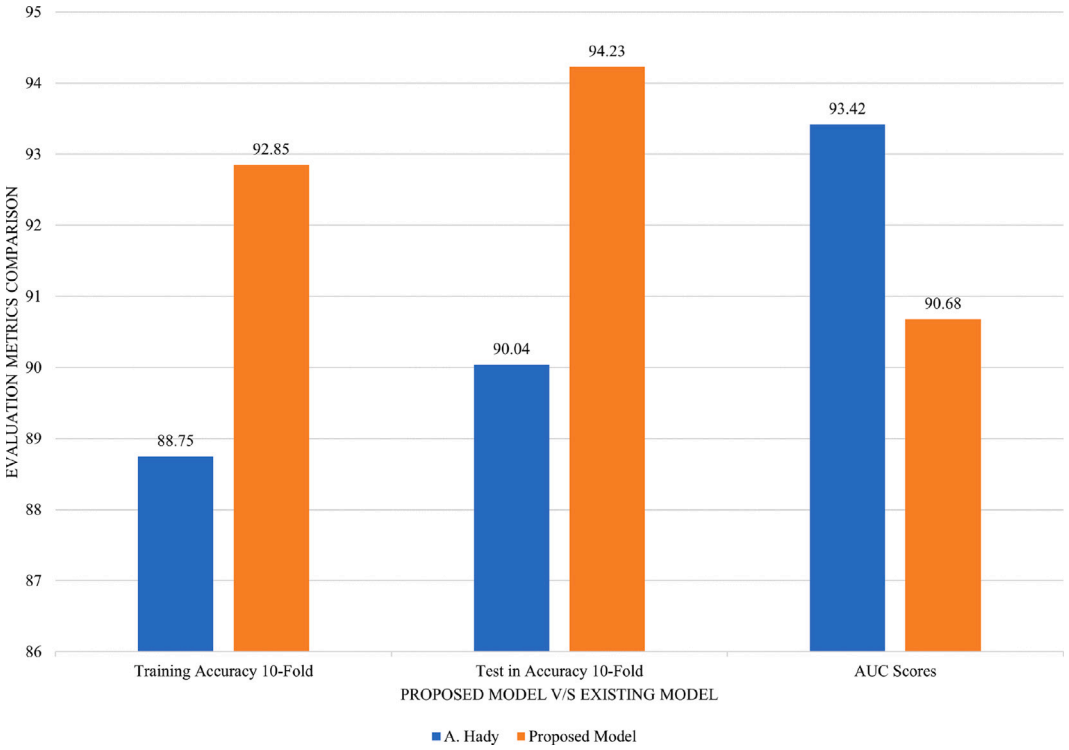

Fig. 4. Accuracy of the proposed model.

F1-Score - It is the harmonic mean of the Precision and the Recall. This evaluation metric is very useful to study imbalanced learning problems. To overcome this issue, algorithm 2 is used which is defined by Eq. (9):

$$F1 - Score = 2 \times \frac{PrecisionX\,Recall}{Precison + Recall} \times 100 \tag{9}$$

Prediction Time (μs) - The time is shown as prediction per sample.

ROC-AUC (Area under the Receiver Operator Characteristic Curve) - Receiver Operating characteristic(ROC) graph is a useful tool to choose models based on its performance for classification with respect to FPR and TPR [24]. The computation is recorded using a threshold decision which is shifted by the classifier. A classifier is said to be perfect when the top left corner of the graph lies with TPR of 1 and FPR of 0 [24]. The AUC value measures the correlation between the actual attacks that occurred during an event w.r.t number of flows that are classified as attacks but are not.

The results are obtained for the proposed model and compared with the other existing models along side with many variations of the dataset with different classifiers.

Table 6 compares 10-Fold Training, Testing Accuracy and AUC scores of Artificial Neural Network proposed by [9] and our proposed approach with combination of above-mentioned algorithms 1, 3, and 8.

The comparison of the existing model [9] and the proposed model is presented in Fig. 4. In the proposed work, four well recognized algorithms are considered Logistic Regression, Decision Tree, Random Forest and Extra Tree Classifier. The techniques are combined with various feature scaling techniques such as Robust Scaler, Standard Scaler and MinMax Scaler. Furthermore, to solve imbalanced learning problem class weight ratio parameter and data Augmentation are deployed. The combination of the Robust Scalar, Data Augmentation and Random Forest has significantly outnumbered the current existing work in terms of the training and testing accuracy for Intrusion Detection in IoMT with the values of 92.85% and 94.23% respectively.

Tables 7–15 and their subsequent Figs. 5–13 describe the evaluation metrics namely — Training Accuracy (%), Testing Accuracy (%), Detection Rate (%), AUC (%), False Acceptance Rate (only in tables), F1-score (%), Precision (%) and Time (μs).

