ISSN 2723-6471

Ensembling Methods for Data Privacy in Data Science

N. Mahendiran[1,*], B. L. Shivakumar[2], Siti Sarah Maidin[3], Hao Wu[4]

[1,2] *Department of Computer Science, Sri Ramakrishna College of Arts and Science, Coimbatore-641006, India*

[3]*Faculty of Data Science and Information Technology (FDSIT), INTI International University, Nilai, Malaysia*

[3]*Department of IT and Methodology, Wekerle Sandor Uzleti Foiskola, Jazmin utca 10, 1083 Budapest, Hungary*

[4]*Faculty of Liberal Arts, Shinawatra University, Thailand*

(Received: July 31, 2024; Revised: August 19, 2024; Accepted: September 14, 2024; Available online: October 15, 2024)

Abstract

The rapid advancement of technology has unified systems, data storage, applications, and operations, providing continuous services to organizations. However, this integration also introduces new vulnerabilities, particularly the risk of cyber-attacks. Malware and digital piracy pose significant threats to data security, with the potential to compromise sensitive information, leading to severe financial and reputational damage. This study aims to develop an effective method for detecting malware-infected files on storage devices within the Internet of Things (IoT) environment. The proposed approach utilizes a stacked regression ensemble for data pre-processing and the Sea Lion Optimization Algorithm (sLOA) to extract salient features, enhancing the classification process. Using malware data from an intrusion detection dataset, an ensemble classification technique is applied to identify malicious infections. The experimental results demonstrate that the proposed method achieved an accuracy of 98%, a precision of 99.6%, a recall of 96%, and an F-measure of 95% by the final iteration, significantly outperforming existing techniques in addressing cyber-security challenges within IoT systems.

Keywords: Cyber Security, Data Privacy, Ensemble Approach, Feature Selection, Optimization, and Classification

1. Introduction

These objects are equipped with sensors, electronic chips, and various technologies. International identification of each device is facilitated using Radio Frequency Identifier (RFID) tags. These intelligent devices can be remotely controlled and monitored, enabling interaction with other nodes within the network [1]. IoT allows a wide range of intelligent physical things, service sectors, cloud platforms, and connected apps. IBM predicts a surge in the number of internet-connected devices, estimating that it will reach 50 billion by 2020.

The proliferation of smart devices is expected to result in an expanded network of communication and a significant increase in the volume of data exchange facilitated by cloud infrastructure [2]. The applications of IoT-enabled solutions are diverse, spanning the creation of educational systems, smart cities, e-banking, e-shopping platforms, entertainment setups, human safety measures, industrial operations, health maintenance, and more [3]. However, the constant connectivity of IoT devices also poses security challenges, making them susceptible to open attacks.

The industrial IoT-cloud, in particular, becomes a target for malware infections, posing a threat to security. Another concern is software piracy, where illicit use of source codes from existing software is employed to create versions that appear original. This unauthorized duplication is achieved through reverse engineering processes, allowing individuals to replicate the logic of the original software in a different source code [4] and [5].

In order to detect stolen source code in pirated software, and sophisticated software plagiarism approaches are necessary [6]. Clone identification, source code resemblance identification, software flaw investigation, and software birthmark inquiry are some of the plagiarisms checking approaches are presented. Structural and text-based evaluation

***Corresponding author: N. Mahendiran (nmahendran.snr@gmail.com)**
DOI: https://doi.org/10.47738/jads.v5i4.341

are the most common strategies used [6]. The structure-based approach looks at the fundamental structure of source codes, syntax trees, graph behavior, and subroutine function call graphs. As a result, it is restricted to a single programming language framework. As a result, if a cracker overuses the logic of the source program in a diverse programming language, it is difficult to detect owing to the differences in structural behavior [7]. Creating advanced software for detecting plagiarism and malware, industrial IoT cloud services can be employed to ensure the security of smart devices. With the proliferation of IoT networks, malicious attacks are becoming more prevalent [8]. Malware threats are often crafted to compromise the privacy of IoT nodes, computers, and smartphones connected to the internet. Numerous scanning strategies, primarily based on specific signatures, have been developed to identify malware targeting Windows systems. Two fundamental methodologies for malware recognition include dynamic and static techniques [9].

