International Journal of Informatics and Communication Technology (IJ-ICT)
Vol. 15, No. 2, June 2026, pp. 565~577
ISSN: 2252-8776, DOI: 10.11591/ijict.v15i2.pp565-577

Ensemble windows intrusion detection system using XGBoost and deep learning

Pranitha Kedambady Shiva, Pushparaj D. Shetty
Department of Mathematical and Computational Sciences, National Institute of Technology Karnataka (NITK), Surathkal, India

Article Info

Article history:

Received Aug 7, 2025
Revised Jan 7, 2026
Accepted Mar 30, 2026

Keywords:

Cyber-attacks
Deep neural networks
Intrusion detection
Random forest
XGBoost

ABSTRACT

Intrusion detection systems (IDS) are critical for preserving the Windows environment from an ever-changing collection of cyber threats. Current IDS uses deep learning (DL), which are heavy models if used for detection, while others use machine learning (ML) techniques, which require external feature extraction. To resolve this challenge, this paper introduces XGBNN, a new ensemble model that combines the benefits of ML and DL to identify and mitigate attacks against Windows machines effectively. The various ML methods are trained on the publicly available dataset to classify eight types of attacks in a Windows environment. Additionally, deep neural networks (DNNs) are proposed by optimizing the layers and hyperparameters to achieve the best accuracy. Then, the DNN model and XGBoost model are integrated to detect intrusions by utilizing the feature extraction ability of DNN and providing the intermediate features extracted from the last second layer of the DNN to the XGB for classification. The Ensemble model XGBNN optimizes features and offers better decisions. The proposed model achieves an exceptional accuracy of 100%, as demonstrated by the empirical results, and outperforms the benchmark models with an improvement of 0.004%. The purpose of this study is to highlight the effectiveness of hybrid architectures in intrusion detection. These architectures offer a more robust, scalable, and effective method to improve the security of the Windows system against more sophisticated attacks.

Corresponding Author:

Pranitha Kedambady Shivppa
Department of Mathematical and Computational Sciences
National Institute of Technology Karnataka (NITK)
Surathkal, India
Email: pranithaks12@gmail.com or pranithaks.217ma501@nitk.edu.in

1. INTRODUCTION

The number of linked computers and the volume of data exchanged between them are growing exponentially in tandem with ongoing technological breakthroughs. This condition enables cybercriminals to focus their attention and satisfy their needs. Cyberattacks, such as malware and other network attacks, seek to undermine the security, dependability, and accessibility of a system or a linked device [1]. Thus, detecting cyberattacks is essential to computer network and component security. Consequently, numerous Intrusion detection systems (IDS) have been proposed to identify and detect intrusions within the system. IDS detect, evaluate, and report any suspicious activity they find in order to prevent unauthorized access to computer networks and systems [2]. IDS monitors computer and network activities, collects data about them, and checks this information for harmful actions [3]. If they detect a suspicious attack or activity, they will alert the system administrator [4].

Network-based IDS (NIDS) and Host-based IDS (HIDS) are two types of IDS and are categorized based on where they get their data. IDS uses detection in various ways, but two of the most common ones are signature detection and anomaly detection [5]. By monitoring and analyzing network traffic, NIDS detect anomalous activity or network-based attacks. NIDS are typically implemented on network infrastructure to supervise all network activities. The NIDS has a monitoring phase where it collects data about network traffic, such as packet size, number of packets, port number, and IP address. In order to identify malicious network activity, NIDS employ signature and anomaly detection methods to examine collected data. During the reaction phase, any identified attack or strange IP address in the network or activity is communicated to the network administrator for suitable intervention [6].

In contrast to NIDS, HIDS observes an individual host or device to identify any illegal activities. HIDS systems are designed to include the ability to monitor, identify, and respond to unusual behaviors that are relevant to the host device [7]. Similar to NIDS, it also has Signature-based detection techniques that identify assaults by correlating certain behaviors with established attack signatures [8].

The identification of the profile that outlines abnormal process activities is established through the use of ML in IDS [9]. These approaches are trained on standard data samples. An anomaly-based HIDS will notify the system administrator of any unusual or harmful processing activity if it deviates from this profile. IDS often uses ML and DL methods that autonomously identify unexpected and unpredictable attacks [10], [11].

This paper proposes an innovative hybrid strategy that uses an ML approach to detect Windows 10 attacks. As ML and DL techniques can identify intricate patterns and behaviours in system data, they have both demonstrated significant effectiveness in HIDS. HIDS is capable of compiling a vast and diverse array of information by monitoring a wide range of events, including system calls, file modifications, and network traffic. Rule-based systems typically have difficulty recognizing complex and ever-changing attack patterns, mainly when the environment is in a state of constant change. Nonetheless, ML and DL algorithms have demonstrated superiority in this domain owing to their ability to autonomously learn from data and detect anomalies or harmful activities that diverge from standard patterns.

In order to find likely incursions, ML techniques can examine data from multidimensional systems and reveal subtle relationships between various features. This idea is advanced by deep learning models, particularly neural networks, which make use of multiple layers of representation. This enables them to differentiate between more complex and non-linear correlations within the data. Deep learning algorithms are very good at finding advanced risks, like zero-day attacks, which often leave minor signs in the activities of a system [12]. Furthermore, ML and DL approaches' ability to continuously improve and adapt through learning allows them to advance in tandem with emerging threats.

A lot of research has been conducted to develop various techniques for classifying attacks in multiple environments. Figueiredo *et al.* [13] presented the DL-based IDS [13] that uses the stacking long-short-term memory (LSTM) model and the improved pre-processing approach, returns context-free information, and achieves over 99% accuracy on the CICIDS 2017 dataset. Khan *et al.* [14] utilizes a recurrent neural network combined with gated recurrent units (RNN-GRU) to classify attacks. The data used is the ToN-IoT dataset, which was integrated with the three-tiered IoT system and consists of novel attack types not found in other open-source datasets. The recommended model attains an accuracy of 88% for the Windows 10 dataset.

Similarly, Lilhore *et al.* [15] introduced the combination of OCNN-LSTM and a transfer learning approach. The suggested model employs an improved convolutional neural network (CNN) by enhancing its parameters through the grey wolf optimization technique, which refines the CNN hyperparameters and improves the CNN's accuracy. The recommended model attained a precision of 92.7% on the Windows 10 dataset. Awotunde *et al.* [16] also used ML techniques on TON-IoT datasets and achieved the highest accuracy of 99.73%.