Table 7
Comparison with other algorithms.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	FAR	F1-Score (%)	Precision (%)	Time (μs)
RF–Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	0.06	93.8	93.45	3.5
LR–Data augmentation	75.13	74.89	55.98	75.14	0.05	69.30	90.94	57.10
DT–Data augmentation	90.55	92.56	89.55	90.81	0.07	90.7	92.06	82
ET–Data augmentation	88.86	89.01	93.25	89.09	0.06	88.54	84.45	3.6

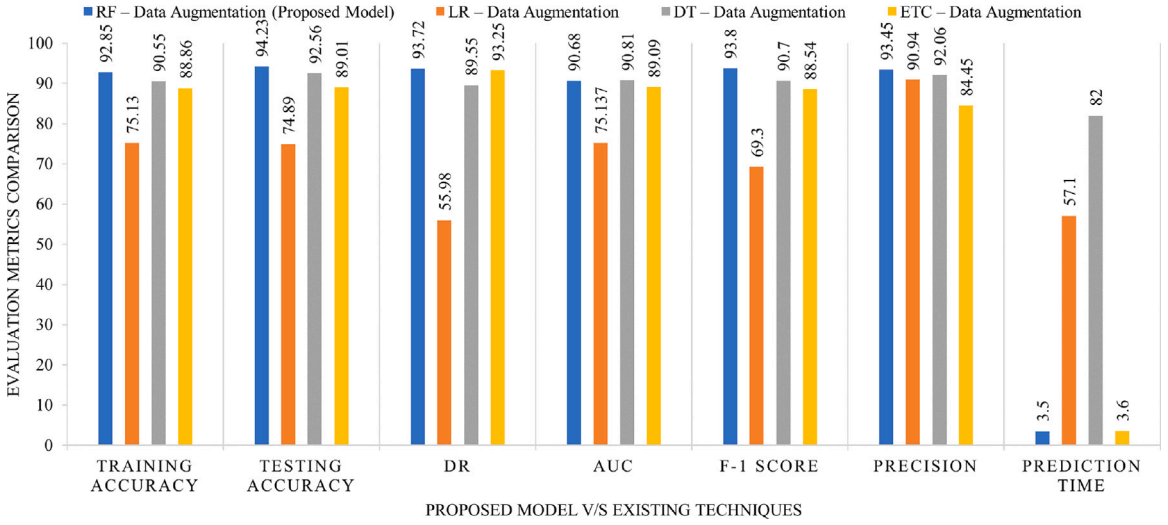

Fig. 5. Performance metrics of the proposed model using Robust Scalar.

Table 8
Comparison with other algorithms having class weight and Robust Scalar.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	FAR	F1-Score (%)	Precision (%)	Time (μs)
RF–Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	0.06	93.8	93.45	3.5
LR–Class weight	89.88	89.00	55.52	74.70	0.06	56.27	57.03	42
DT–Class weight	89.71	91.81	89.90	91.00	0.07	73.69	62.43	41
RF–Class weight	92.58	91.85	67.78	81.57	0.04	67.95	68.11	0.39
ET–Class weight	92.45	92.95	59.61	78.72	0.02	68.31	80.00	4.7

The evaluation metrics are calculated using Eq. (4)–(9). From the results, it is evident that the proposed model based on the Data Augmentation and Random Forest, performed better than the other techniques, since random forest along with feature scaling, performs better than the other techniques specifically for large dataset with categorical data.

The next comparison is done with the other algorithms 6, 7 and 9 using Data Augmentation Algorithm 1 and Robust Scaler Algorithm 3 and the results are shown in Table 7 and Fig. 5.

In the next set of experimentation, class weight algorithm 2 and Robust Scaler Algorithm 3 are used to compare the proposed model with the other techniques. The results are shown in Table 8 and Fig. 6

In the next analysis only Robust Scaler is used for the comparison and shown in Table 9 and Fig. 7.

In the next comparison, Standard Scaler Algorithm 5 is only considered and corresponding results are in Table 10 and Fig. 8.

Table 11 and Fig. 9 depicts the next comparison of the other techniques with the proposed model using class weight ratio and Standard Scaler Algorithms.

In the next experimentation, a combination of Data Augmentation and Standard Scaler Algorithms are used. The results are shown in Table 12 and Fig. 10.

The results in Table 13 and Fig. 11 shows the performance of the proposed model using MinMax Scaler. Then the MinMax Scaler is combined with the Class Weight and the results are presented in Table 14 and Fig. 12.

In the last set of experimentation MinMax is combined with the Data Augmentation technique and the comparative results are shown in Table 15 and Fig. 13.

