Computers & Security 151 (2025) 104320

Contents lists available at ScienceDirect

Computers & Security

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

Stochastic gradient boosted distributed decision trees security approach for detecting cyber anomalies and classifying multiclass cyber-attacks

J C Sekhar [a,*], R Priyanka [b], Ashok Kumar Nanda [c], P Joel Josephson [d], M J D Ebinezer [e], T Kalavathi Devi [f]

[a] Department of Computer Science and Engineering, NRI Institute of Technology, Guntur, Andhra Pradesh, India
[b] Department of Networking and Communications, School of Computing, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur, Tamilnadu 603203, India
[c] Department of Computer Science and Engineering, B V Raju Institute of Technology, Narsapur, Telangana, India
[d] Department of Electronics and Communication Engineering, Malla Reddy Engineering College, Hyderabad, Telangana, India
[e] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India
[f] Department of Electronics and Instrumentation Engineering, Kongu Engineering College, Perundurai, India

ARTICLE INFO

Keywords:
Cybersecurity
Cyber-attack
Artificial intelligence
Machine learning
Cyber anomalies
Stochastic gradient boosted distributed decision trees
Honeybees mating optimisation

ABSTRACT

Identifying cyber anomalies and attacks in today's cybersecurity environment is essential. We can solve these difficulties by combining artificial intelligence (AL) and machine learning (ML) methods. The specifics of the existing security mechanisms and the supply quality define how effective ML-based security systems will be in strengthening such measures. Developing a security system to identify unusual activity and classify threats in the growing complexity and regularity of attacks is essential. This article provides a successful method to identify and classify cyber anomalies. We use a novel method in combination with Stochastic Gradient Boosted Distributed Decision Trees (SGB-DDT) with Honeybees Mating Optimisation (HBMO). To improve the detection accuracy, we use SGD-DDT, a distributed learning technique that is both highly scalable and effective by combining the collective wisdom of several decision trees. The SGB approach's adaptability and error-learning properties make the model less vulnerable to dynamic cyberattacks. The complications of classifying cyber-attacks into different types have prompted this research to propose an enhanced HBMO method. The HBMO method aims to improve model performance while reducing processing overhead, which takes inspiration from honeybee mating behaviour. This proposed method, SGB-DDT, can accurately identify several categories of cyberattacks using the enhanced HBMO method. We assess the proposed method using a large and varied dataset of cyberattack incidents from NSL-KDD and UNSW-NB15, encompassing common and uncommon attack types. The experiment results show that the SGB-DDT with higher HBMO outperforms traditional ML techniques.

1. Introduction

In recent years, the importance of cybersecurity services and defence against various attacks—such as malware, botnets, worms, denial-of-service (DoS), and unlawful access—has increased. Massive amounts of computer network aberrations resulted in significant losses and serious consequences (Sarker et al., 2020). An excellent illustration of this is the ransomware attack that occurred in May 2017 and resulted in $8 billion in damages to several industries, including education, healthcare, banking, and power (Qu et al., 2021). safeguarding Internet of Things (IoT) networks and cyber systems Security breaches and attacks toward Internet of Things (IoT) networks and cyber systems have risen. A smart system that can detect these changes or invasions is essential for finding a solution.

Even though many well-known tactics, such as firewalls and encryption, are created to block internet assaults, this is of the utmost importance. This study's foremost focus is the area of AI expertise. It emphasises the use of security models based on ML in particular. These models can become more effective by learning independently from security-related data.

Intelligent security services can be created using ML-based security models that assess numerous cyberattacks or irregularities and

* Corresponding author.
E-mail addresses: Jcsekhar78@outlook.com, jcsekhar9@nriit.ac.in (J.C. Sekhar), priyankr6@srmist.edu.in (R. Priyanka), ashokkumarnanda@yahoo.com (A.K. Nanda), joeljosephsonp@gmail.com (P.J. Josephson), ebinezer.mjd@gmail.com (M.J.D. Ebinezer), kalavathidevi@gmail.com (T.K. Devi).

https://doi.org/10.1016/j.cose.2025.104320
Received 7 March 2024; Received in revised form 10 August 2024; Accepted 6 January 2025
Available online 15 January 2025

eventually detect or foresee risks (Sarker et al., 2021). The term "multiclass" refers to a challenge involving various connected cyberattacks, and "binary class" refers to a situation where abnormalities must be detected. Botnet detection (Soe et al., 2020; Hasan et al., 2019), the assessment of attacks and anomalies in IoT sensors within IoT environments (Gauthama Raman et al., 2020), the classification of attacks for the development of intrusion detection systems, the identification of unusual network connections, differentiating between regular traffic and attacks, and other related topics have all been the focus of recent research. Although different ML techniques are used for various applications, their scope frequently includes investigating differences in the significance of security attributes or doing empirical assessments utilising a constrained set of security intelligence modelling tools. Numerous security solutions can use the abnormal conduct referred to as anomalies in the case of unknown attacks and the accompanying model, which differs from regular traffic (Sarker et al., 2020). Because of this, efficient, intelligent modelling in cybersecurity requires categorising connected attacks into various known subcategories, including but not limited to DoS, botnets, malware, worms, and the like. The classification of anomalies is crucial for spotting strange attacks among routine network traffic. Given the complexities described previously, the effectiveness of various ML models can vary depending on their capacity to extract valuable insights from security-related data. This variation results from the fact that a learning-driven security model's efficacy might change based on the significance of relevant security features and the inborn features of the data. Anomalies, well-known or unheard-of attack types, and a wide range of security features frequently appear in real-world cybersecurity difficulties. As a result, creating a reliable classification model and a streamlined feature selection process are commonly required for effective intrusion detection systems.

A network and computer security systems are the two critical components of a traditional cybersecurity framework. Despite developing numerous technologies to combat web-based attacks, such as firewalls and encryption, an intrusion detection system (IDS) is more effective at safeguarding the computer network from external threats. An IDS's primary role is thus to detect and prevent unauthorised computer systems and network activities. Traditional technology, such as firewalls, is unsuitable for the job. An intrusion detection system observes and evaluates a network or computer system's operations. Its goal is to recognise potential security flaws, including aggressive cyberattack tactics and denial-of-service (DoS) attacks. This system identifies, confirms, and deals with illegal activities within a system, such as intrusions, changes, or harm (Mohammadi et al., 2019). Differentiating between various cyberattack types and unexpected occurrences within a network is crucial for improving system security overall. It is also necessary to create a potent IDS, which is essential to contemporary network security.

GBDT have performed extraordinarily in various classification tasks among the many ML methods. However, because of the ever-increasing volume of data, traditional GBDT models may encounter difficulties processing the enormous amounts of information required in cybersecurity applications. This restriction necessitates the creation of novel, scalable, and effective methods. To address these problems, stochastic gradient boosted - distributed decision trees (SGB-DDT), a unique and potent solution, combine the benefits of distributed decision trees and stochastic gradient boosting. SG-BDDT enhances cyber-attack classification accuracy and has the scalability to handle large-scale datasets.

HMO is a nature-inspired optimisation method based on the mating process of honeybees, which is a highly ordered and efficient event. Honeybees utilise various strategies, including waggle dances, to share information about suitable mating places and maximise their collective decision-making process. The potential of this algorithm resides in its capacity to effectively search through complex solution spaces and identify optimal solutions by replicating honeybee swarms' cooperative decision-making and communication processes. Effective cyberattack modelling challenges are increased by several irrelevant features in modern security datasets with extensive security traits and dimensions. These additional qualities also pave the way for various concerns, such as higher variances that cause data overfitting in tree-based models that rely on single pathways, longer calculation and execution times for training models, and a lack of model generalisation. Consequently, there is reduced accuracy in predicting the rate of attack detection. One of the most precise metrics, the accuracy score, demonstrates how well the model produces accurate forecasts.

This paper introduces an innovative SGB-DDT method-based security approach for detecting anomalies and classifying multiclass cyberattacks. Decision trees have long been acknowledged as powerful tools for classification problems because of their capacity to manage complicated data structures and interpretability. Combining the competencies of Decision Trees with Stochastic Gradient Boosting suggests improved accuracy, scalability, and adaptability, making it compatible with the actual detection and discouragement of cyber-attacks. The security system processes enormous amounts of data quickly, utilising the combined computational power of networked nodes or devices, which is one significant advantage of using a distributed framework. By decentralising decision-making, the system can handle massive datasets, adapt to the ever-changing nature of cyberattacks, and boost accuracy and responsiveness.