Likewise, Chalichalamala *et al.* [17] recommend an ensemble classifier based on logistic regression (LR) for the efficient development of IDS. The suggested model also integrates AdaBoost and random forest (RF) to create a proficient classifier employing the iterative ensemble methodology. The adaptive synthetic sampling method mitigates the problem of data imbalance. Moreover, unsuitable features are removed by recurrent feature elimination. The suggested model attains an accuracy of 99.99% for the BoT-IoT dataset, surpassing that of TL-IDS and LSTM. Singh *et al.* [12] select attributes using a mix of the information gain calculation and the simulated annealing technique. The suggested DL-based model achieves remarkable accuracies of 99.990% and 99.996% on the TON-IOTwin7 and TON-IOTwin10 datasets [18], [19].

Similar to the above approach, Cherfi *et al.* [20] suggested a DL-IDS model that integrates an autoencoder with a deep neural network (DNN). The information gain assessment and the simulated annealing approach are used for attribute selection. The highest accuracy of 99.90% and 99.86% is achieved on datasets TON-IOTwin7 and TON-IOTwin10, respectively. Alternatively, Ding *et al.* [21] introduce an alternative DL model for the detection of cyberattacks targeting IoT devices. DeepAK-IoT is founded on three components: the blocks for spatial modeling based on residuals (RSR), temporal modeling, and

detection. A skip link is incorporated into each RSR block of the four convolutional layers, which is done in order to prevent vanishing or expanding gradients. These layers are connected in parallel. The last block classifies the input. The recommended model's accuracy is 90.57% for TON IoT.

The analyzed works emphasize the increasing application of sophisticated ML and DL methodologies in the creation of efficient IDS, particularly in managing intricate, large-scale datasets within diverse network settings. Techniques such as LSTM, GRU, CNN, and hybrid models have been effectively utilized, demonstrating excellent accuracy in intrusion detection, particularly in IoT and industrial networks. These models have been augmented with many optimization techniques, including grey wolf optimization and sophisticated optimizers like Adam, to boost performance. Prevalent issues such as data imbalance, feature selection, and the classification of new attacks are still present. Thus, this paper proposes a novel hybrid approach by utilizing the DNN and XGBoost models to develop an IDS that can detect attacks and classify them into various classes. Hence, the significant contribution of this research is given below.

This paper introduces a novel hybrid model that combines DNN for feature extraction and ML for classification, resulting in improved accuracy and robustness.

- The various ML algorithms, such as Gaussian naïve Bayes (GNB), LR, K nearest neighbors (KNN), decision tree (DT), RF, and XGB, are trained on the open-source datasets.
- A customized DNN architecture has been developed and trained using the same dataset.
- Intermediate Feature Extraction: The model extracts detailed features from the second-to-last layer of the Customized DNN and uses these features as input for the XGB classifier, which significantly improves performance compared to using raw data.
- The study also emphasizes data scaling through standardization to normalize input characteristics and one-hot label encoding for optimal input for the DNN and XGB classifiers.
- The hybrid model is proposed by combining the DNN's representational power with XGB's ability to refine decision bounds.

2. RESEARCH METHOD

The proposed method aims to integrate the DNN and the XGB model to develop novel ensemble techniques. Different ML techniques are trained on the open-source dataset to identify and then classify the attack into eight distinct classes. The DNN model is fine-tuned and optimized on the same dataset to achieve the optimum performance. Finally, to take advantage of the DNN and boosting method, the new model XGBNN was developed, which is the interaction of DNN and XGB. The trained DNN model's intermediate features from the last second layer are used as input to the XGB model. Figure 1 provides a visual representation of the proposed approach. The proposed Algorithm 1 for the Hybrid IDS is given below.

Algorithm 1: XGBNN Algorithm

Input: Training data $X_{train} \in \mathbb{R}^{n \times d}$ and labels $Y_{train} \in \mathbb{R}^{n}$, where n is the number of samples and d is the number of features.

Output: Classify the instances into eight classes of attacks

Step 1: Data Pre-processing

Step 1.1.: Normalize the input data using the formula as $X_{train} = \frac{X_{train} - \mu_{train}}{\sigma_{train}}$ and $X_{test} = \frac{X_{test} - \mu_{test}}{\sigma_{test}}$, where μ_{train} is the mean of the training instances, and σ_{train} is the standard deviation of the training instances.

Step 1.2: Apply label encoding as $Y^{(i)} = \begin{cases} 1, if\ y^{(i)=c} \\ 0, otherwise \end{cases}$, $\mathrm{i} = 1,..,\mathrm{n}, \mathrm{g} = 1,..,\mathrm{G}$, where G is the number of classes, and Y is the label feature matrix.

Step 2: Develop a customized DNN model

Input: X_{train} with shape (n x d), where d is the dimensionality of the features.

DNN architecture is defined as:

Input layer: $h_0 = X_{train}$

Hidden Layer 1: $h_1 = f(W_1 h_0 + b_1)$, where f is the RELU activation function $f(x) = \max(0, x)$, $W_1 \in \mathbb{R}^{1024 \times d}$, and $b_1 = \mathbb{R}^{1024}$.

Hidden layer 2: $h_2 = f(W_2 h_1 + b_2)$, where $W_2 \in \mathbb{R}^{512 \times d}$, and $b_2 = \mathbb{R}^{512}$.

Hidden layer 3: $h_3 = f(W_3 h_2 + b_3)$, where $W_3 \in \mathbb{R}^{256 \times d}$, and $b_3 = \mathbb{R}^{256}$.

Output layer: $\hat{y} = softmax\,(W_4 h_3 + b_4)$, where $W_4 \in \mathbb{R}^{C \times 256}$, and $b_4 = \mathbb{R}^{C}$

Loss Function (Categorical Cross Entropy) is calculated as $L(y, \hat{y}) = -\sum_{i=1}^{n} \sum_{c=1}^{C} y_{i,c}\ log\, \hat{y}_{i,c}$

Step 3: Training the DNN model

Step 3.1: Use ADAM optimizer with learning rate η and Train for E epochs

Step 3.2: Early Stopping: If the validation loss does not improve for p consecutive epochs, training should be terminated.

Step 4: Extract the intermediate features from DNN

Step 4.1: After training DNN, use the last second layer dense layer h_3 as the feature map.