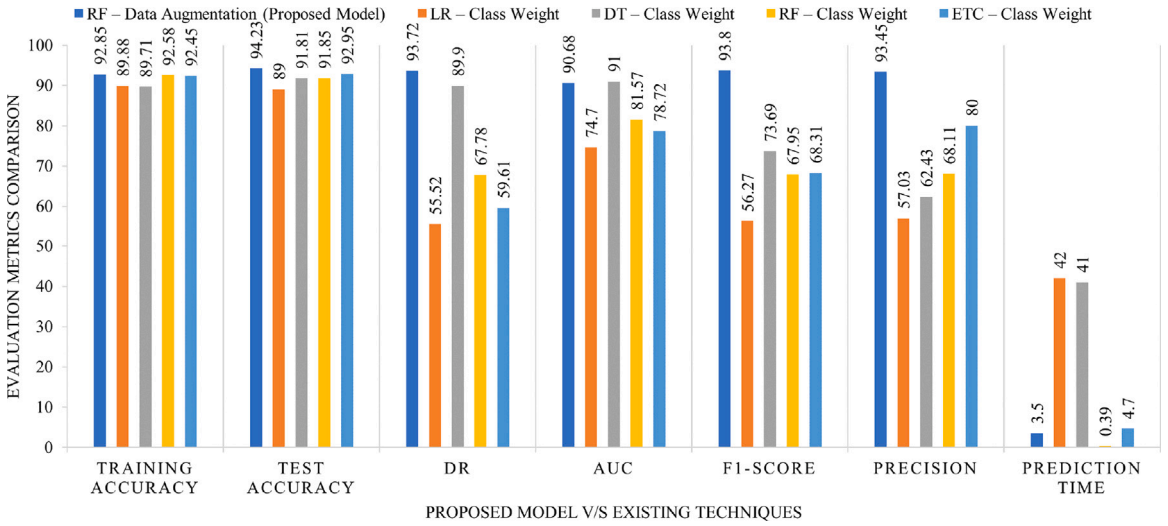

Fig. 6. Performance metrics of the proposed model using Robust Scalar with class weight ratio.

Table 9
Comparison with other algorithms using Robust Scaler.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	FAR	F1-Score (%)	Precision (%)	Time (μs)
RF–Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	0.06	93.8	93.45	3.5
LR	93.4	93.32	50.48	75.02	0.00	65.82	94.59	27
DT	94.87	93.90	63.94	81.11	0.01	72.78	84.45	48
RF	94.10	93.71	51.20	75.56	0.00	67.93	100	0.13
ET	93.97	93.71	51.20	75.56	0.00	67.51	99.06	0.62

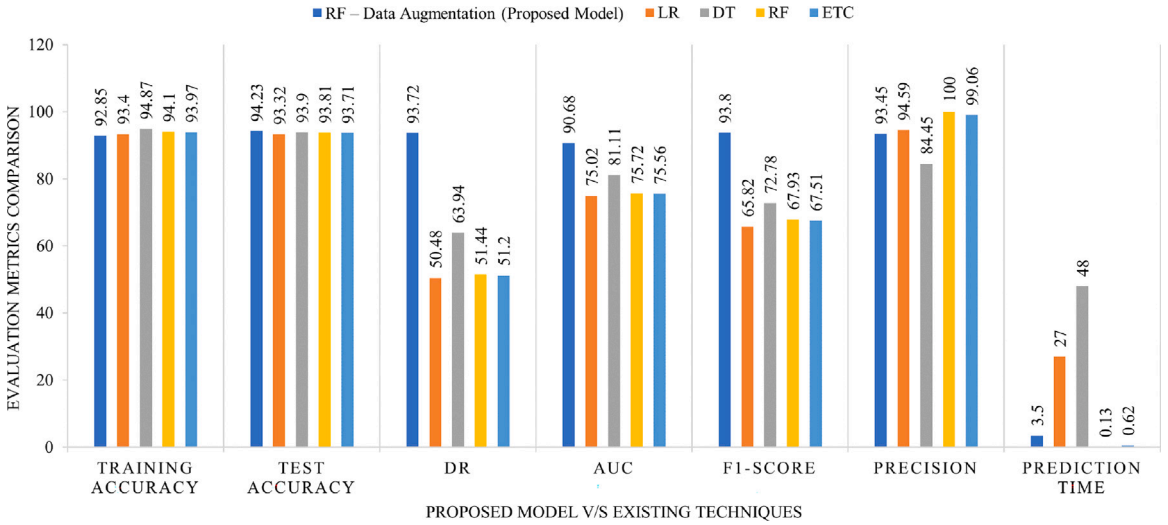

Fig. 7. Performance metrics of the proposed model using Robust Scalar with data augmentation.

5. Discussions

5.1. Confusion matrix for proposed model

Table 16 represents the confusion matrix for the proposed model. The False Positive Rate is 6.07% and the false Negative rate is 5.84% after classification on the augmented dataset.

Table 10
Comparison of proposed model with other algorithms using Standard Scaler only.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	FAR	F1-Score (%)	Precision (%)	Time (µs)
RF–Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	0.06	93.8	93.45	3.5
LR	93.06	92.95	44.95	72.45	0.00	61.92	99.46	20.3
DT	95.02	93.90	63.90	81.11	0.01	72.77	84.45	32.19
RF	94.09	93.81	51.44	75.72	0.00	67.93	100	2.9
ET	93.97	93.75	50.96	75.48	0.00	67.51	100	1.77