The main contribution of the study:

- Develop a comprehensive framework for effective cyber anomaly detection and multiclass cyber-attack categorisation by combining the strength of SGB-DDT and a novel enhancement to the HBMO algorithm.
- The suggested architecture uses the collective wisdom of many decision trees to improve detection accuracy using SGB-DDT, a highly scalable and distributed learning technique.
- The stochastic gradient boosting technique enables the model to adjust and learn from mistakes, strengthening its resistance against dynamic cyber-attacks. Due to the complexity of multiclass cyber-attack classification, this work provides an improved HBMO method.
- The HBMO method efficiently optimised the selection of significant features and hyperparameters, decreased computational overhead, and resulted in increased model performance. It was inspired by the mating behaviour of honeybees.
- The enhanced HBMO methodology enables the SGB-DDT framework to precisely classify cyberattacks across multiple categories.
- The proposed technique is estimated on a large and diverse dataset of cyber-attack occurrences from UNSW-NB15 and NSL-KDD, including known attack patterns and unique anomalies.
- In standings of recall, precision, and F1-score for tasks involving anomaly detection and multiclass cyber-attack categorisation, the testing data shows that the upgraded SGB-DDT, driven by HBMO, outperforms standard machine learning algorithms.

Following the introduction section, Section 2 discusses the research background and present state of the IDS research; then, Section 3 delivers information about the proposed SGB-DDT and the tree topologies employed in MLTs for IDSs; Section 4 presents an analysis of the findings from experiments performed on the cybersecurity dataset, along with the performance assessment of the final security model; finally, Section 5 summarises the research work along with its anticipated future work.

2. Literature survey

Zhang et al. (2021) described the best data partitioning technique for any network under cyber threats. The data partition considers both inner and outside attacks (risk propagation). Under both limited and unlimited risk propagation, the likelihood of a data breach and the possibility of its recovery are explored for various situations. It is found that the optimal partition method can be significantly more affected by risk propagation than by external attacks and that unrestricted risk

propagation increases cyber risk. Given its significant influence on the partitioning technique, the network topology should always be considered. The corruption caused by compromise may call for different partitioning tactics.

Hasan et al. (2020) framed the interaction between the attacker and defence as a two-player game for power systems by accounting for dynamic cyber-attacks. Two efficient algorithms were developed to identify the worst-case dynamic attack, which maximises the loss of system load, and a defence plan, which minimises the loss. Xing (2020) provides a recent summary of cascading failure in reliability engineering. Cascading failure and the spread of cyber risk are related but not identical application domains. Cascading failure, on the other hand, essentially studies physical failure, whereas cyber risk propagation mainly examines the failure of the communication/network layer due to virus infection.

Sarker et al. (2021) thoroughly analyse the efficacy of several ML-centered security techniques using their CyberLearning concept. In this method, machine learning drives cybersecurity modelling, emphasising correlated feature selection. A binary classification model is also included in their study to help identify anomalies inside the CyberLearning framework. These components combine to produce a multiclass classification system for coding diverse cyber-attacks.

Zhang et al. (2019) proposed a method for detecting dangers in cyber-physical systems. Their study used classification models, including decision trees, bootstrap aggregations, random forests, and k-nearest neighbours. They included an auto-associative kernel regression model in their suggested research to speed up the detection of threats. Even though their advice was sound, the results fell below expectations due to technological difficulties.

Ibor et al. (2020) work conceptualised predicting cyberattacks as a classification problem. They also introduced a novel technique that combines a deep learning architecture with the addition of rectified linear units (ReLU) as the activation function within a deep feed-forward neural network's hidden layers. This method uses an iterative layer-wise learning technique to pull out critical characteristics from a dataset that includes both good and bad network traffic. Notably, the algorithm that underlies the model does dimensionality reduction, clustering, and early-stage feature selection, producing a set of input vectors known as hyper-features. The model's effectiveness is assessed in a Python environment using datasets from CICIDS2017 and UNSW_NB15.

Li et al. (2021) advocated studying and completely reviewing the standard advancements made in the field of cyber security and investigating the problems, benefits, and drawbacks of the offered methods. The new descendant assaults are thoroughly discussed. The history of early-generation cyber-security techniques and conventional security frameworks is discussed. Furthermore, recent developments in cyber security, security concerns, and threats are explored. The comprehensive review study supplied to IT and cyber security researchers is expected to be beneficial.

According to Furnell et al. (2020), a full explanation of a cyber-attack would significantly impact the legal system. This influence would guarantee continued awareness of and comprehension of the effects of these strikes. Sarker et al. (2021) emphasise that traditional national security concerns, which often have more visible characteristics and involve identifiable federal or regional players, differ dramatically from cyber threats. As a result of this transition, the traditional national security paradigm is becoming obsolete and ineffective.

Mishra (2022) focuses on developing novel Intrusion Detection Systems (IDSs) by utilising Genetic Algorithms (GAs) in combination with Optimised Gradient Boost Decision Trees (OGBDTs). The study employs Enhanced African Buffalo Optimisations (EABOs) and OGBDT-IDS for data exploration, preprocessing, standardisation, and feature evaluation tasks. The proposed methods significantly enhance the detection of cyber intrusions, effectively handling both new and previously unseen cases, and surpass traditional machine learning techniques in performance. The OGBDT-IDS notably demonstrates the highest attack detection rates while offering the fastest prediction times.

Kumar and Sabeeena (2023) introduce a robust method for detecting potential cyber-attacks utilising the Gradient Boosting Classifier, a machine learning algorithm renowned for its predictive solid capabilities. The approach incorporates advanced data preprocessing, careful feature engineering, and stringent model evaluation metrics. Its extensive applications cover sectors such as finance, healthcare, and e-commerce, where it serves as a critical defence against data breaches and unauthorised access.

Hassan et al. (2024) leverage machine learning (ML) techniques to analyse network traffic, detect anomalies, and classify activities, ultimately enhancing security and performance. The methodology incorporates logistic regression, decision trees, and ensemble learning techniques. The results demonstrate improvements in identifying network inefficiencies and classifying traffic, which help reduce delays and bottlenecks. ML models offer robust protection against cyber threats, increasing user satisfaction and bolstering organisational reputation. The research highlights the critical role of ML in network traffic analysis.

Alrefaei and Ilyas (2024) introduce a real-time intrusion detection system with a PySpark architecture to identify IoT attacks. The system leverages machine learning algorithms and utilises the IoT-23 dataset, incorporating data preprocessing and feature selection techniques. Results indicate that extreme gradient boosting achieves an accuracy of 98.39%, while the random forest algorithm surpasses current methods with a prediction time of just 0.0311 seconds.

Zhang et al. (2023) suggest a deep capsule convolution neural network-based method for automatically identifying and categorising different cyber-attacks. The convolution neural network extracts temporal characteristics from historical operation status in the delivered data packets and spatial correlations between various nodes. The proposed structure's capsules significantly preserve the measurement matrix's topological consistency. The recommended method avoids the requirement for models and lessens the impact of ambiguous system characteristics on detection effectiveness. The suggested method can reach a remarkable 99.97% accuracy in detecting single cyberattacks and a 96.25% accuracy in detecting multiple simultaneous cyberattacks, according to numerical findings from experiments on the IEEE 39-bus test system.

2.1. *Limitations of existing system*

- Any anomaly detection is susceptible to both false positives and false negatives. Accomplishing a balance between these two is difficult and can influence the system's success.
- Skilled attackers might devise attacks that are expressly tailored to escape detection systems. These adversarial attacks can exploit flaws in detection algorithms, rendering them ineffective.
- Implementing an effective cyber anomaly detection and categorisation system frequently necessitates large computational resources, such as memory and processing capacity. For firms with insufficient IT infrastructure, this may not be viable.
- The approach's scalability may be constrained, particularly when working with large and dynamic networks or data streams. Adapting the system to increased network traffic might be difficult.
- Anomaly detection systems may detect anomalies from regular activity but do not always provide context or information about the precise attack type or its possible impact.
- Making, implementing, and maintaining an efficient cyber anomaly detection and categorisation system can be costly, predominantly for smaller firms with limited properties.