Step 5: Train different ML techniques to find the best among them

Step 6: Train the highest-performing ML model with the intermediate features from h_3 layer.
Step 7: Final evaluation of the proposed Hybrid model
Step 7.1: Combine DNN softmax predictions and the highest-performing ML model's predictions to get the final classification result.
$\hat{y}_{final} = \alpha . \hat{y}_{dnn} + (1 - \alpha) . \hat{y}_{ML}$, where α is the weighting factor
Step 7.2: Measure accuracy, recall, precision, F1 score, and Area under curve

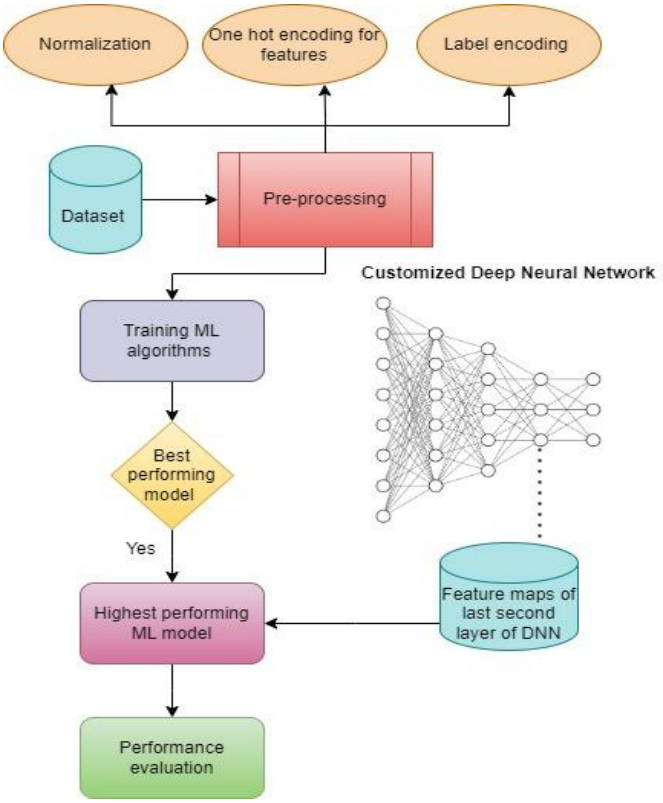

Figure 1. The flow chart of the proposed hybrid DNN+ML IDS

2.1. Dataset description

The dataset used for the experimentations is the open-source dataset "Windows 10" from the TON_IoT dataset UNSW, Sydney 19. In Windows 10, new features are extracted from the processor, memory, processes, and hard disk audit trails using an innovative IoT network design. Three tiers-edge, fog, and cloud-were used to implement the testbed. There are three levels to a network: the edge, which houses network devices and the IoT; the fog, which contains virtual machines and gateways; and the cloud, which includes services offered by the cloud, like statistical analysis and visualization.

Software-defined networking and network function virtualization dynamically administered the levels using the VMware NSX and vCloud NFV platforms. Throughout the construction and deployment of the testbed, both standard and malicious activities were executed to gather annotated samples of data based on an authentic ground truth table for evaluating the effectiveness of innovative cybersecurity solutions [18]. There are 21,104 instances and 126 features in the Windows 10 dataset, comprising 10,000 normal records and 11,104 malicious records classified into eight categories: DDoS, DoS, injection, normal, xss, password, scanning, and MITM [19]. The number of instances in each category of the dataset is given in Figure 2. The highest number of attacks is for DDoS, which has 4,608 samples.

2.2. Data pre-processing

The data is processed to normalize it for better performance. Normalization is an essential preprocessing procedure for datasets, particularly those including many characteristics with disparate scales. Normalization [22] means standardizing all features to a uniform scale, hence enhancing the efficacy of ML algorithms. In the absence of normalization, features with greater numerical ranges may overshadow the learning process, resulting in biased or inferior model performance.

A prevalent method of normalization is standardization, commonly referred to as Z-score normalization. This approach normalizes each feature to have a mean of 0 and a standard deviation of 1. When the data is close to being normally distributed, standardization becomes even more useful because it ensures that each feature provides an equal contribution to the model. For a feature x, the normalized value x' is calculated as given in (1).

$$x' = \frac{x-\mu}{\sigma} \quad (1)$$

where x is the original feature, μ is the mean, and σ is the standard deviation.

Label encoding [23] transforms each distinct category into an integer number. This method is straightforward and decreases the data's dimensionality as it does not generate additional binary features.

The benefits of label encoding encompass its simplicity and efficiency, particularly in the presence of numerous unique categories. Nonetheless, the drawback is that it may create unwanted associations across categories, resulting in erroneous predictions. For example, if an ML model perceives higher values as "superior" or "enhanced," it may erroneously infer a hierarchy among the encoded categories.

Label encoding is suitable when there exists an ordinal relationship among the categories, such as "Low," "Medium," and "High," or when employing algorithms that do not regard the encoded numbers as possessing an order, such as decision trees. The equation for calculating the label encode,

$$Y^{(i)} = \begin{cases} 1, if\ y^{(i)=g} \\ 0, otherwise \end{cases} \quad (2)$$

for, $i = 1,..,n,$ and $g = 1,..,G$. where G is the number of classes, and Y is the label feature matrix.

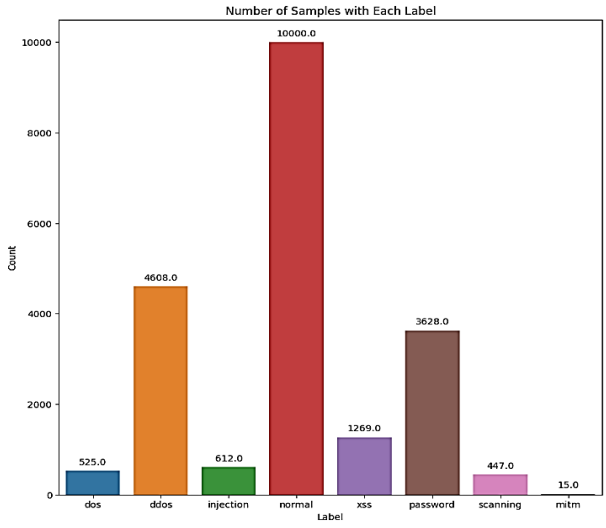

Figure 2. Number of samples in each label

2.3. Deep learning architecture

The customized DNN, which consists of an architecture, is proposed, as shown in Figure 3. The diagram depicts a sequence of four completely linked (dense) layers within a neural network. Every layer is designated with its name, input dimensions, and output dimensions.

Layer dense_16: This constitutes the inaugural dense layer within the network. The input has a shape of (None, 125), where "None" signifies a variable batch size, and 125 denotes the dimensionality of the input features. The output dimension of this layer is (None, 1024), indicating it converts the input into a 1024-dimensional space. This suggests that the layer comprises 1024 neurons, with each neuron linked to every input characteristic.