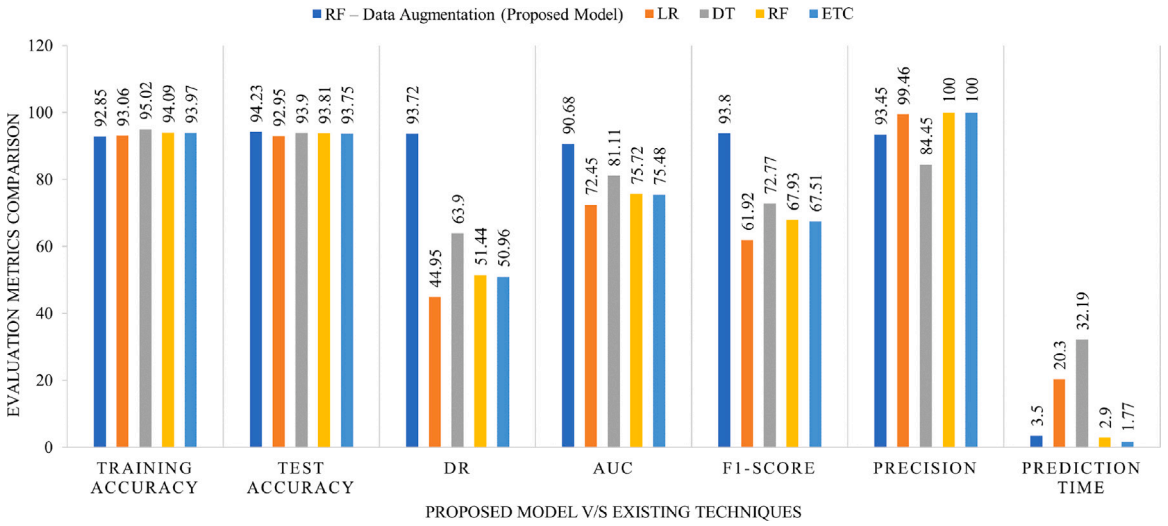

Fig. 8. Performance metrics of the proposed model using standard scalar.

Table 11
Comparison of proposed model with other algorithms using Standard Scaler and class weight ratio.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	FAR	F1-Score (%)	Precision (%)	Time (µs)
RF–Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	0.06	93.8	93.45	3.5
LR-Class weight	89.49	89.062	57.211	75.42	0.06	57.14	57.07	33.84
DT-Class weight	89.80	91.85	90.14	91.12	0.07	73.81	62.50	2.73
RF-Class weight	92.85	91.81	67.30	81.35	0.04	67.30	68.12	26.41
ET-Class weight	92.50	93.07	58.41	78.27	0.01	68.25	82.09	5.02

Table 12
Comparison of proposed model with other algorithms using Standard Scaler and data augmentation.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	FAR	F1-Score (%)	Precision (%)	Time (µs)
RF–Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	0.06	93.8	93.45	3.5
LR-Data augmentation	94.94	73.78	53.80	74.02	0.05	67.50	90.56	70
DT-Data augmentation	90.5	90.8	89.58	90.87	0.07	90.8	92.1	86.5
RF-Data augmentation	90.22	90.78	87.88	90.82	0.06	90.61	93.51	4.9
ET-Data augmentation	88.81	90.2	85.00	90.32	0.04	89.83	95.26	7.5

5.2. Role of data augmentation in classification

Initially, the dataset used to develop this model contained fewer attack samples. After data augmentation, better results were obtained in terms of Testing Accuracy, Detection Rate, AUC Score, F1-Score, and Precision. To achieve better results robust scaling algorithm 3 was used. The result are shown in Table 17.

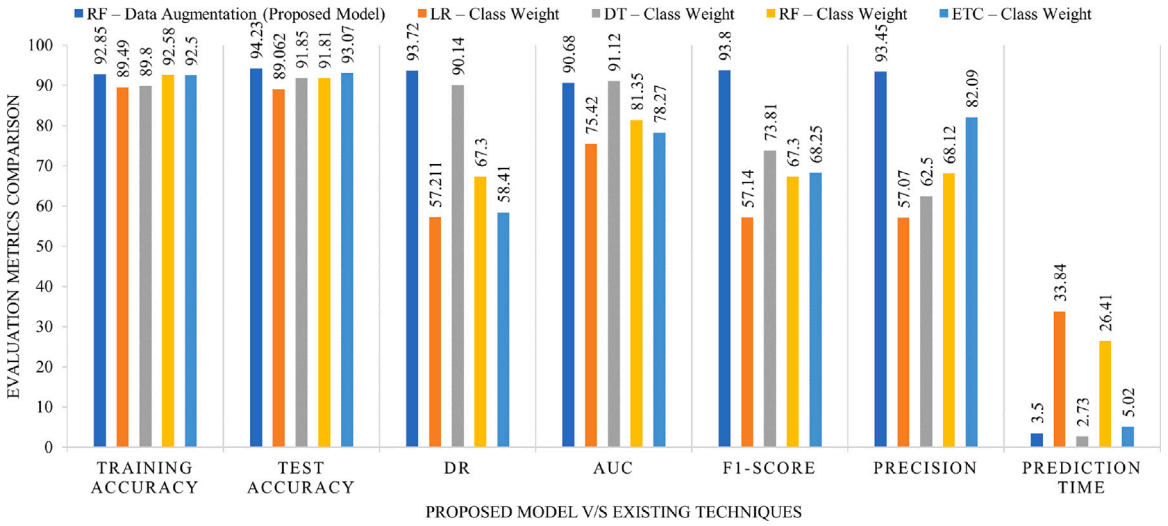

Fig. 9. Performance metrics of the proposed model using standard scalar with class weight ratio.