2.2. Problem identification of existing system

- Cybercriminals are always devising new attack methods and strategies. Conventional security techniques typically fail to keep up with modern threats, leaving businesses vulnerable to new cyberattacks.
- Cyberattacks differ in goal, impact, and techniques. It is difficult to classify and categorise these multiclass attacks effectively. Existing approaches may lack the graininess required to discern between different types of cyber threats effectively.
- Anomaly detection is a severe cybersecurity constituent, yet gaining high accuracy while avoiding false positives remains a resolute struggle. A continuous experiment must establish a balance between sensitivity and specificity.
- The sheer volume and diversity of data generated by network and system activity might load security systems. Collecting, analysing, and deriving meaningful visions from this data is difficult, especially when dealing with large-scale networks and complicated environments.
- Real-time monitoring is critical for early threat detection and response in today's related environment. Developing systems that can continually pathway network traffic, system behaviour, and user actions in real-time is dangerous without cooperating performance.
- Many companies have limited resources and accomplished cybersecurity specialists. Developing cost-effective solutions that businesses with negligible resources can organise is challenging.

3. Proposed system

This section clarifies our security architecture, which uses ML to identify potential attacks and detect anomalies. To achieve this, several processing steps were required, including the analysis of the security dataset, the preparation of raw data, the development of an efficient cyber anomaly detection and multiclass cyber-attack classification that combines the strength of stochastic gradient boosted distributed decision trees (SGB-DDT) and a novel improvement to the Honeybees Mating Optimisation (HBMO) algorithm, as well as the development of a security model. The SGB-DDT method's block diagram is depicted in Fig. 1.

3.1. Exploring security dataset

Security datasets usually comprise various informative pieces covering numerous security-related topics and essential information. The construction of security models intended to detect anomalies is based on these databases. Identifying malicious activity or deviations requires a thorough understanding of unprocessed cybersecurity data's fundamental properties and insights into the patterns noticed inside security incidents (Sarker et al., 2020). Notably, the widely used UNSW-NB15 (Moustafa & Slay, 2015) and NSL-KDD (Tavallaee et al., 2009) security datasets were utilised to create and test the study's data-centric security model.

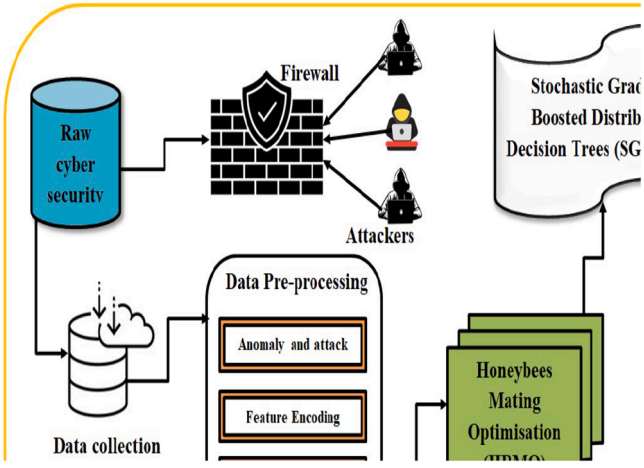

Fig. 1. Block diagram of SGB-DDT method.

Nine different attack types, including Fuzzers, Backdoors, DoS, Study, Exploits, Reconnaissance, Generic Attacks, Shellcode, and Worms, are covered within the dataset known as UNSW-NB1. It has 45 qualities and 257,673 occurrences for training and testing purposes. While analysing its subject matter, the NSL-KDD dataset emphasises Denial of Service (DoS), User-to-root (U2R), Remote-to-local (R2L), and Probing Attacks. From a pool of 494,020 instances in the raw data source, we have chosen 41 security measures for our empirical study. Remembering that a dataset may contain attributes from different categories is essential. The UNSW-NB15 dataset's security attributes are listed in Table 1 with a range of values. Because of this, a key component of our research is properly evaluating this data and developing a robust security model that can accurately identify the wide range of attacks and abnormalities indicated above.

3.2. Secure data preprocessing

Data preparation components that specialise in method adaptation to meet the demands of a given dataset include anomaly and attack detection, feature encoding, and scaling.

Attacks and anomalies: As mentioned, nine different attack types may be found in the UNSW-NB15 dataset. Our study's multiclass classification model includes each of these distinct attacks. A binary classification model is used to classify anomalies in our dataset, which are identified as such. This categorisation model covers four attack types: DoS, U2R, R2L, and probing. It is interesting to note that the NSL-KDD dataset also classifies these identical attack types as anomalies.

As shown in Table 1, the UNSW-NB1 dataset includes a variety of feature types, including nominal, integer, float, timestamp, and binary values. Our first step is vectorising any attributes with nominal values to align the data with the security model. Although "One Hot Encoding" is a common strategy, our study chooses "Label Encoding." The one-hot encoding strategy's significant increase in feature dimensions poses a considerable problem. The label encoding technique, in contrast, quickly converts feature values into accurate numerical representations, which makes it easier to build a machine learning classification model. Similarly, the attributes of the NSL-KDD dataset are encoded to generate the final security model.

Feature scaling: A critical phase in data preprocessing is feature scaling, often known as data normalisation. Notably, the security characteristics built into a dataset may differ, showing variations across different aspects concerning their data distribution patterns. The distribution of data for the two separate properties, sbyte, and snack, in the dataset UNSW-NB15. Consequently, we normalised the range of feature values to 0 with a standard deviation of 1 using the data scaling

Table 1
UNSW-NB15 dataset features.

Feature Name	Value Type	Feature Name	Value Type
ct_srv_src	Integer	sport	Integer
Stcpb	Integer	res_bdy_len	Integer
Swin	Integer	Dpkts	Integer
Spkts	Integer	ct_ ftp_cmd	Integer
trans_depth	Integer	dwin	Integer
Ackdat	Float	ct_srv_dst	Integer
Swin	Integer	dwin	Integer
is_ ftp_login	Binary	Dpkts	Integer
Smeansz	Integer	dmeansz	Integer
Sload	Float	Dload	Float
Spkts	Integer	is_sm_ips_ports	Binary
Srcip	Nominal	state	Nominal
Proto	Nominal	dtcpb	Integer

approach Standard Scaler.

Data splitting: Data splitting is crucial in our effort to develop learning-based security modelling. The rationale is that a bad data partitioning foundation could support a good security model. Our initial stage entails analysing the input data from diverse sources to develop an equitable model and assessment process. A k-fold cross-validation process is used to achieve this segmentation. When employing the k-fold cross-validation method, the previous section's input data is arbitrarily separated into k distinct, mutually exclusive "folds," represented by the letters d1, d2,..., dk. There are about equal numbers of data instances in each fold. The task must be finished by the model in k iterations. Except for di, we use the data instances from each fold as the training dataset when creating the security model in iteration i. I use di as the testing dataset for each iteration of the evaluation. The model's final output takes the average result into account.

3.3. Stochastic gradient boosting with distributed decision trees

To improve the cybersecurity system's overall performance, we propose combining Distributed Decision Trees (DDT) and Stochastic Gradient Boosting (SGB), a potent ensemble learning technique. While DDT facilitates parallel processing and scalability over distributed computer resources, SGB addresses overfitting and generalisation concerns (Ye et al., 2009).

3.3.1. Distributed training data

The parallelisation of our decision tree training method necessitates disseminating training data across numerous computers. Our focus is on techniques that efficiently partition the data among many devices instead of merely copying the data to conserve memory. Several distributed tree construction methods now scale distributed tree construction in terms of memory utilisation and training efficiency. Our research used vertically and horizontally partitioned strategies to distribute our data.

3.3.2. MapReduce implementation

This strategy tried to align the issue with the MapReduce paradigm's guiding principles. Each mapper would gather enough information to create a tree, and then by summing the various attribute-value pairs, each would determine the probable cut points. A thorough explanation of the mapper and reducer code used to find potential split locations is provided by Algorithm 1 in this section. The central component, designated as(e,v), is a combination of features f and v. This is accompanied by the equivalent value (k_i,w_i)created using the continuing residual and the sample i weight. These elements work together to create the thorough statistical framework underpinning the mapping procedure. For each key, weight and residual sums build up during the reduction stage. A single traversal through the arranged cutpoints leads to achieving the globally optimal cut using the generated output file.

The MapReduce technique speeds up choosing the best cutpoint for each feature by reducing the sample count to match the number of distinct sample values. This method scales incredibly well when applied to datasets with boolean or categorical attributes. After training a full tree, two more map operations are required: one to apply the current ensemble and the other to divide the data across the nodes. The pseudocode displayed in Algorithm 2 is used to update the residuals for each sample. The partitioner publishes each sample to a distinct output file depending on which side of the divide it falls. It is not included because creating the applier code in MapReduce is straightforward.