Layer dense_17: The output from the preceding layer, a 1,024-dimensional vector, is utilized as input for this layer. The input form is (None, 1,024), while the output is diminished to (None, 512), signifying that this layer condenses the feature space to 512 dimensions. This is accomplished via 512 neurons in this layer.

Layer dense_18: The third layer processes a 512-dimensional input and reduces it to a 256-dimensional output, indicated by its output shape of (None, 256). This compression perpetuates the process of feature abstraction and reduction via a fully connected layer.

Layer dense_19: The concluding dense layer compresses the input from 256 dimensions to 8 dimensions, as evidenced by the output shape (None, 8). This layer probably functions as the output layer or a component of the ultimate decision-making process in the model, contingent upon the specific task (e.g., classification with eight potential classes or other task types).

This architecture exemplifies a conventional framework that systematically diminishes the feature space from a higher-dimensional input, 125 features, to a significantly lower-dimensional output, 8 classes, via a series of dense layers. Every layer function to capture and abstract varying levels of feature representations [24].

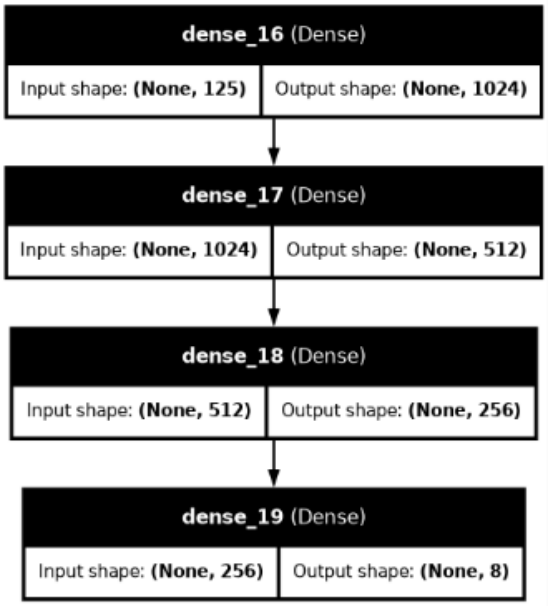

Figure 3. Proposed customized DNN

2.4. Machine learning algorithm

The various ML algorithms [25] trained are described below. GNB is a probabilistic classification technique derived from Bayes' Theorem, predicated on the premise that all features are independent and conform to a Gaussian distribution [26]. LR is a linear model employed for binary classification applications. It calculates the likelihood of a data point being classified into one of two categories by a logistic function, yielding a value ranging from 0 to 1 [27]. KNN is a straightforward and intuitive instance-based learning technique. Rather than constructing an explicit model, it categorizes new data points by aggregating a majority vote from the "k" nearest data points in the training set [28]. DT are non-linear models that partition data into subsets according to feature values, forming a tree-like structure in which each internal node signifies a decision rule, and each leaf node denotes a predicted class label [29]. RF is an ensemble learning method that mitigates the overfitting problem of decision trees through the aggregation of several trees. Each tree is trained on a distinct random subset of the data and characteristics, with the final prediction derived from aggregating the predictions of all trees (bagging) [30]. Decision trees serve as the essential learners in (XGB), an improved version of gradient boosting. Regularization, parallel processing, and the handling of missing data are some of the improvements that give it its speed and effectiveness [31].

2.5. Proposed hybrid model

The DNN is trained to extract the features from the last second layer, and these features are provided to the XGB for classification. DNNs are developed to acquire hierarchical representations of data. As input data traverses the several layers of a DNN, the model acquires progressively intricate features at each layer. The lower layers identify the most notable features like borders and background. The higher layers are responsible for more insightful information, such as the shape and the kind of objects. Extracting feature

maps from intermediate layers yields rich and valuable data representations that encapsulate essential information, advantageous for subsequent tasks. Utilizing these attributes allows the future ML model to capitalize on existing knowledge, hence diminishing the need for training data and frequently enhancing performance. Intermediate feature maps often function as a method of dimensionality reduction. Rather than utilizing raw data with possibly large dimensionality, the feature maps distill pertinent information in a more accessible manner.

2.6. Performance measures

The various standard performance metrics are used to evaluate the performance of the proposed hybrid model. The equations of accuracy, recall, precision, and F1 score are given in (3)-(6), respectively [32].

Accuracy: The ratio of correctly predicted cases to the total number of instances. An uneven dataset may yield deceptive results.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ instances} \tag{3}$$

Recall indicates the proportion of actual positive cases that the model successfully identified. It is advantageous when the potential for false negatives exists.

$$Recall = \frac{True\ Positives}{True\ Postives + False\ Negatives} \tag{4}$$

Precision: It determines the proportion of correctly anticipated positive cases relative to all instances predicted as positive. False positives are beneficial when they are necessary.

$$\text{Precision} = \frac{True\ Positives}{True\ Postives + False\ \text{Positives}} \tag{5}$$

F1 score: It is the harmonic mean of Precision and Recall. It is used in the case of class imbalance.

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

3. RESULTS AND DISCUSSION

The experiments are performed on Google Collab using the P100 graphical processing units (GPUs) for faster training of deep learning algorithms. This paper identifies the best ML algorithm by implementing various ML and DL techniques. The hyperparameters used for the Customized DNN are given in Table 1. The evaluation of models according to precision, accuracy, recall, and F1-score reveals differing performance levels, as shown in Table 2.

Table 1. Finetuned Hyperparameters for DNN

Hyperparameters	Value
Number of Dense layers	3
Epochs	100
Batch size	128
Loss	Cross Entropy

Table 2. Comparative analysis of all the ML, DL, and proposed hybrid algorithm

Model	Accuracy	Precision	Recall	F1 score
GNB	89.91%	90.90%	89.91%	89.09%
LR	99.90%	99.85%	99.90%	99.88%
KNN	99.76%	99.77%	99.76%	99.76%
DT	99.95%	99.95%	99.95%	99.95%
RF	99.95%	99.95%	99.95%	99.95%
XGB	99.91%	99.85%	99.91%	99.88%
DNN	99.95%	99.95%	99.95%	99.95%
Proposed XGBNN	100%	100%	100%	100%

As given in Table 2, GNB has the least effective performance, achieving an accuracy of 89.91% and an F1-score of 89.09%, attributable to its basic assumptions that constrain its capacity to manage intricate

data patterns. LR achieves a notable accuracy of 99.90% and an F1-score of 99.88%, indicating reliability. However, it remains marginally inferior to advanced models. KNN attains an accuracy and F1-score of 99.76%, slightly inferior to LR, perhaps because of its susceptibility to the selection of the "k" parameter. The DT and RF both attain 99.95% across all metrics, which indicates their effectiveness and high decision rate. However, XGB attains 99.86% across all metrics, which makes it the second-best-performing model. DNN also achieved 99.95%, which indicates that more prominent features are extracted than with XGB. Finally, the highest accuracy is observed by the proposed XGBNN model, which exceeds all the benchmarked methods by achieving 100 % accuracy along with other performance metrics, hence proving its excellence in precision, accuracy, recall, and F1-score.