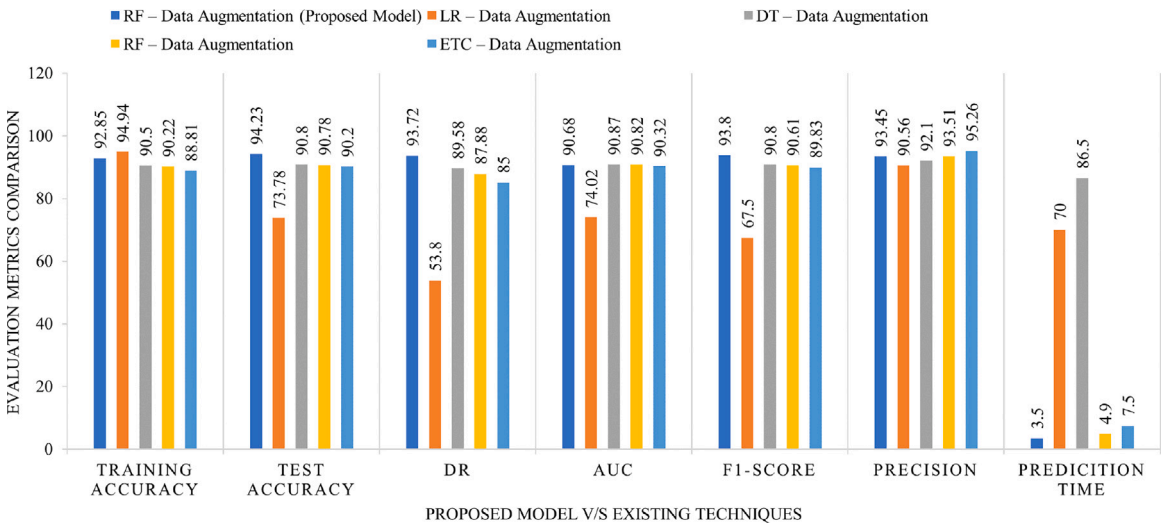

Fig. 10. Performance metrics of the proposed model using standard scalar with data augmentation.

Table 13
Comparison of proposed model with other algorithms using MinMax Scaler only.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	FAR	F1-Score (%)	Precision (%)	Time (μs)
RF–Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	0.06	93.8	93.45	3.5
LR	93.31	92.98	44.95	72.47	0.00	62.03	100	25.8
DT	95.11	94.33	66.10	82.29	0.01	74.82	86.62	67.03
RF	94.11	93.81	51.44	75.72	0.00	67.94	100	2.87
ET	93.97	93.78	51.20	75.60	0.00	67.72	100	2.62

5.3. Prediction time

The Prediction time is reduced by data cleaning processes such as scaling, removal of irrelevant features, label encoding, and feature selection based on the mean and standard deviation. It is also reduced by choosing an appropriate algorithm, to show the further investigation the results are compiled in Table 18. The Prediction time is also relatable to the system specification on which

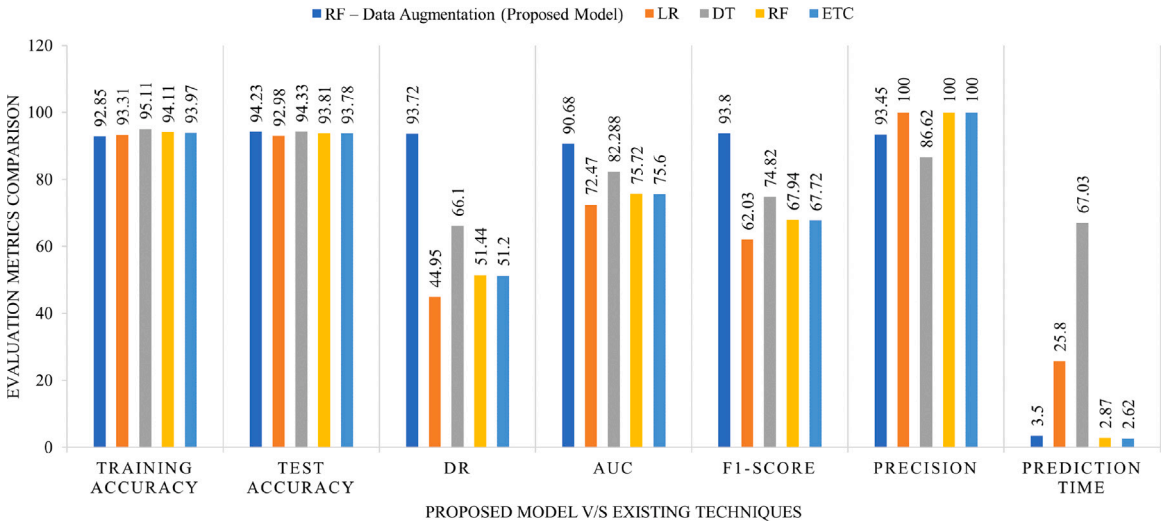

Fig. 11. Performance metrics of the Proposed Model using MinMax scalar.