Only a few lines of code are needed for the MapReduce implementation. However, a sizable system overhead appears as we use HDFS for communication and create numerous files during node split. Hadoop's substantial communication load currently makes it unsuitable for algorithms of this kind. In the following parts, we focus on our MPI strategy because of the significant communication overhead.

3.3.3. Learning a distributed regression tree with MPI on hadoop

To improve communication, we take a different approach, substituting Hadoop streaming MPI for MapReduce as our second option. Since vertical partitioning reduces the communication costs associated with computing a tree node, we chose it for this implementation. Unless otherwise stated, we shall use vertically partitioned data for the remainder of the study. Load balancing was done to cut down on the amount of time spent waiting for stragglers.

3.3.3.1. MPI on hadoop. We changed OpenMPI to launch, utilising Hadoop streaming to take advantage of existing Hadoop clusters. Our method's main advantage is that we may leverage existing clusters instead of setting up a new MPI cluster, which saves time and money. The techniques needed to overcome technological obstacles include locating and getting in touch with the master node, starting jobs without SSH, and fault tolerance.

3.3.3.2. Finding the best split for a node. Among all possible splits for the feature$D_{i,j}$, where the feature$i \in E$ is a member of the set of all features (E), the best split for a node is the one that yields the most significant gain (d).

Each machine in the vertical partitioning context can access only the data required to establish precise local divisions $G'_{i,j}$at a particular segmentation point j regarding feature$i \in Es$. Every machine in the feature spaceE_L is dedicated to managing a specific sector.

$$G'_{i,j} = \text{argmax}_{i,j}\{gain(D_{i,j})\} \tag{1}$$

Each machine in the network selects the most advantageous benefit from its collection of features. The remaining machines are then informed of this discovery via an MPI broadcast. When considering the averaged local splits acquired from each machine, the split $G^*_{i,j}$associated with the dividing point$d'_{i,j}$ that delivers the maximum gain is evaluated to determine the overall optimum split$G'_{i,j}$.

Algorithm 1
Aggregating candidate splits.

```
map(key, value):
  E ⇐ set of features
  sample ⇐ split(value, delim)
  for e in E do key = (e, sample[e])
  value = (sample[residual], sample[weight])
  emit(key, value)
  end for
  reduce(key, values):
  residual_sum ⇐ 0
  weight_sum ⇐ 0
  for v in values do
  residual_sum ⇐ residual_sum + v.residual
  weight_sum ⇐ weight_sum + v.weight
  end for emit(key, (residual_sum, weight_sum))
```

Algorithm 2
Partitioning a Node n.

```
map(key, value):
  sample ⇐ split(value, delim)
  if sample[n.feature] < n.splitpoint then
  residual = sample[residual] + n.left_response
  else
  residual = sample[residual] + n.right_response
  end if
  emit(key, value)
```

$$G_{i,j}^{*} = \text{argmax}_{i,j}\left\{gain\left(d_{i,j}^{'}\right)\right\} \tag{2}$$

Every machine knows the optimum global division because the division characteristic has been stored in memory. Each machine then divides the samples, after which it delivers the precisely updated post-split indices. After that, each machine is waiting.

3.3.3.3. Partitioning the data. After learning a split point, data are divided into two subgroups for classic decision tree construction. When using distributed partitioning, only one computer has the feature to partition the dataset in memory. Because of this, the system can only partition the data with the best split, which then updates the other workstations.

The training data for each stochastic GBDT tree is initially selected randomly. Since the gradient of the loss function serves as the target for each sample in the next tree, we need the current score for each sample in the training set while boosting. This presents a complicated issue in the distributed instance since we must apply the present ensemble on samples with features dispersed across numerous workstations. We adjust our partition technique to consider all samples in the training set while maintaining the incremental score index to fix the problem.

$$score_n(y) = score_{n-1}(y) + \sum_{j=1}^{J_n} \delta_{jn} I\left(y \in R_{jn}\right) \tag{3}$$

When the linked region's response $\delta_{jn}$is in the regionR_{jn}, the indicator function, I, and the sample score from training the prior$n-1$ trees all affect the score at tree n. It is feasible to gradually update the score index as new nodes are trained so that the solution matches the residual mean of the samples found in the inner nodes of the tree. The sample rate is negatively correlated with each tree's additional overhead.

3.3.3.4. Finding the best node for splitting. A similar technique that separates the leaf node with the most significant gain from the others is used to create greedy trees. Although there is no guarantee that this will result in the best tree, it is an effective linear process and uses the same strategy as our non-distributed variant.

Other tree-growing techniques, like growing by level, can be implemented with more parallelism, but we choose to employ the greedy approach to produce identical trees.

3.3.4. Stochastic gradient boosting with distributed decision trees

The structure of the additive regression model known as gradient-boosted decision trees consists of many weak decision tree learners.

$$b_o(y) = \sum_{i=1}^{o} \delta_i b_i(y) \tag{4}$$

where $\delta_i$stands for the pace of learning. In stochastic gradient boosting, a subsample of the training data is used to train each weak learner$b_i(y)$.

The gradient of the loss function$L(x_i,\ s_i)$, with the parameters $x_i$typically being the true label of the sample and the score s_i typically being$s_i = B_{o-1}\left(y_i\right)$, is the typical target$t_i$ of a sample $y_i$for the k-th tree. The loss function could be in the form of least squares or a more specialist function.

A special issue arises when distributed decision trees are trained using data spread over numerous nodes. We must keep track of the sample results while training or collecting feature values from distant devices. This is required since the ensemble's training data ensemble's loss function depends on the scores of each sample. Our solution uses the latter to reduce machine-to-machine communication. Consequently, we change our applier function during training to be

$$B_o(y) = s_o(y) \tag{5}$$

where $s_o(y) = 0$and $s_o(y)$is the index specified in Section 3.3.4.3. The next step is boosting, as demonstrated below:

1. TS$_o$he training data will be randomly sampled with replacement
2. to obtain the subset.
3. Ensure that the goal $k_i$of examples iS_os set to $k_i = L\left(x_i, s_o\left(y_i\right)\right)$where$x_i$ the sample's true label is.
4. With the samples$\left(y_i, k_i\right)$, where$y_i \in S_o$, train the k-th tree,$b_o(y)$

3.4. Honeybees mating optimisation (HBMO)

A few thousand workers, a few hundred drones, and one or more long-living queens make up a typical honeybee colony. The 'royal jelly' is only given to the queen bee. Due to the queen bee's unique diet, she is more significant than any other bee in the colony. The drones are in charge of giving sperm to the queen. A drone's demise occurs after mating. Worker bees are exceptionally skilled at taking care of broods. Eggs can be fertilised or not to form a brood. The former group may produce future queens or workers, whilst the latter may produce drones. The queen(s) mate throughout their flight away from the nest. During this trip, the drones accompany the queen and follow her; a select number are allowed to mate in midair. The spermatheca gathers sperm after each mating, creating the colony's genetic storage facility. When she lays fertilised eggs, the queen occasionally chooses a blend of sperm from this pool to fertilise an egg. When the queen's energy reaches a certain level, her flight is over, and she returns to the nest (Maheri et al., 2017).

Each worker can be considered a heuristic function in charge of maintaining and improving a group of broods while constructing the optimisation strategy. As the research continues, an annealing function can be used to represent the likelihood of a drone mating with a queen.

$$\text{Pro}b(Z, M) = e^{-\frac{\Delta(f)}{S(t)}} \tag{6}$$

In this case, S(t) denotes the queen's velocity at a specific time t. The symbol represents the absolute fitness gap between the drone M and the queen Z$\Delta(f)$. The possibility of drone M successfully mating with queen Z is also indicated$\text{Pro}b(Z,M)$. Notably, the queen mates more quickly and is more likely to do so when the drone is as fit as the queen or when she is just starting her mating flight. In contrast to Simulated Annealing, Eq. (6) solely relies on the queen's speed, energy, and drone energy to estimate mating chance during each trip. Simulated Annealing takes into account comparative threshold considerations. The energy and speed of the queen randomly decreasing upholds the method's stochastic core. The speed and energy of the queen decrease with each shift in spatial direction, as shown by the following equations:

$$S(t+1) = \alpha \times S(t) \tag{7}$$

$$E(t+1) = E(t) - \chi \tag{8}$$

E(t) is the energy lost between 0 and 1 after each transitionα. The valueχ is given to the algorithm. In this study, χit is purposefully set at 0.01 for all occurrences, and the energy content of the queen during each trip varies at random within the range [0, 1]. Since the energy content determines when to stop each mating excursion, the valueχ substantially impacts how EHBMO (Extended Honey Bee Mating Optimisation) converges. Smaller valuesχ could cause the convergence to be slowed down, while larger values could cause the flights to end with an incomplete spermathecal.