In the dynamic approach, malware patterns are learned in real-time as the code executes within a virtual space. Analyzing function calls, function parameters, data flow, instruction traces, and visually inspecting code can all unveil malicious activity [10]. Automated web tools are available for studying the dynamic behavior of harmful code. As a result, these innovative approaches contribute to the enhancement of security measures in the face of escalating cybersecurity threats in the expanding realm of IoT networks. Anubis, CW Sandbox, and the TT Analyzer, for example. Because every dynamic behavior of source code must be monitored, this technique takes longer. Static malware investigational approaches do not necessitate the execution of source code in real time. It might be used to acquire malware binaries' structural framework [11].

Opcode frequency, control flowgraph, string signature, and n-gram are static signature-based malware detection approaches. Before adopting static based methods, disassembly tools such as OllyDbg and IDA Pro are used to reveal the executable [12]. The hidden patterns in binary executable files are extracted using these disassemblers. The encoded text is then retrieved from executable using these criteria [13]. The byte sequence approach is a static-based investigation that may be used to these patterns for extracting n-byte sequences. A static examination for extracting the structural investigation of codes is the functional call graph [14]. This article discusses about the detection of malware and other intrusion across the IoT environment. The collected information may have unwanted information that can be removed using ensemble stacked regression approach and the significant features are retrieved using LOA. The features are utilized for the classification of threats and other cyber-attacks.

The remainder of the article is systemized as follows: related approaches in cyber security is reviewed in Section 2, proposed methodology for data privacy and cyber security is detailed in Section 3, results acquired from the proposed methodology is illustrated in Section 4, and the article is concluded in Section 5.

2. Related Work

Platform plays very crucial role while developing IoT application because it allows you to deploy and execute your developed application. Platform can be a combination of hardware and software or only software. On the top of it, various different applications can operate. IoT cloud platforms offers a complete set of generic functionalities which can be helpful for building IoT application by providing complete life cycle of IoT application development which begins with application design and ends with its deployment along with its maintenance. Now a day ‘s numerous IoT cloud platforms are available [15], which could be used for developing an IoT solution. This subsection surveys most popular IoT cloud platforms. Finally, the aim is to study all the already available IoT platforms specifically which support cloud with their advantages and limitation.

[16] projected deep learning approach for the identification of cyber-attacks. This article emphasizes the prominent aspects and the disadvantages of deep learning architectures. The drawback of CNN is rectified by the hybridization process of CNN with LSTM. The approach is effective in identification and classification of threats. The proposed scheme achieves 92% accuracy during the process of classification of attacks.

Amin Dastanpour and RajaAzlina Raja Mahmood et al., [17] proposed IDS for network attacks that detects intrusion based on genetic algorithm (GA) with support vector machine (SVM) to increase detection accuracy with reduced false alarm rates. Results show that the system achieves about 99% detection rate at different number of reduced features with comparatively low false positive rates ranging between0.43% to 0.6%. The main demerit of this system is the increased time consumption for training the model because of the 21 number of features selected by GA with SVM and LCFS and the 31 features required by FFSA to detect the attacks.

E. Kesavulu Reddy et al. [18] have designed a novel Intrusion Detection System (IDS) for network-based attacks, focusing on recognizing distinctive attributes of system users and pinpointing statistically significant deviations from typical user behavior through the utilization of neural networks. This neural network is trained on discernible patterns, leading to a system with high detection accuracy attributed to its learning capacity. A key advantage of neural networks lies in their swift ability to promptly identify misuse patterns in user behavior due to their inherent speed.

However, a noteworthy drawback of the system is its substantial demand for a considerable amount of training data to ensure that the outcomes are statistically reliable.

Harshit Saxena, Dr. Vineet Richariya et al., [19] proposed an IDS based on PSO for effective training and SVM acts as a classifier to identify network-based attack effectively. Here K-means clustering algorithm is mainly used for clustering different training subsets to reduce the false alarms thereby increasing detection accuracy. The main advantage is the increased detection accuracy. The major demerit observed is that the proposed system is good for Denial of Service (DoS) attack but fails to achieve good detection rate in case of U2R and R2Lattacks.