With the combination of DNN feature extraction and XGB, it is possible to take advantage of the benefits that both models offer, which ultimately results in greater performance in comparison to each model when used alone. The layered architecture of DNNs enables them to effectively acquire complex, high-level, and abstract features from data, thereby identifying nuanced patterns that may be obscured in the raw data.

However, even though DNNs are quite powerful, they can sometimes experience overfitting or require a lot of adjustment to improve performance. When it comes to managing tabular data with minimal bias and volatility, XGB, an incredibly efficient gradient-boosting approach, is second to none. By means of XGB's capacity for producing thorough, high-quality data representations, using features obtained from the DNN as input helps the system to use this capability, hence improving XGB's efficacy in classification or regression tasks. This hybrid technique improves performance by combining deep learning models' representational capabilities with XGB's precision and efficiency.

The Confusion matrix for GBN, LR, KNN, DT, RF, XGB, and the proposed model is given in Figure 4. The heatmap of the confusion matrix illustrates the efficacy of a multi-class classification model over eight categories (designated 0 to 7), where 0 represents DDoS, 1 represents DoS, 2 represents injection, 3 represents MITM, 4 represents normal, 5 represents a password, 6 represents scanning, and 7 represents xss. Each cell denotes the count of cases categorized as a specific class, with rows reflecting the actual labels and columns reflecting the anticipated labels. The diagonal cells, extending from the top left to the bottom right, signify accurate classifications. The highest misclassification is observed in GBN, as shown in Figure 4(a). Class 4, i.e., normal class, has the most accuracy, with 1,015 occurrences accurately categorized, and class 2, i.e., injection, demonstrates the lowest, with merely 35 correct predictions. Misclassifications occur in the off-diagonal cells, where, for instance, 37 instances of class 0, i.e., DDoS, were erroneously categorized as class 7, i.e., xss, and 75 instances of class 0, i.e., DDoS, were misclassified as class 5, i.e., password. Class 6, i.e., scanning, exhibits minimal misclassifications, with only one occurrence erroneously categorized as class 7, i.e., xss. This matrix delineates the model's strengths and weaknesses across many classes, offering insights into potential areas for enhancement.

The confusion matrix for LR is given in Figure 4(b), which shows 2 missclassifications; for KNN is given in Figure 4(c), which shows 3 missclassifications; for DT is given in Figure 4(d) which shows 1 missclassification; for RF is given in Figure 4(e) which shows 1 missclassifications, for XGB is given in Figure 4(f) which shows 2 missclassifications, for DNN is given in figure 4(g) which shows 2 missclassifications, and for proposed XGBNN is given in Figure 4(h), which shows no missclassifications.

However, all the other algorithms have very few misclassifications, and the Proposed XGBNN has no misclassifications, thus having 100 % accuracy. The accuracy graph shown in Figure 5 illustrates a rapid enhancement in the DNN model's performance on both the training and validation datasets. The training accuracy, denoted by the blue line, begins at a reasonably elevated level and approaches 99.95% during the initial epochs, signifying that the model rapidly assimilates the patterns in the training data. Correspondingly, the validation accuracy (shown by the orange line) increases significantly in the early epochs and subsequently stabilizes near 99.95%, implying that the model generalizes effectively to unseen data. The elevated, consistent accuracy for both training and validation datasets indicates that the model is functioning effectively without considerable overfitting.

The training and validation losses, which measure prediction error across the same epochs, are depicted in the loss graph in Figure 6. As the blue line shows, the training loss starts higher but quickly drops down in the early epochs, getting close to zero, which means the model is doing a good job of eliminating errors in the training data. The validation loss represented by the orange line reduces significantly over the initial epochs and stabilizes near zero, showing that the model's predictions remain accurate on previously unknown data. The proximity of the training loss level and the validation loss level, both of which are close to zero, lends confidence to the idea that the model has not overfit and is effectively generalizing knowledge.

Figure 5 indicates that the DNN model is effectively optimized. It attains elevated accuracy and reduced loss on both the training and validation datasets, exhibiting negligible divergence between them. The equilibrium of high accuracy and minimal loss on unknown data signifies that the model has acquired knowledge efficiently without overfitting, rendering it dependable for practical applications.