Individual queens' spermatheca capacity, energy reserves, and genetic makeup can vary. Because of this, the queen's energy and speed are randomly initialised before each mating flight. A genotype and a genetic marker serve as a representation of a drone in the algorithm. All drones are haploid by nature. Therefore, half of the genes can be randomly marked with a genotype marker while the other remains unmarked. Only the unmarked genes, in this instance, produce sperm that is used randomly during mating.

The drones are chosen depending on how well they mate with the queen during each cross-over flight. These drones are a collective of changing structures that strive for optimal configuration. These specially chosen drones are kept and used to produce and enhance broods. The number of drones, or "Max Sperm," refers to the extreme amount of trial constructions that can be kept in the spermathecal.

To improve the genetic makeup of the progeny, the workers—representing a variety of heuristics—work. Overall fitness is influenced by the genetic makeup that determines how quickly the kids will improve. The drone's genes are reproduced into the genetic composition of the progeny during development, while the queen's genes account for the remaining genes. The fitness of the resulting genetic composition is assessed by assessing the value of the objective function and its normalised equivalent within the genetic makeup of the young ones. A brood only has one genotype; it should be acknowledged. The following are the main steps of the HBMO algorithm.

1) Following the creation of the initial population at random, the leading queen, which represents the ideal member, starts her mating flight. Throughout the flight, she compiles the spermatheca (a list of drones) by probabilistically choosing the drones. The next step is to randomly select a drone from the list to start a brood.
2) New broods (test solutions) are produced by fusing the drone and the queen's genotypes.
3) Broods (trial solutions) are searched locally using workers (heuristics).
4) The degree of improvement made on the brood is used to gauge the fitness of the workers.
5) The stronger broods take the place of the weaker queens.

One structural analysis is performed each time the fitness of a drone, brood, or queen is evaluated. The number of structural inspections in each flight varies depending on the number of bees counted, such as the number of drones travelling to the queen and broods.

3.5. Advantages of the proposed method

- HBMO, in combination with Stochastic Gradient-Boosted Distributed Decision Trees, enhances the overall accuracy of cyber anomaly detection and cyber-attack classification, resulting in more dependable security measures.
- This technique can adequately classify multiclass cyber-attacks, providing for a thorough awareness of varied threats, which is critical in current cybersecurity.
- The system becomes more resistant to false positives and negatives, lowering the odds of overlooking serious threats or triggering unneeded warnings.
- The stochastic gradient boosting method expedites potential cyber-attack responses by increasing detection efficiency.
- The system is well-suited to dynamic and ever-changing cybersecurity landscapes because HBMO and Stochastic Gradient Boosted Distributed Decision Trees can adapt to developing cyber threats and anomalies

4. Result and discussion

This result section discusses classifying incursions into distinct groups and identifies cyber anomalies, then describes the evaluation metrics, the environmental setup, and the final result of the system-to-security system

4.1. Experimental setup

The proposed method is evaluated through an experiment using a Windows 10 computer with 16 GB of RAM and an Intel Core i5–10210U CPU. The experiment uses the datasets UNSW-NB15 and NSL-KDD for testing purposes. A variety of significant libraries, including matplotlib (version 3.3.2), NumPy (version 1.19.2), pandas (version 1.1.3), sci-kit-learn (version 0.23.2), Keras (version 2.6.0), and TensorFlow (version 2.6.0), are utilised with Spyder Python, notably version 3.8.

4.2. Evaluation Metrics

As shown in the following image, the efficiency of our security technique in classifying multiclass intrusions and spotting cyber abnormalities is determined by computing the final results using metrics including precision, recall, f-score, accuracy, and execution time.

The accuracy score's precision outweighs the value of the performance evaluation criteria listed above: true negatives, false positives, true positives, and false negatives must be calculated.

- TP (true positive): A situation where the security model successfully categorises or recognises advantageous threats or abnormalities.
- TN (true negative): A circumstance in which the security model successfully detects or categorises threats or anomalies as unfavourable.
- FP (false positive): A situation where the security model incorrectly recognises or classifies a particular group of threats or abnormalities.
- FN (false negative): A case in which the security model incorrectly identifies or fails to detect a negative class of threats or abnormalities.

4.2.1. Precision analysis

The classification precision (detection rate) is defined as the percentage of relevant instances (e.g., attacks) among the retrieved instances.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

In Fig. 2 and Table 2, the SGD-DDT technique's precision is contrasted with other widely used approaches. The graph shows how the ML approach performs better in terms of precision. In comparison to the SGD-DDT model, which has a precision of 93.425% for 100 data, the Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM, and RNN models have precision values of 89.213%, 87.314%, 85.314%, 91.213%, 85.132% and 90.023% respectively. The proposed SGD-DDT model has a precision of 95.692% under 500 data compared to the models for Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM and RNN,

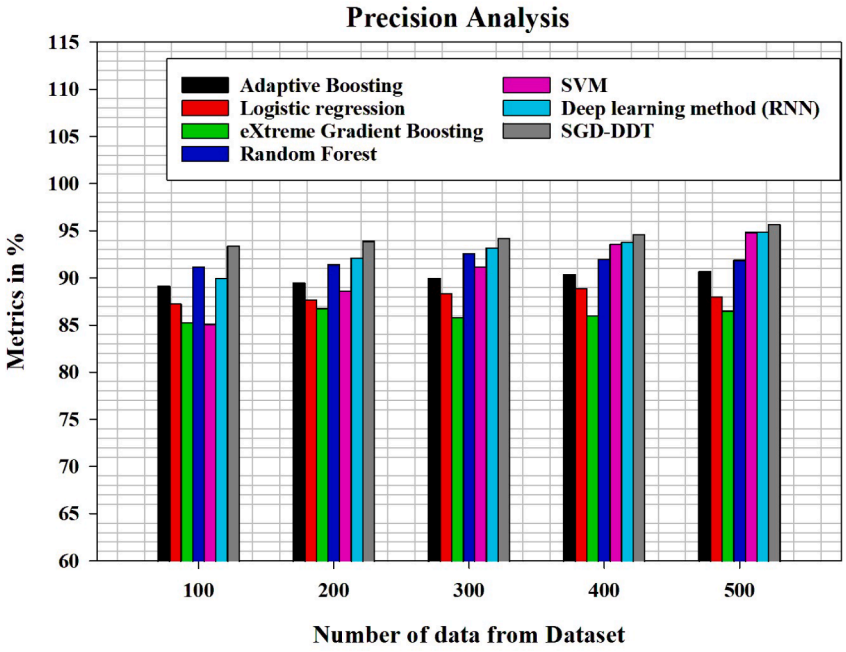

Fig. 2. Precision Analysis for SGD-DDT method.

Table 2
Precision analysis for SGD-DDT method.

Number of data	Adaptive Boosting	Logistic regression	eXtreme Gradient Boosting	Random Forest	SVM	Deep learning method (RNN)	SGD-DDT
100	89.213	87.314	85.314	91.213	85.13	90.023	93.425
200	89.526	87.728	86.819	91.454	88.65	92.18	93.928
300	90.029	88.415	85.839	92.637	91.23	93.24	94.225
400	90.426	88.917	86.063	92.038	93.64	93.82	94.627
500	90.728	88.043	86.552	91.926	94.87	94.93	95.692

which have precision values of 90.728%, 88.043%, 86.552%, 91.926%, 94.87% and 94.93%.