In their proposed system, Wang et al. [20] utilized the Artificial Bee Colony (ABC) feature selection algorithm in conjunction with SVM. By employing only, the top five feature subsets selected by the ABC algorithm, their system demonstrated impressive accuracy. The authors also investigated the performances of Particle Swarm Optimization (PSO) and GA) when integrated with SVM. The ABC-SVM combination achieved a perfect accuracy of 100%, while PSO-SVM and GA-SVM yielded accuracies of 98% and 97%, respectively.

3. Methodology

3.1. Pre-Processing

By reducing artefacts, pre-processing methods can improve the quality of analytical signals. To improve the quality of analytical signals and address the challenges of reducing artefacts in various applications, three primary techniques for ensemble pre-processing selection and fusion are, multi-block data analysis-based ensemble approaches, combine complementary information from multiple pre-processing techniques to boost chemometrics models. Stratified bagging for training base classifiers and integrating data preprocessing methods where a novel framework employs stratified bagging to train base classifiers and integrate data preprocessing methods, Dynamic Ensemble Selection and Data Preprocessing, this approach evaluates the performance of different ensemble selection techniques and data preprocessing methods, are used to reduce the over-fitting.

The stacked regression methodology entails the training of numerous predictive models employing varied preprocessing, followed by their combination using cross-validation or model averaging techniques. This amalgamation of complementary information gleaned from diverse preprocessing procedures frequently leads to the creation of a calibration model that exhibits enhanced stability and accuracy, a conclusion supported by insights derived from Monte Carlo cross-validation (MCCV) stacking regression results [21].

To alter the data, certain typical preprocessing procedures are performed initially. Second, optimum calibration methods are based on data that has been preprocessed in different ways. The next stage is to merge the models to create an ensemble calibration and forecasting framework. MCCV stacked regression is /used to calculate the combination coefficients vector W. W is calculated using non-negative least squares (NNLS) in MCCV stacked regression using the following equation:

$$X = [\widehat{x_1}, \widehat{x_1} \ldots \ldots \ldots .., \widehat{x_n}]K \tag{1}$$

Here, X represents the reference level values for the samples excluded during MCCV resampling, and $\widehat{x_i}$ signifies the corresponding intensity calculated by the i-th model. The variable n denotes the number of models, corresponding to the various preprocessing evaluated. The stacking method is straightforward to implement and can be carried out using conventional chemometric toolboxes that support Partial Least Squares (PLS) modeling.

3.2. Feature Selection

Lions stand out as the most socially oriented among all wild cat species, exhibiting a unique combination of cooperation and aggression. The intrigue surrounding lions is heightened by their significant sexual dimorphism, impacting both their social behavior and appearance. These wild felids are divided into two distinct social groups: residents and nomads. Resident lions organize themselves into pride groups, typically comprising approximately five females, along with their male and female offspring, and one or more adult males. As young males mature, they are expelled from their natal pride. In contrast, nomads, representing the second organizational behavior, roam intermittently either in pairs or individually. Notably, males excluded from maternal prides are more inclined to form pairs. It's essential to recognize that a lion's lifestyle is dynamic; residents may transition into nomads, and vice versa, adding a layer of complexity to their social dynamics.

The LOA, or Lion Optimization Algorithm, is a meta-heuristic method that operates on a population-based approach. In its initial stage, the population is randomly generated across the solution space, and each outcome within this algorithm is referred to as a “Lion”. The SLOA algorithm mimics the way sea lions hunt. The artificial hunting behavior with random or optimal search agent to chase the bait ball (prey) and the use of sea lions' vocalizations and whiskers are the algorithm's strong points.

3.2.1. Phases of Detection and Tracking:

As previously noted, sea lions utilize their whiskers to determine the size, shape, and location of their prey. A sea lion's ability to perceive and locate prey is enhanced when their whiskers point in the opposite direction of the water's waves. The whiskers did, however, vibrate less than when they were oriented in the same direction as they are now.