Table 14
Comparison of proposed model with other algorithms using MinMax Scaler and class weight.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	FAR	F1-Score (%)	Precision (%)	Time (μs)
RF–Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	0.06	93.8	93.45	3.5
LR-Class weight	90.98	90.53	49.27	72.339	0.034	57.00	68.65	33.67
DT-Class weight	90.33	92.21	92.66	90.92	0.03	95.40	98.32	50
RF-Class weight	92.66	92.31	69.71	82.66	0.04	69.79	69.87	2.7
ET-Class weight	92.50	93.04	59.13	78.56	0.02	68.42	81.18	2.25

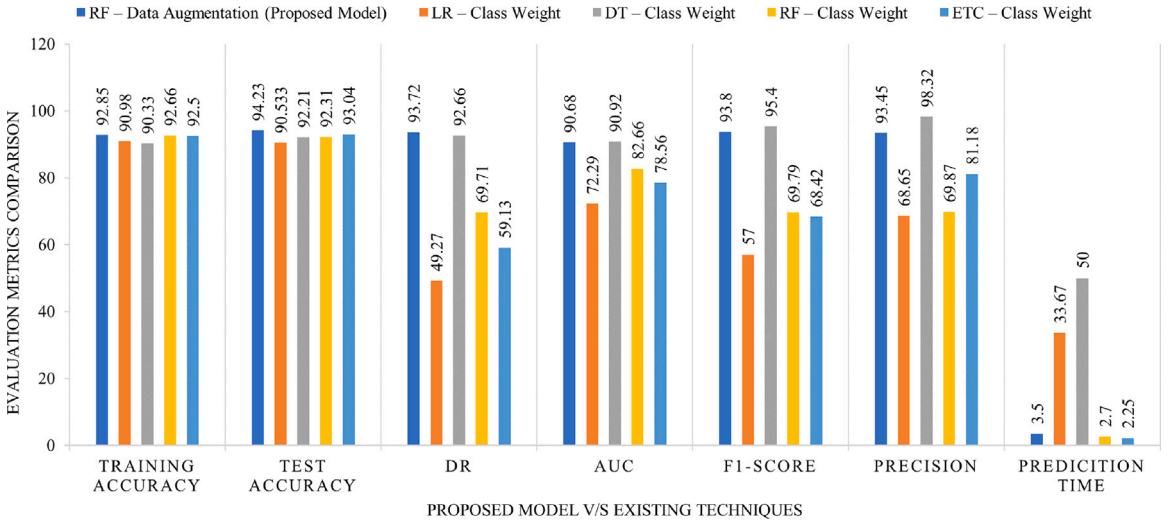

Fig. 12. Performance metrics of the proposed model using MinMax scaler with class weight ratio.

the model is being trained and tested. The proposed model is being constructed using system specification Intel Core i7-9750H @ 2.6 GHz Frequency and 16 GB RAM. The Table 18 represents the justified results.

Table 15
Comparison of proposed model with other algorithms using MinMax Scaler and data augmentation.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	FAR	F1-Score (%)	Precision (%)	Time (μs)
RF–Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	0.06	93.8	93.45	3.5
LR–Data augmentation	72.72	73.02	48.33	73.33	0.01	0.64.46	96.74	65.6
DT–Data augmentation	90.83	90.48	89.02	90.50	0.02	90.4	91.77	58.46
RF–Data augmentation	90.31	90.79	93.75	90.08	0.06	90.93	88.27	3.3
ET–Data augmentation	88.95	90.05	95.77	90.121	0.04	90.48	85.74	3.1

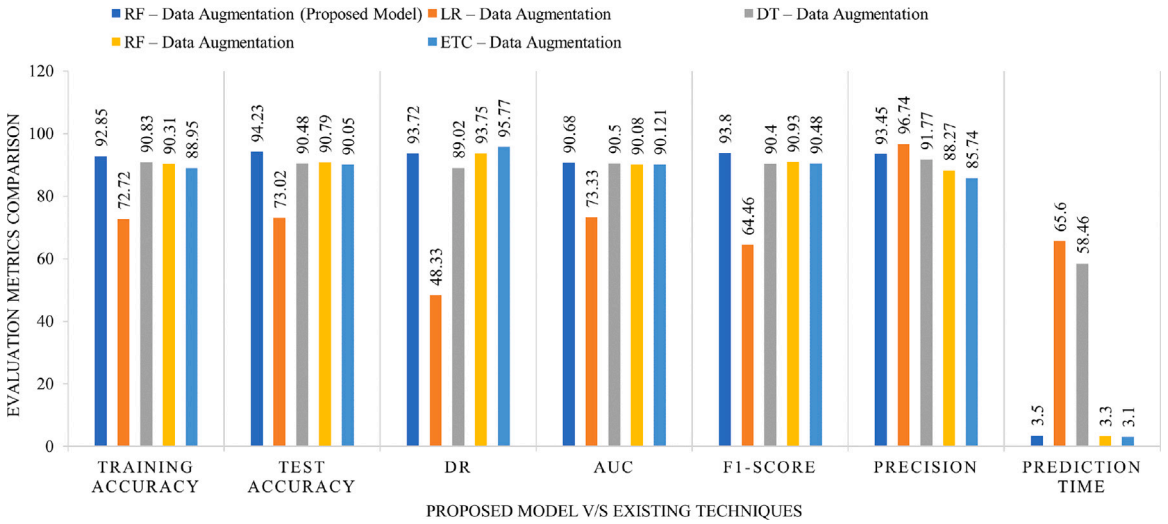

Fig. 13. Performance metrics of the proposed model using MinMax scalar with data augmentation.