4.2.2. Recall analysis

The proportion of positive examples that are correctly labelled is known as the classification recall (sensitivity);

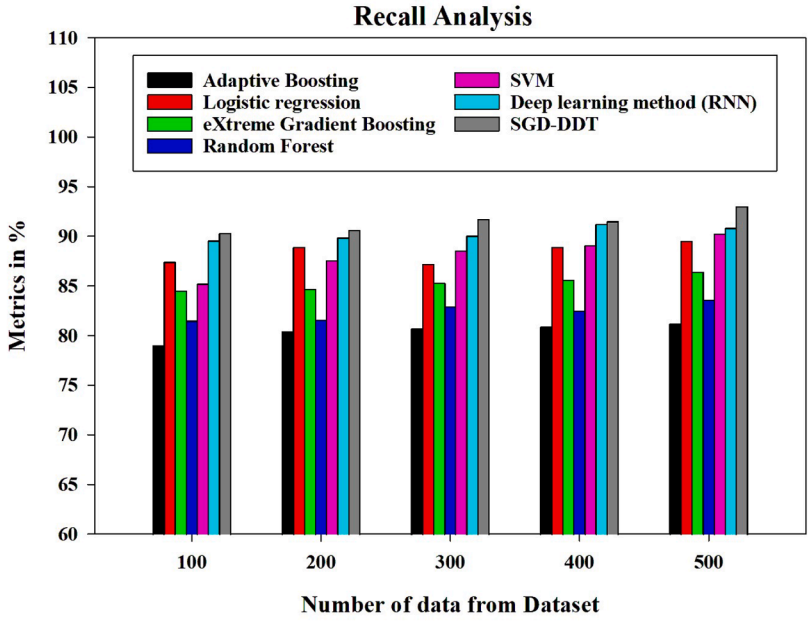

Fig. 3. Recall Analysis for SGD-DDT method with existing systems.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

In Fig. 3 and Table 3, the SGD-DDT technique's recall is contrasted with other widely used approaches. The graph illustrates how the ML approach enhances recall performance. Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM and RNN, for instance, have recall values of 79.017%, 87.435%, 84.526%, 81.526%, 85.234% and 89.567% respectively. In contrast, the SGD-DDT model has a recall value of 90.324% for 100 data. The proposed SGD-DDT model has a recall value of 93.028% under 500 data, as opposed to recall values of 81.242%, 89.536%, 86.435%, 83.628%, 89.567% and 90.856% for the Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM and RNN, respectively.

4.2.3. F-score analysis

The mean score that incorporates recall and precision (i.e., false positives and false negatives are utilised) is known as the F-score.

$$F - Score = 2((\text{Precision} * \text{Recall}) / \text{Precision} + \text{Recall})) \tag{11}$$

In Fig. 4 and Table 4, the SGD-DDT technique's f-score contrasts with other widely used approaches. The graph shows how the ML approach performs better with the f-score. For instance, the Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM and RNN models have f-scores of 80.627%, 81.928%, 83.928%, 84.627%, 85.736% and 86.003% respectively. In contrast, the SGD-DDT model has an f-score of 87.918% for 100 data. The proposed SGD-DDT model has an f-score value of 89.516% under 500 data, compared to the Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM and RNN models, with f-score values of 81.324%, 84.563%, 82.536%, 86.314%, 86.897% and 87.753%, respectively.

4.2.4. Accuracy Analysis

Accuracy is determined by the ratio of correctly classified records to all records or counts, as stated in Eq. (12).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{12}$$

In Fig. 5 and Table 5, the accuracy of the SGD-DDT technique is contrasted with that of other widely used approaches. The graph shows how the ML approach performs more accurately and efficiently. Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM and RNN models, for instance, have accuracies of 91.029%, 93.029%, 94.910%, 96.314%, 96.879% and 97.005% for 100 data, however the SGD-DDT model has a value of 97.415%. The proposed SGD-DDT model has an accuracy of 98.718% under 500 data, compared to the accuracies of 91.902%, 94.028%, 95.829%, 96.615%, 97.164% and 98.567% for the Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM and RNN models, respectively.

4.2.5. False alarm rate analysis (FAR)

The false alarm rate refers to the number of incorrectly labelled routine occurrences the system detects. The following formulas can be used to compute these measures.

$$False\ Alarm\ Rate = \frac{FP}{TN + FP} * 100\% \tag{13}$$

Fig. 6 and Table 6 display a FAR comparison of the SGD-DDT strategy with other well-known approaches. The ML technique has an enhanced performance with reduced FAR, as shown in the graph. For instance, the FAR value for the SGD-DDT model is 21.029% for 100 data, compared to the FAR values of 33.029%, 30.213%, 27.342%, 24.782%, 24.678% and 23.879% for the Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM and RNN models, respectively. The SGD-DDT model performs effectively on various data sets with low FAR values. Similarly, the FAR with 500 data for the SGD-DDT is 23.256%. Still, for the Adaptive Boosting, eXtreme Gradient Boosting, Logistic Regression, Random Forest, SVM and RNN models, it is 36.829%, 32.938%, 29.782%, 26.452%, 25.546% and 22.347%, respectively.

4.2.6. Classification Time Analysis

The duration it takes for a system or algorithm to categorise and identify potential cybersecurity threats or attacks into different predefined classes or categories based on their characteristics and patterns is called classification time in the security approach to detect cyber anomalies and multiclass cyber-attack classification. This procedure enables security experts to respond rapidly to possible dangers and effectively defend against cyber-attacks (Table 7).

Fig. 7

The proposed SGB-DDT method showed a notable enhancement in accuracy and performance measures when compared to conventional and deep learning methods, with an accuracy rate of 98.718%, precision of 95.692%, recall of 93.028%, and an F-score of 89.516%. These measures indicate that the SGB-DDT approach successfully identifies cyber irregularities and categorises different forms of cyber assaults. The comparison demonstrates that it surpasses classical machine learning techniques, which usually attain accuracies between 85% and 95%. In contrast, deep learning methods can achieve similar results but usually demand greater computational resources and tuning. The SGB-DDT method better balances accuracy and resource efficiency, making it particularly beneficial for organisations with minimal IT infrastructure. The technique, designed for efficient processing of large amounts of data, uses distributed computing to improve scalability, an essential factor in the ever-expanding field of cybersecurity with increasing data size and complexity. Its structure enables fast data processing for real-time monitoring and threat detection. Moreover, incorporating the Honeybees Mating Optimisation (HBMO) algorithm improves the method's resilience to adversarial attacks, an essential characteristic in an environment where attackers constantly change their tactics to avoid being detected. The SGB-DDT method also performs better in real-time detection abilities, which are crucial for immediate threat reaction, enabling constant surveillance of network traffic and system activity to mitigate the chances of substantial harm.

The Stochastic Gradient Boosting (SGB) algorithm and the Distributed Decision Trees (DDT) framework impact the SGB-DDT method's computational complexity. The SGB complexity is approximately O(T * N * D), where T is the number of trees, N is the number of training samples, and D is each tree's depth. The DDT framework improves efficiency by using parallel processing, which reduces overall computational time. As a result, the overall complexity is O(T * N * D), with actual performance varying depending on the implementation, dataset size, and computational resources available. The distributed nature of DDT effectively reduces computational burdens, making the SGB-DDT

Table 3
Recall analysis for SGD-DDT method.

Number of data	Adaptive Boosting	Logistic regression	eXtreme Gradient Boosting	Random Forest	SVM	Deep learning method (RNN)	SGD-DDT
100	79.017	87.435	84.526	81.526	85.234	89.567	90.324
200	80.425	88.920	84.738	81.627	87.564	89.876	90.627
300	80.728	87.231	85.324	82.938	88.567	90.045	91.728
400	80.926	88.926	85.627	82.536	89.123	91.234	91.526
500	81.242	89.536	86.435	83.628	90.258	90.856	93.028

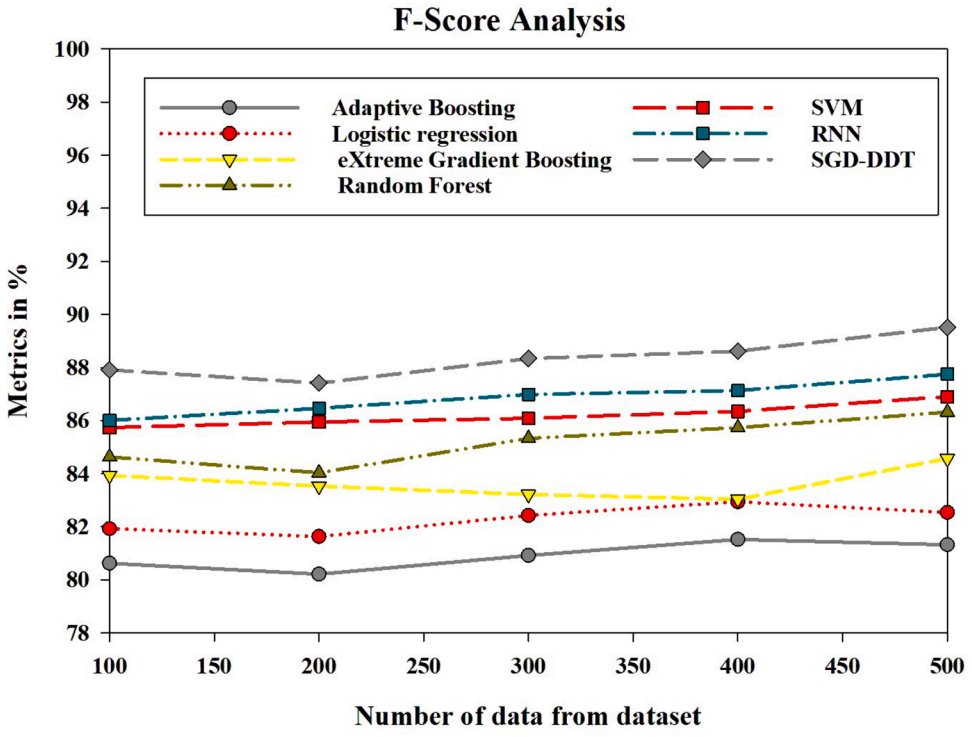

Fig. 4. F-Score Analysis for SGD-DDT method.