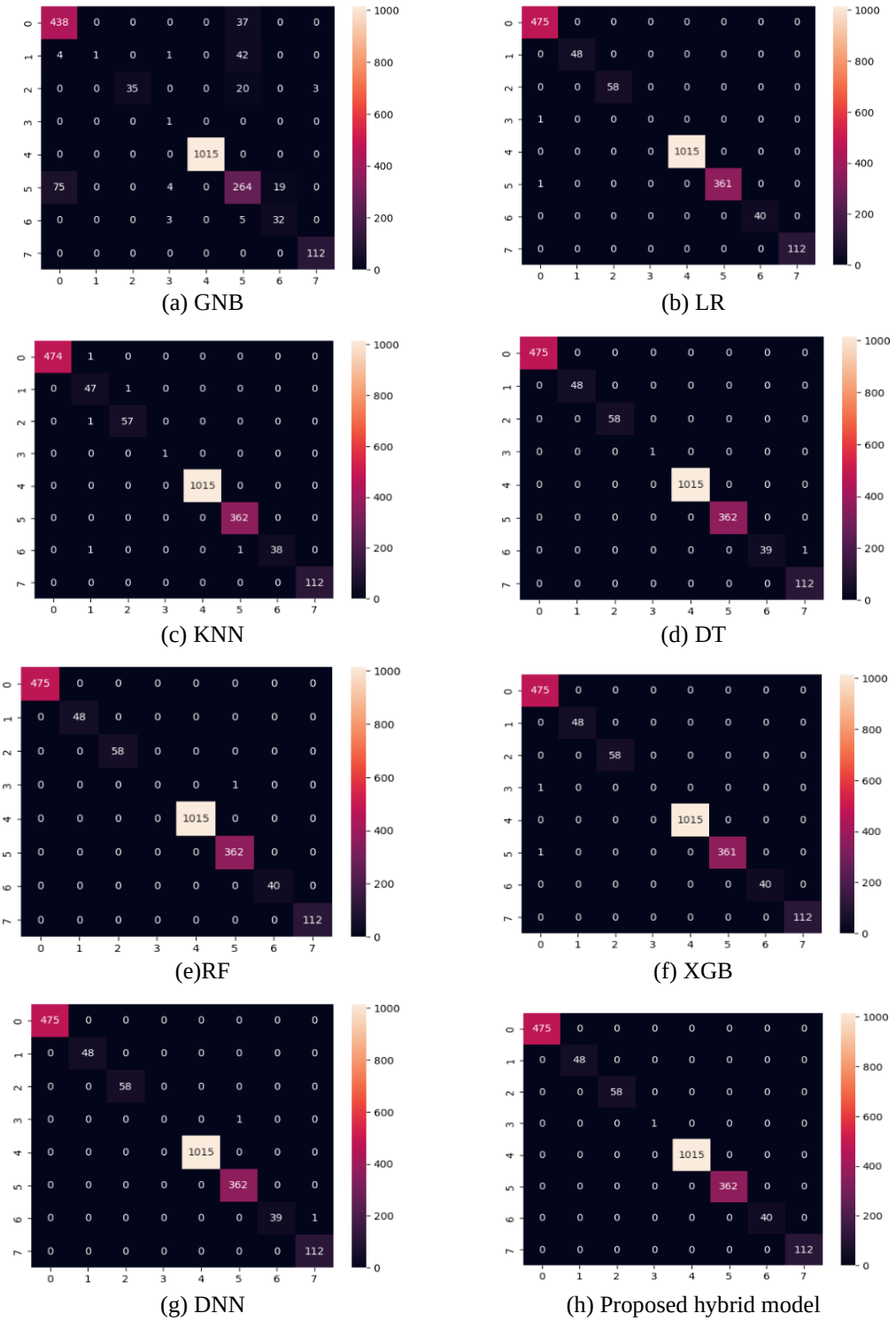

Figure 4. The confusion matrix where 0 represents DDoS, 1 represents dos, 2 represents injection, 3 represents mitm, 4 represents normal, 5 represents a password, 6 represents scanning, and 7 represents xss. a) GBN; b) LR; c) KNN; d) DT; e) RF; f) XGB; g) DNN; h) Proposed XGBNN

Figure 7 illustrates the receiver operating characteristic (ROC) curves for a multi-class classification model GNB encompassing eight classes, designated 0 to 7. This graph illustrates that classes 4 and 7 exhibit the most significant AUC values of 1.00, indicating near-optimal efficacy in differentiating these groups from the remainder. Classes 0, 2, and 6 exhibit excellent performance, evidenced by high AUC values of 0.99 and 0.98, signifying the model's robust capacity to distinguish these classes. Classes 1 and 5 exhibit marginally

lower AUC values of 0.95; however, these are still suggestive of commendable performance. The elevated AUC values for all classes demonstrate that the model effectively distinguishes each class from the others with negligible confusion. This indicates robust model performance and efficient categorization across several classes. Figure 8 shows the ROC curve for LR, RF, XGB, DNN, and Proposed XGBNN, which indicates that all the classes have an AUC of 1, while Figure 9 shows the ROC curve of the KNN and DT classifiers, indicating that all the classes have an AUC of 1, except class 6, i.e., scanning. The proposed XGBNN algorithm is compared with the benchmarked techniques, as shown in Table 3.