Table 16
Confusion matrix of proposed model.

		True labels	
		0 (Normal)	1 (Attack)
Predicted labels	0(Normal)	2737 (TN)	177 (FP)
	1(Attack)	170 (FN)	2623 (TP)

Table 17
Comparison of proposed model with and without data augmentation.

Dataset status	Type of sample	Number of samples	Training accuracy (%)	DR (%)	AUC (%)	F1-Score (%)	Precision (%)
Without data augmentation	Benign	14,272	93.81	51.44	75.22	67.93	100
	Malignant	2046					
With data augmentation	Benign	14,272	94.23	93.72	90.68	93.80	93.8
	Malignant	14,272					

5.4. Effect of data size on DR

Without data augmentation i.e., with imbalanced dataset, the True Positive values were less which resulted in a poor Detection Rate i.e., is 51.44%. After performing the Data Augmentation there was an increase in data size which resulted in a good detection rate i.e., 93.72%. The following results are available in Table 19.

5.5. Limitations

The Dataset only focuses on two attacks of Man in the Middle Attack i.e., Data Alteration and Data Spoofing. In addition, the model has been simulated in a restricted network environment, which provided a combination of network and biometric data as an input to the model.

Table 18
Comparison of proposed model with other algorithms using MinMax Scaler and data augmentation.

Model	Training accuracy (%)	Testing accuracy (%)	DR (%)	AUC (%)	F1-Score (%)	Precision (%)	Time (μs)
Robust Scaler + RF + Data augmentation (Proposed model)	92.85	94.23	93.72	90.68	93.8	93.45	3.5
Robust Scaler + LR + Class weight ratio	89.88	89.00	55.52	74.70	5672	57.03	42
Robust Scaler + DT + Class weight ratio	89.71	91.81	89.90	91.00	73.69	62.43	41
Standard Scalar + RF + Class weight ratio	92.85	91.81	67.30	81.35	67.30	68.12	26.41

Table 19
Effect of data size on detection rate.

Data size	True negative	False positive	False negative	True positive	Benign samples	Malignant samples	Detection rate(%)
Without data augmentation	2848	0	202	214	14272	2046	51.44
With data augmentation	2737	177	170	2623	14727	14727	93.72

6. Conclusion and future work

The increase in the demand for IoMT services and the massive rise in the number of users requires robust security measures to counter any malicious activity. The security model for such distributive networks also needs a detection mechanism to identify if the data in the network has been tampered with by an intruder. The proposed model combines the random forest algorithm with the proper feature scaling method i.e, robust scaling. The model efficiently handles large and complex categorical data and effectively performs the classification in Intrusion Detection for IoMT networks. The methodology is highly appropriate for e-healthcare systems which deal with a vast amount of data. The proposed framework reduces the feature dimension and number of instances efficiently to improve the time of the classification process without compromising accuracy. The model reported an average accuracy of 94.23% and an F1-score of 93.8%. The performance of the proposed model has been analysed with the other ML based classification techniques namely logistic regression, decision tree and extra tree classifier. The results also infer that there is a significant reduction in the classification time. The reduction in time is achieved by processes involving scaling, elimination of extraneous features, encoding, and feature selection.

In future work, consideration for more attacks may be included. In addition, lightweight modelling of deep neural networks can be used for the development of the Intrusion Detection System for IoMT, to possibly improve the detection rate and AUC Score for better stability of the model. The work also has a scope of reducing the feature space using a combination of more than one dimension reduction algorithm.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.compeleceng.2022.108158.

References

[1] Gatouillat A, Badr Y, Massot B, Sejdić E. Internet of medical things: A review of recent contributions dealing with cyber-physical systems in medicine. IEEE Internet Things J 2018;5(5):3810–22. http://dx.doi.org/10.1109/JIOT.2018.2849014.

[2] Yanambaka VP, Mohanty SP, Kougianos E, Puthal D. PMsec: Physical unclonable function-based robust and lightweight authentication in the internet of medical things. IEEE Trans Consum Electron 2019;65(3):388–97. http://dx.doi.org/10.1109/TCE.2019.2926192.

[3] Sharma M, Pant S, Kumar Sharma D, Datta Gupta K, Vashishth V, Chhabra A. Enabling security for the Industrial Internet of Things using deep learning, blockchain, and coalitions. Trans Emerg Telecommun Technol 2021;32(7):e4137. http://dx.doi.org/10.1002/ett.4137, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.4137. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4137.

[4] Thamilarasu G, Odesile A, Hoang A. An intrusion detection system for internet of medical things. IEEE Access 2020;8:181560–76. http://dx.doi.org/10.1109/ACCESS.2020.3026260.

[5] Goel A, Sharma DK, Gupta KD. LEOBAT: Lightweight encryption and OTP based authentication technique for securing IoT networks. Expert Syst 2021;n/a(n/a):e12788. http://dx.doi.org/10.1111/exsy.12788, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/exsy.12788. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12788.