Table 4
F-score analysis for SGD-DDT method.

Number of data	Adaptive Boosting	Logistic regression	eXtreme Gradient Boosting	Random Forest	SVM	Deep learning method (RNN)	SGD-DDT
100	80.627	81.928	83.928	84.627	85.736	86.003	87.918
200	80.213	81.627	83.526	84.029	85.943	86.456	87.415
300	80.917	82.425	83.213	85.324	86.078	86.978	88.341
400	81.526	82.938	83.029	85.728	86.347	87.123	88.617
500	81.324	82.536	84.563	86.314	86.897	87.753	89.516

method appropriate for large-scale cybersecurity applications.

5. Conclusion

This work introduces the idea of "CyberLearning," which includes a multiclass classification model intended to recognise various kinds of cyberattacks and a binary classification model for identifying anomalies. Our initial focus was on investigating the impact of several security features to construct a robust machine learning-driven model employing a careful feature selection process. We created these thorough security models by fusing famous machine learning classification approaches with deep learning methodologies, notably using artificial neural networks. After that, we thoroughly assessed the performance of these learning-centric security models. In this result analysis, we performed tests with data from the security datasets. We strongly believe that our experimental research and analysis findings will be helpful in theoretical and specialised settings. These discoveries can be used to build data-oriented solid security models and systems supported by ML approaches. With the following testing results: precision of 95.692%, recall of 93.028%, f-score of 89.516%, accuracy of 98.718%, False alarm rate Analysis of 23.256%, and Classification time 1.526ms, the recommended framework SGB-DDT demonstrates impressive performance. Future studies will apply the model to anticipate the different sorts of cyber-attacks and assess their effectiveness with other security attribute variables. Deep learning methods for supervised and semi-supervised machine learning responsibilities will be addressed to boost classification accuracy and shorten the training and testing phases for cyber-attack classification models.

Code availability

Not applicable.

Funding

The authors did not receive any funding.

CRediT authorship contribution statement

J C Sekhar: Conceptualization, Writing – review & editing. **R Priyanka:** Writing – review & editing. **Ashok Kumar Nanda:** Methodology, Writing – original draft, Writing – review & editing. **P Joel Josephson:** Investigation, Methodology. **M J D Ebinezer:** Supervision, Validation. **T Kalavathi Devi:** Visualization, Writing – review & editing.

Declaration of competing interest

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

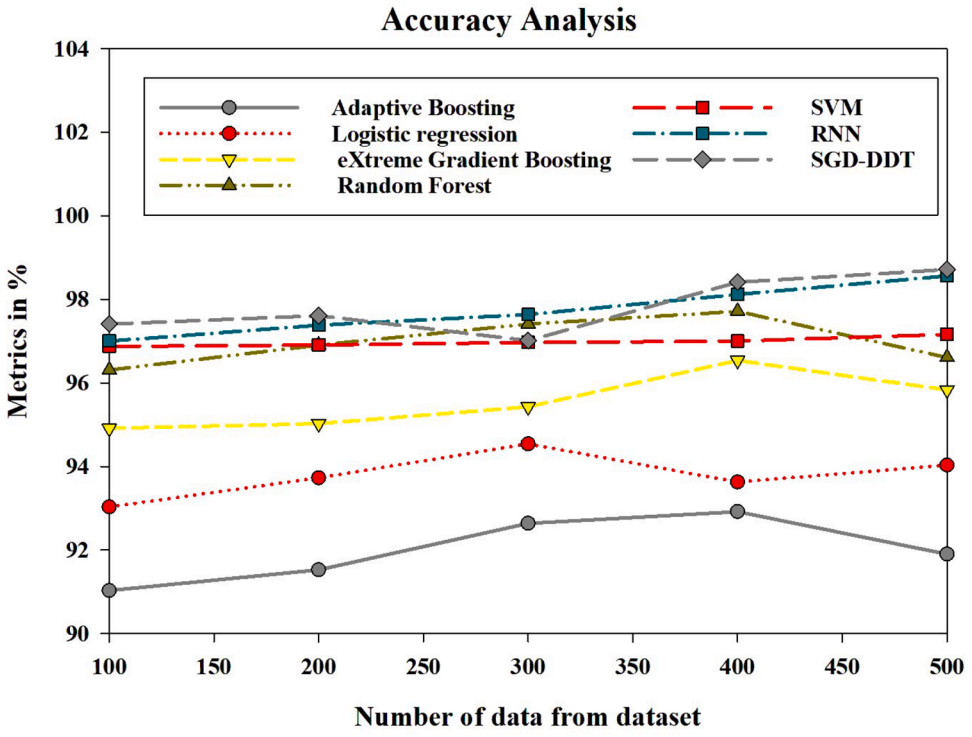

Fig. 5. Accuracy analysis for SGD-DDT technique.

Table 5
Accuracy analysis for SGD-DDT method.

Number of data	Adaptive Boosting	Logistic regression	eXtreme Gradient Boosting	Random Forest	SVM	Deep learning method (RNN)	SGD-DDT
100	91.029	93.029	94.910	96.314	96.879	97.005	97.415
200	91.526	93.728	95.021	96.911	96.912	97.385	97.615
300	92.638	94.536	95.426	97.415	96.976	97.643	97.017
400	92.918	93.626	96.542	97.718	97.005	98.124	98.415
500	91.902	94.028	95.829	96.615	97.164	98.567	98.718

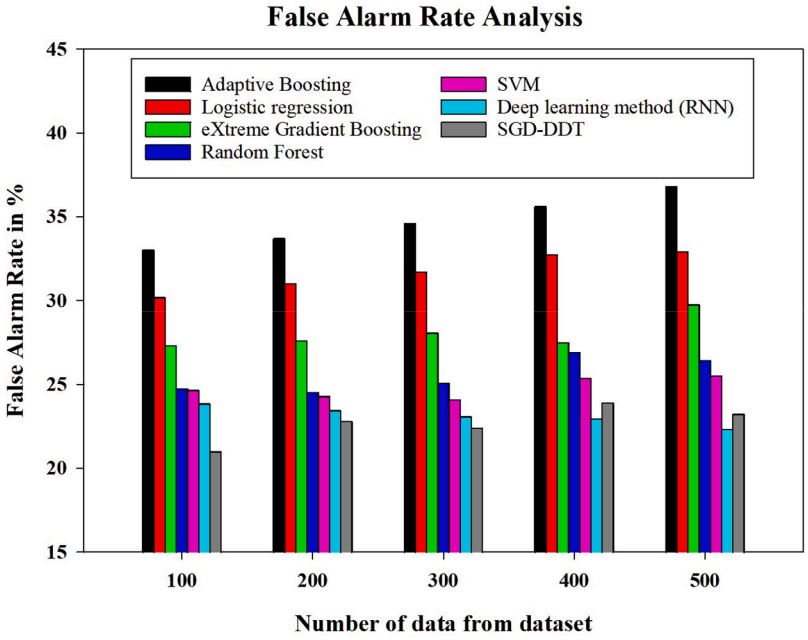

Fig. 6. False alarm rate Analysis for SGD-DDT method.

Table 6
False alarm rate analysis for SGD-DDT technique.