3.2.2. Phase of Vocalization:

Sea lions are regarded as amphibians. Put differently, sea lions are both aquatic and terrestrial animals. In water, their sounds travel four times more quickly than in air. When pursuing and hunting in groups, sea lions use a variety of vocalizations to communicate with one another. They also call other members who remain on the coast using their sound. Sea lions pursue and imprison their prey in order to get them as close to the ocean's surface as possible. They also have tiny ears that can hear sounds both above and below the water. When a sea lion spots a meal, he signals to his fellows to surround and attack the victim.

3.2.3. Exploitation Phase

Sea lions can locate their intended victim and circle around them. The leader, or top search agent, directs the hunt by spotting the prey and alerting other hunters to its location.

3.2.4. Exploration Phase

Sea lions use their whiskers to haphazardly search for prey, swimming in zigzag patterns. Because of this, random values are used in this investigation. Sea lions are forced to move away from the intended prey and the sea lion leader if is larger than one or less than negative one. Starting with random solutions, the suggested (SLOA) algorithm operates. Whether it is a random search agent or the greatest solution, each search agent updates its location. Over the course of iterations, parameter (C) is reduced from 2 to 0 to supply both the exploration and exploitation phases. More specifically, a search agent is selected at random when it is bigger than one. Conversely, search agents update their positions when it is less than one. Ultimately, the SLOA algorithm stops when an ending requirement is satisfied.

3.3.Feature Selection

The features acquired from SLOA is passed to the diverse classifiers in the ensemble approach. To establish a strong classifier, the ensemble classifier is utilized that combines numerous basis classifiers. Ensemble classifiers have traditionally been utilized to improve the accuracy and performance of base learning approaches.

3.3.1. Linear SVM

The SVM stands as a globally employed supervised learning approach rooted in statistical learning theory. Widely utilized, SVM plays a crucial role in determining the abnormality of images. Its applications span a diverse range, initially designed for binary classification and subsequently enhanced for multiclass classification. SVM operates as a generalized linear classifier, utilizing a learning method based on the linear function hypothesis space. The features are intricately linked to the class labels of the respective topics. The SVM process involves creating a series of binary classifiers, denoted as f1, f2 ,...,fM, each proficient in distinguishing one class from the remaining (M-1) classes. Subsequently, these classifiers are amalgamated to form a multi-class classification system.

The training process incorporates error reduction as a crucial component, and its formulation is expressed as:

$$\phi(w,\xi) = \frac{1}{2}\sum_{a=1}^{A}\left(w_a^T, w_m\right) + C\sum_{i=1}^{N}\sum_{a\neq y_i}\xi_i^a \tag{2}$$

Subject to criteria

$$y_i\left(w^T\phi(x_i) + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0, i = 1, \dots, N\ \ m \in \{1,2, \dots . A\}\{y_i\} \tag{3}$$

In SVM classification, C represents the volume constant, b is a constant, w is the vector of coefficients, and ξ_i denotes parameters for non-separable data in SVM classification. The volume constant, C, remains consistent, and b is determined as a constant, while w represents the vector of coefficients, and ξ_i signifies parameters for non-separable data. The training instances, denoted by the index i, encompass N instances. Here, yi ∈ {1,...,M} signifies the multiclass labels of the feature vectors, and ∈ {1,...,M}/yi represents multiclass labels, excluding yi, with xi being the self-regulating variables. The kernel serves to transform the data from the input space to the feature space. In the realm of statistical learning theory, minimizing prediction error aligns with augmenting the margin, where $\gamma = \frac{2}{\|w\|}$. The expression for the decision boundary is articulated as follows:

$$f(x) = \arg\max_a(w_m^T . x + b_a) \qquad \text{for } a = 1,2,3 \dots a \tag{4}$$

The classifier is in the form of

$$f(x) = \sum k_i x_i^T x + b_a \tag{5}$$

Initially, it gained widespread recognition for its success in handwritten digit recognition and demonstrated empirical performance. The linear kernel is the elementary form of the kernel, represented as follows:

$$K\left(x_i, x_j\right) = x_i^T, x_j \tag{6}$$

3.3.2. Random Forest

Bagging and the random subspace approach are combined in random forest. Each subset in bagging has the same size as the original training dataset and is used to build a decision tree. Bagging produces more consistent results than a single tree. In the random forest, an ensemble of distinct models is formed. It averages all of the forecasts from these different trees. The random forest combines many decision trees to produce a more accurate and consistent forecast. It uses random subspace sampling to generate a tree from each bootstrap sample. One of the most appealing features of the random forest is that it is extremely adaptable, never overfits, and provides superior forecast accuracy.