Figure 5. Accuracy over the epochs for DNN

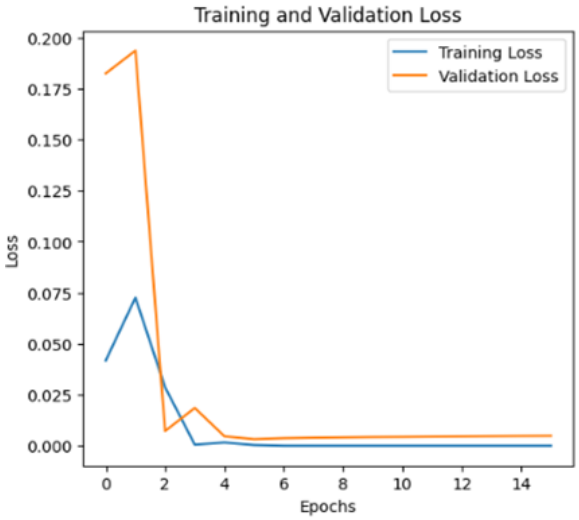

Figure 6. Loss over the epochs for DNN

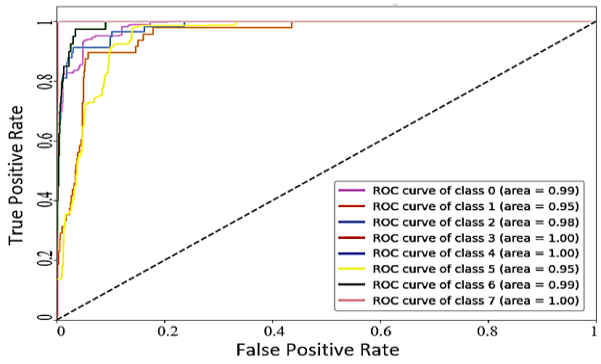

Figure 7. ROC of GNB

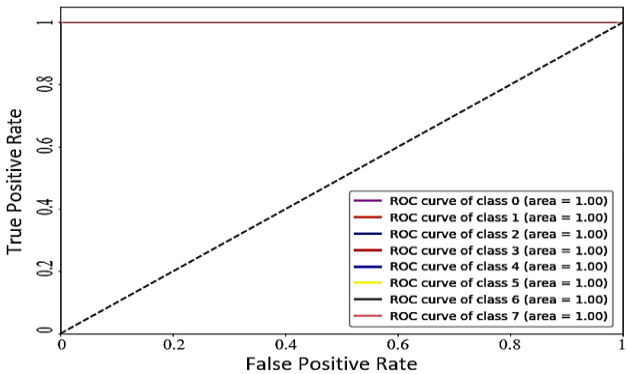

Figure 8. ROC of LR, RF, XGB, DNN, and Proposed XGBNN

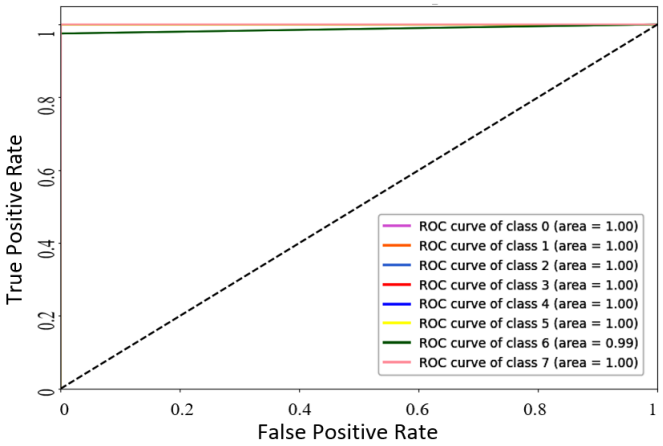

Figure 9. ROC of KNN and DT

Table 3. Comparison with benchmark methods

Authors	The method proposed or used	Accuracy	Precision
Khan *et al.* [14]	RNN-GRU with optimization techniques ADAM and ADAMX	88%	-
Lilhore *et al.* [15]	OCNN-LSTM with the transfer learning model	-	92.7%
Awotunde *et al.* [16]	ML algorithms	99.73%	-
Chalichalamala *et al.* [18]	DL-based techniques with information gain calculation and the simulated annealing technique	99.996%	-
Ding *et al.* [21]	Integration of an autoencoder with a DNN	99.86%	-
Singh and Singh [22]	DeepAK-IoT (TON_IoT)	90.57%	-
Proposed XGBNN	Integration of XGB and Customized DNN	100%	100%

The suggested XGBNN methodology, which integrates XGB with a customized DNN, achieved a benchmark of flawless accuracy and precision at 100%. These developments highlight the revolutionary capacity of hybrid and deep learning methodologies in addressing intricate challenges.

4. CONCLUSION

This paper presents XGBNN, a novel ensemble model that integrates DL and ML methodologies to improve the detection of cyberattacks on Windows systems. XGBNN attains an unparalleled accuracy and precision of 100% by combining a custom DNN for feature extraction with the robust classification capabilities of XGB. The comparative comparison with existing methods indicates that the suggested model surpasses conventional ML and independent deep learning techniques, highlighting its robustness, scalability, and flexibility to various attack patterns. The efficacy of XGBNN underscores the capability of hybrid architectures to tackle the increasing intricacy of cybersecurity issues in present-day computing settings. Despite achieving 100 % accuracy, this research is limited to the Windows OS and the dataset used. It needs to be extended to concentrate on expanding the model to more operating systems, integrating real-time detection functionalities, and enhancing computing efficiency for extensive deployment. XGBNN facilitates the establishment of more secure and resilient Windows environments against advancing cyber threats.

FUNDING INFORMATION

The author states no funding is involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Pranitha Kedambady Shiva	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
Pushparaj D. Shetty		✓				✓		✓		✓	✓	✓		

C : **C**onceptualization	I : **I**nvestigation	Vi : **Vi**sualization
M : **M**ethodology	R : **R**esources	Su : **Su**pervision
So : **So**ftware	D : **D**ata Curation	P : **P**roject administration
Va : **Va**lidation	O : Writing - **O**riginal Draft	Fu : **Fu**nding acquisition
Fo : **Fo**rmal analysis	E : Writing - Review & **E**diting	

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are openly available in https://research.unsw.edu.au/projects/toniot-datasets.

REFERENCES

[1] B Priyanka, "Enhanced CNN and SVM with adaptive modality switching and audio-based video summarization for real-time agricultural intrusion detection," *Journal of Information Systems Engineering and Management*, vol. 10, no. 33s, pp. 880-896, 2025, doi: 10.52783/jisem.v10i33s.5668.

[2] Z. Azam, M. M. Islam, and M. N. Huda, "Comparative analysis of intrusion detection systems and machine learning-based model analysis through decision tree," *IEEE Access*, vol. 11, pp. 80348–80391, 2023, doi: 10.1109/ACCESS.2023.3296444.

[3] R. A. Al Hasan and E. K. Hamza, "An improved intrusion detection system using machine learning with singular value decomposition and principal component analysis," *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 4, pp. 25-38, 2023, doi: 10.22266/ijies2023.0831.03.

[4] A. Alagarsamy, T. Elumalai, S. P. Ramesh, T. Karuppiah, P. Kaliyaperumal, and R. Perumal, "A hybrid framework for enhanced intrusion detection in cloud environments leveraging autoencoder," *International Journal of Informatics and Communication Technology*, vol. 14, no. 2, pp. 555-564, 2025, doi: 10.11591/ijict.v14i2.pp555-564.

[5] S. Kumar, S. Gupta, and S. Arora, "Research trends in network-based intrusion detection systems: a review," *IEEE Access*, vol. 9, pp. 157761-157779, 2021, doi: 10.1109/ACCESS.2021.3129775.

[6] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, 2021, doi: 10.1002/ett.4150.

[7] I. Martins, J. S. Resende, P. R. Sousa, S. Silva, L. Antunes, and J. Gama, "Host-based IDS: A review and open issues of an anomaly detection system in IoT," *Future Generation Computer Systems*, vol. 133, pp. 9-113, 2022, doi: 10.1016/j.future.2022.03.001.

[8] Z. T. Sworna, Z. Mousavi, and M. A. Babar, "NLP methods in host-based intrusion detection systems: A systematic review and future directions," *Journal of Network and Computer Applications*, vol. 220, 2023, doi: 10.1016/j.jnca.2023.103761.

[9] F. Khan and S. K. B. Shivabasappa, "Detecting evolving cyber threats in iot environments using machine learning," *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 6, pp. 12-23, 2024, doi: 10.22266/ijies2024.1231.02.

[10] W. T. Valavan and N. Joseph, "Intrusion detection system using K-means SMOTE algorithm with multi-dense layer bidirectional long short-term memory," *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 6, pp. 59-68, 2024, doi: 10.22266/ijies2024.1231.06.

[11] M. S. Dalvi, "Machine learning based intrusion detection system," *Journal of Information Systems Engineering and Management*, vol. 10, no. 36s, pp. 550–555, 2025, doi: 10.52783/jisem.v10i36s.6528.