[6] Butun I, Morgera SD, Sankar R. A survey of intrusion detection systems in wireless sensor networks. IEEE Commun Surv Tutor 2014;16(1):266–82. http://dx.doi.org/10.1109/SURV.2013.050113.00191.

[7] Baburaj E, Rani AAV. Secure and intelligent architecture for cloud-based healthcare applications in wireless body sensor networks. Int J Biomed Eng Technol 2019.

[8] Chakraborty S, Aich S, Kim H-C. A secure healthcare system design framework using blockchain technology. In: 2019 21st International Conference on Advanced Communication Technology. 2019, p. 260–4. http://dx.doi.org/10.23919/ICACT.2019.8701983.

[9] Hady AA, Ghubaish A, Salman T, Unal D, Jain R. Intrusion detection system for healthcare systems using medical and network data: A comparison study. IEEE Access 2020;8:106576–84. http://dx.doi.org/10.1109/ACCESS.2020.3000421.
[10] Rao BB, Swathi K. Fast kNN classifiers for network intrusion detection system. Indian J Sci Technol 2017;10(14):1–10.
[11] Shapoorifard H, Shamsinejad P. Intrusion detection using a novel hybrid method incorporating an improved KNN. Int J Comput Appl 2017;173:5–9.
[12] Sarhan M, Layeghy S, Moustafa N, Portmann M. Towards a standard feature set of NIDS datasets. 2021, ArXiv abs/2101.11315.
[13] Sarhan M, Layeghy S, Moustafa N, Portmann M. NetFlow datasets for machine learning-based network intrusion detection systems. 2020, ArXiv abs/2011.09144.
[14] Farhan RI, Maolood AT, Hassan NF. Performance analysis of flow-based attacks detection on CSE-CIC-IDS2018 dataset using deep learning. Indones J Electr Eng Comput Sci 2020;20:1413–8.
[15] Hussain FB, Abbas SG, Fayyaz UU, Shah GA, Toqeer A, Ali A. Towards a universal features set for IoT botnet attacks detection. 2020, ArXiv abs/2012.00463.
[16] Kumar P, Gupta GP, Tripathi R. An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for IoMT networks. Comput Commun 2021;166:110–24. http://dx.doi.org/10.1016/j.comcom.2020.12.003, URL https://www.sciencedirect.com/science/article/pii/S0140366420320090.
[17] Rathore H, Al-Ali AK, Mohamed A, Du X, Guizani M. A novel deep learning strategy for classifying different attack patterns for deep brain implants. IEEE Access 2019;7:24154–64. http://dx.doi.org/10.1109/ACCESS.2019.2899558.
[18] Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. J Big Data 2019;6:1–48.
[19] Chawla N, Bowyer K, Hall LO, Kegelmeyer WP. SMOTE: Synthetic minority over-sampling technique. J Artificial Intelligence Res 2002;16:321–57.
[20] Ho TK. Random decision forests. In: Proceedings of 3rd international conference on document analysis and recognition. vol. 1, 1995, p. 278–82. http://dx.doi.org/10.1109/ICDAR.1995.598994.
[21] Gron A. Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques To Build Intelligent Systems. 2nd ed.. O'Reilly Media, Inc.; 2020.
[22] Claesen M, Moor BD. Hyperparameter search in machine learning. 2015, ArXiv abs/1502.02127.
[23] Hosmer Jr DW, Lemeshow S, Sturdivant RX. Applied Logistic Regression. vol. 398, John Wiley & Sons; 2013.
[24] Raschka S, Mirjalili V. Python Machine Learning. 2nd ed.. Birmingham, UK: Packt Publishing; 2017.
[25] Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. Mach Learn 2006;63:3–42.

Karan Gupta is currently pursuing his master's degree in technology at the Netaji Subhas University of Technology. During his Master's programme, he focused in the field of Mobile Computing and Security. Before that, he obtained his B.Tech degree in Computer Science and Engineering from S.R.M. University, Haryana.

Deepak Kumar Sharma is working as an Associate Professor in the Department of Information Technology, Indira Gandhi Delhi Technical University for Women (IGDTUW), Delhi, India. Earlier, he has worked as Assistant Professor at Netaji Subhas University of Technology (formerly N.S.I.T.), Delhi. His research interests include opportunistic networks, wireless ad hoc and sensor networks, Software Defined Networks and IoT Networks

Koyel Datta Gupta is a gold medalist in M.Tech (Computer Technology) from Jadavpur University, Kolkata (2007). She received her Doctorate in 2015 from Jamia Milia Islamia, New Delhi. Her areas of research include Network Security, Digital Signal Processing, Pattern Recognition and Machine Learning. She is currently working as Associate Professor in Maharaja Surajmal Institute of Technology (MSIT) (under the GGSIP University), New Delhi.

Anil Kumar is currently working as a Professor CSE, Accreditation Coordinator, and Head-Data Science Research Group, DIT University. He received his M.Tech from Delhi College of Engineering, Delhi, and Ph.D. from Manipal Group. He has more than 24 years of teaching and industrial experience. His research interests include Image processing algorithms, Cryptography, Artificial Intelligence, Signal and System, Neural Systems, and Genetic algorithms.