Number of data	Adaptive Boosting	Logistic regression	eXtreme Gradient Boosting	Random Forest	SVM	Deep learning method (RNN)	SGD-DDT
100	33.029	30.213	27.342	24.782	24.678	23.879	21.029
200	33.728	31.029	27.627	24.562	24.324	23.473	22.829
300	34.627	31.728	28.092	25.113	24.113	23.105	22.425
400	35.628	32.754	27.526	26.930	25.397	22.987	23.928
500	36.829	32.938	29.782	26.452	25.546	22.347	23.256

Table 7
Classification time analysis for SGD-DDT technique.

Number of data	Adaptive Boosting	Logistic regression	eXtreme Gradient Boosting	Random Forest	SVM	Deep learning method (RNN)	SGD-DDT
100	6.062	4.982	3.029	2.102	2.005	2.156	1.029
200	6.314	4.213	3.526	2.425	2.348	2.345	1.425
300	6.526	5.029	3.728	2.827	2.697	2.763	1.298
400	6.728	5.342	4.029	2.726	2.546	2.873	1.928
500	6.073	5.827	4.627	3.627	2.982	1.997	1.526

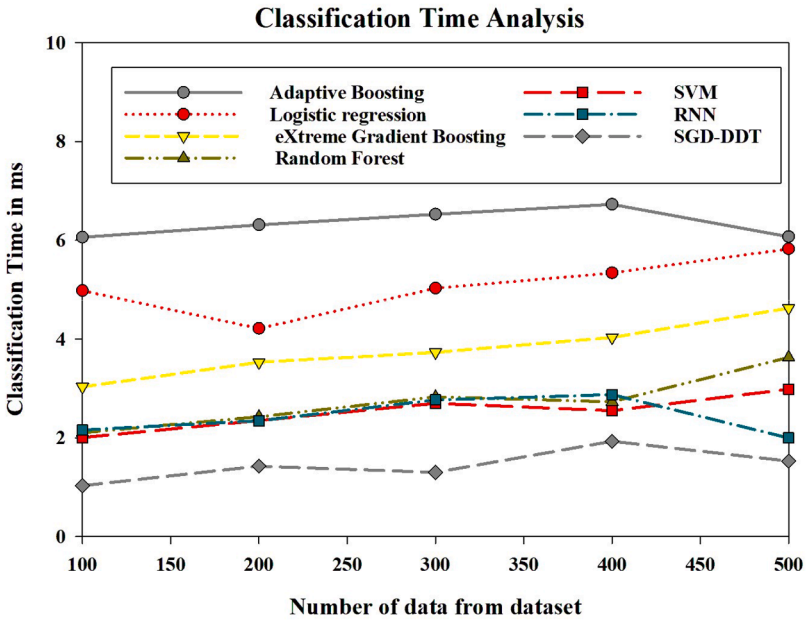

Fig. 7. Classification time analysis for SGD-DDT method.

Data availability

No data was used for the research described in the article.

References

Alrefaei, A., Ilyas, M., 2024. Using machine learning multiclass classification technique to detect IoT attacks in real time. Sensors 24 (14), 4516. https://doi.org/10.3390/s24144516.

Furnell, S., Heyburn, H., Whitehead, A., Shah, J.N., 2020. Understanding the full cost of cyber security breaches. Comp. Fraud Secur. 2020 (12), 6–12. https://doi.org/10.1016/S1361-3723(20)30127-5.

Gauthama Raman, M.R., Somu, N., Jagarapu, S., Manghnani, T., Selvam, T., Krithivasan, K., Shankar Sriram, V.S., 2020. An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm. Artifi. Intell. Rev. 53, 3255–3286. https://doi.org/10.1007/s10462-019-09762-z.

Hasan, M., Islam, M.M., Zarif, M.I.I., Hashem, M.M.A, 2019. Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. Int. Things 7, 100059. https://doi.org/10.1016/j.iot.2019.100059.

Hasan, S., Dubey, A., Karsai, G., Koutsoukos, X., 2020. A game-theoretic approach for power systems defence against dynamic cyber-attacks. Int. J. Electrical Power Energy Syst. 115, 105432. https://doi.org/10.1016/j.ijepes.2019.105432.

Hassan, S.E.H., Duong-Trung, N., 2024. Machine learning in cybersecurity: Advanced detection and classification techniques for network traffic environments. EAI Endorsed Transac. Indust. Network. Intell. Syst. 11 (3). https://doi.org/10.4108/eetinis.v11i3.5237.

Ibor, A.E., Oladeji, F.A., Okunoye, O.B., Ekabua, O.O., 2020. Conceptualisation of cyberattack prediction with deep learning. Cybersecurity 3, 1–14. https://doi.org/10.1186/s42400-020-00053-7.

Kumar, N.C., & Sabeena, J. (2023). Cybersecurity attack detection using gradient boosting classifier. doi:10.21203/rs.3.rs-3711213/v1.

Li, Y., Liu, Q., 2021. A comprehensive review of cyber-attacks and cyber security; Emerging trends and recent developments. Energy Reports 7, 8176–8186. https://doi.org/10.1016/j.egyr.2021.08.126.

Maheri, M.R., Shokrian, H., Narimani, M.M., 2017. An enhanced honey bee mating optimisation algorithm for the design of side sway steel frames. Adv. Eng. Softw. 109, 62–72. https://doi.org/10.1016/j.advengsoft.2017.03.006.

Mishra, S., 2022. An optimised gradient boost decision tree using enhanced African buffalo optimisation method for cyber security intrusion detection. Appl. Sci. 12 (24), 12591. https://doi.org/10.3390/app122412591.

Mohammadi, S., Mirvaziri, H., Ghazizadeh-Ahsaee, M., Karimipour, H., 2019. Cyber intrusion detection by combined feature selection algorithm. J. Infor. Secur. Applic. 44, 80–88. https://doi.org/10.1016/j.jisa.2018.11.007.

Moustafa, N., Slay, J., 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 military communications and information systems conference (MilCIS). IEEE, pp. 1–6. https://doi.org/10.1109/MilCIS.2015.7348942.

Qu, X., Yang, L., Guo, K., Ma, L., Sun, M., Ke, M., Li, M., 2021. A survey on the development of self-organising maps for unsupervised intrusion detection. Mobile Network. Applic. 26, 808–829. https://doi.org/10.1007/s11036-019-01353-0.

Sarker, I.H., Abushark, Y.B., Alsolami, F., Khan, A.I., 2020. Intrudtree: a machine learning-based cyber security intrusion detection model. Symmetry 12 (5), 754. https://doi.org/10.3390/sym12050754.

Sarker, I.H., Furhad, M.H., Nowrozy, R., 2021. Ai-driven cybersecurity: an overview, security intelligence modelling and research directions. SN Comp. Sci. 2, 1–18. https://doi.org/10.1007/s42979-021-00557-0.

Soe, Y.N., Feng, Y., Santosa, P.I., Hartanto, R., Sakurai, K., 2020. Machine learning-based IoT-botnet attack detection with sequential architecture. Sensors 20 (16), 4372. https://doi.org/10.3390/s20164372.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defence applications. Ieee, pp. 1–6. https://doi.org/10.1109/CISDA.2009.5356528.

Xing, L., 2020. Cascading failures in the Internet of things: review and perspectives on reliability and resilience. IEEE Int. Things J. 8 (1), 44–64. https://doi.org/10.1109/JIOT.2020.3018687.

Ye, J., Chow, J.H., Chen, J., Zheng, Z., 2009. Stochastic gradient boosted distributed decision trees. In: Proceedings of the 18th ACM conference on Information and Knowledge Management, pp. 2061–2064. https://doi.org/10.1145/1645953.1646301.

Zhang, F., Kodituwakku, H.A.D.E., Hines, J.W., Coble, J., 2019. Multilayer data-driven cyber-attack detection system based on network, system, and process data for industrial control systems. IEEE Transac. Industr. Infor. 15 (7), 4362–4369. https://doi.org/10.1109/TII.2019.2891261.

Zhang, G., Li, J., Bamisile, O., Xing, Y., Cao, D., Huang, Q., 2023. Identification and classification for multiple cyber-attacks in power grids based on the deep capsule CNN. Eng. Applic. Artific. Intell. 126, 106771. https://doi.org/10.1016/j.engappai.2023.106771.

Zhang, X., Xu, M., Da, G., Zhao, P., 2021. Ensuring confidentiality and availability of sensitive data over a network system under cyber threats. Reliab. Eng. Syst. Safety 214, 107697. https://doi.org/10.1016/j.ress.2021.107697.