[12] S. Singh, S. V. Fernandes, V. Padmanabha, and P. E. Rubini, "MCIDS-Multi classifier intrusion detection system for IoT cyber attack using deep learning algorithm," *Proceedings of the 3rd International Conference on Intelligent Communication Technologies and Virtual Mobile Networks, ICICV 2021*, pp. 354–360, 2021, doi: 10.1109/ICICV50876.2021.9388579.

[13] J. Figueiredo, C. Serrão, and A. M. de Almeida, "Deep learning model transposition for network intrusion detection systems," *Electronics (Switzerland)*, vol. 12, no. 2, 2023, doi: 10.3390/electronics12020293.

[14] N. W. Khan *et al.*, "A hybrid deep learning-based intrusion detection system for IoT networks," *Mathematical Biosciences and Engineering*, vol. 20, no. 8, pp. 13491–13520, 2023, doi: 10.3934/mbe.2023602.

[15] U. K. Lilhore *et al.*, "HIDM: Hybrid intrusion detection model for industry 4.0 networks using an optimized CNN-LSTM with transfer learning," *Sensors*, vol. 23, no. 18, 2023, doi: 10.3390/s23187856.

[16] J. B. Awotunde *et al.*, "An ensemble tree-based model for intrusion detection in industrial internet of things networks," *Applied Sciences (Switzerland)*, vol. 13, no. 4, 2023, doi: 10.3390/app13042479.

[17] S. Chalichalamala, N. Govindan, and R. Kasarapu, "Logistic regression ensemble classifier for intrusion detection system in internet of things," *Sensors*, vol. 23, no. 23, 2023, doi: 10.3390/s23239583.

[18] N. Moustafa, M. Keshk, E. Debie, and H. Janicke, "Federated TON_IoT windows datasets for evaluating AI-based security applications," *Proceedings - 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020*, pp. 848-855, 2020, doi: 10.1109/TrustCom50675.2020.00114.

[19] N. Moustafa, "TON_IoT datasets." https://research.unsw.edu.au/projects/toniot-datasets (accessed Apr. 27, 2025).

[20] S. Cherfi, A. Boulaiche, and A. Lemouari, "Enhancing IoT security: A deep learning approach with autoencoder-DNN intrusion detection model," *PAIS 2024 - Proceedings: 6th International Conference on Pattern Analysis and Intelligent Systems*, 2024, doi: 10.1109/PAIS62114.2024.10541183.

[21] W. Ding, M. Abdel-Basset, and R. Mohamed, "DeepAK-IoT: An effective deep learning model for cyberattack detection in IoT networks," *Information Sciences*, vol. 634, pp. 157-171, 2023, doi: 10.1016/j.ins.2023.03.052.

[22] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, 2020, doi: 10.1016/j.asoc.2019.105524.

[23] D. Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," *ACM SIGKDD Explorations Newsletter*, vol. 3, no. 1, pp. 27-32, 2001, doi: 10.1145/507533.507538.
[24] S. Mittal, "A survey on modeling and improving reliability of DNN algorithms and accelerators," *Journal of Systems Architecture*, vol. 104, 2020, doi: 10.1016/j.sysarc.2019.101689.
[25] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, 2021, doi: 10.1007/s42979-021-00592-x.
[26] S. A. Sushma, "Comparative study of naive Bayes, Gaussian naive Bayes classifier and decision tree algorithms for prediction of heart diseases," *International Journal for Research in Applied Science and Engineering Technology*, vol. 9, no. 3, pp. 475-486, 2021, doi: 10.22214/ijraset.2021.33228.
[27] S. Sperandei, "Understanding logistic regression analysis," *Biochemia Medica*, vol. 24, no. 1, pp. 12-18, 2014, doi: 10.11613/BM.2014.003.
[28] O. Kramer, "K-nearest neighbors," pp. 13–23, 2013, doi: 10.1007/978-3-642-38652-7_2.
[29] Y. Y. Song and Y. Lu, "Decision tree methods: applications for classification and prediction," *Shanghai Archives of Psychiatry*, vol. 27, no. 2, pp. 130-135, 2015, doi: 10.11919/j.issn.1002-0829.215044.
[30] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197-227, 2016, doi: 10.1007/s11749-016-0481-7.
[31] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-August-2016, pp. 785-794, 2016, doi: 10.1145/2939672.2939785.
[32] P. Israni, "Breast cancer diagnosis (BCD) model using machine learning," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 10, pp. 4456-4463, 2019, doi: 10.35940/ijitee.J9973.0881019.

BIOGRAPHIES OF AUTHORS

Pranitha Kedambady Shiva received his first degree from St Philomenas college, Puttur, Dakshina, Kannada, India. She has also Master degree from NMAMIT, University of Nitte, Udupi. She is currently a computer science researcher in Department of Mathematical and Computational Sciences, NITK Surathkal, India. Her main research interests focus on cyber security, artificial intelligence, cyber crime, deep neural networks, intrusion detection, cyber-attacks random forest data mining, and text mining. She can be contacted at email: pranithaks.217ma501@nitk.edu.in and Pranithaks12@gmail.com.

Dr. Pushparaj D. Shetty is working as a faculty at National Institute of Technology Karnataka (NITK) Surathkal since 2000. He is presently holding the position of Professor in department of Mathematical and Computational Sciences (MACS) at NITK Surathkal. He obtained his B.E in Computer Science and Engineering in 1999 from Mangalore University and M.E in Computer Science and Engineering in 2005 from Indian Institute of Engineering Science and Technology, Shibpur, Kolkata (formerly Bengal Engg. and Science University). He obtained his Ph.D from the Computer Science and Applications group, Department of Mathematics at the Indian Institute of Technology Delhi (July 2014). His research interests are in the area of Wireless sensor networks, Graph algorithms and their applications in Engineering, Cloud computing, and High-performance computing. He teaches Data Structures and Algorithms, Design and Analysis of Algorithms, Parallel programming etc. He has co-authored more than 45 research papers in peer reviewed journals and conference proceedings including a few book chapters. He has guided more than 50 postgraduate students for their project work. He has successfully guided 5 Ph.D. students and 4 MTech by research students. 2 students are pursing Ph.D. under his guidance. He is associated with IEEE Mangalore subsection since its inception. He is a senior member of IEEE, Member of IEEE Computer Society, senior member of ACM, and life member of Computer Society of India (CSI) and Institution of Engineers India (IEI). Dr. Pushparaj also volunteered as the Chair of IEEE Mangalore Subsection for the year 2021. He is additionally holding responsibility as Professor in-charge of Hostels, NITK Surathkal, which hosts about 6000 plus students. He can be contacted at email: prajshetty@nitk.edu.in and prajshetty@gmail.com.