3.3.3. Boosted Tree

The eXtreme Gradient Boosting (XGBoost) software was designed and constructed to be effective, versatile, and adaptive. A tree learning algorithm and an effective linear model generator are included in the software. Regression, ranking, and classification are among the objective functions that are supported. The gradient boosting paradigm is derivated in an efficient and scalable manner. The regularised model is used to control the complexity of the model, making it easier to learn and preventing overfitting. XGBoost allows for parallelization.

It employs parallelization to select the best dividing points for iteration, which speeds up the training process. The tree building is halted ahead of schedule when the forecasted outcomes are favourable, enabling an acceleration in the training pace, it also provides the flexibility to select sample weights for the first and second derivatives, denoted as g and h, respectively. Adjusting the weight allows for a heightened focus on specific samples. XGBoost stands out as one of the most extensively utilized machine learning approaches. The fundamental principles of classic gradient boosting are as follows:

$$F_m(x) = \sum_{i=1}^{m} \beta_i f_i(x) \tag{7}$$

The variable 'm' represents the proportion of base learners, the coefficient is denoted by 'β', 'f' represents the base learner, and 'F' signifies the overarching framework. The ultimate objective is to attain a superior, comprehensive model that minimizes the loss function to the greatest extent possible. This objective is expressed as follows:

$$F_m(x) \arg\min \sum_{i=1}^{n} L(y_i F_m(x)) = \beta_m(x) \arg\min \sum_{i=1}^{n} L(y_i, F_{m-1}(x) + \beta_m F_m(x)) \tag{8}$$

The variable F is subsequently assigned weights from several base learners, preventing a simultaneous solution. Gradient boosting employs a greedy approach, with model F initially assuming the form of a constant function containing only one base learner. Over time, the coefficients of this base learner are adjusted to enhance the performance of F. The primary objective of gradient boosting is to rapidly reduce the loss function by incorporating new terms that are determined by the negative gradient of the loss function at each step. A potential structure for the new term is outlined as follows:

$$\beta_m F_m(x) = \gamma \frac{\partial L(y_i, F_{m-1}(x)}{\partial F_{m-1}(x)} \tag{9}$$

where the size of step is indicated by γ that holds negative sign as well as gradient β. The gradient with pseudo residual is,

$$R_{im} = \gamma \frac{\partial L(y_i, F_{m-1}(x)}{\partial F_{m-1}(x)} \tag{10}$$

For each sample in the training space, the algorithm determines the value F. The analysis identifies the Pseudo residual R. The current base learner fm is defined and trained based on the values x and y in the training space. The value of a trained base learner is incorporated into an equation to minimize the loss function, and the resulting value is obtained. The initial general approach is outlined as follows:

$$F_m(x) = F_{m-1}(x) + \beta_m f_i(x) \tag{11}$$

Nevertheless, Xgboost improves the regularization by adapting L within the conventional XGB framework. L is:

$$L = \sum_{i} l(\hat{y}_i, y_i) + \sum_{k} \Omega(f_k) \tag{12}$$

The convex loss is the initial differentiable term that computes the disparity between the objective and predicted values. In XGBoost, the objective function Object and the optimal leaf node fraction w are defined as:

$$\text{objec} = -\frac{1}{2} \sum_{j-1}^{T} \frac{G_j}{H_j + \lambda} + \gamma T \tag{13}$$

$$w_i^* = \frac{G_j}{H_j + \lambda} \tag{14}$$

Various machine learning frameworks are suited to different tasks. Deep neural networks excel at handling high-dimensional and extensive data, such as images, sounds, and text, by modeling spatial and temporal relationships. In principle, the application impact of the XGBoost approach is expected to be more effective due to the low-dimensional nature of the observed tabular data in the evaporation duct and the relatively small dataset.

4. Results and Discussion

The performance of classification is investigated by comparing the performance metrics namely accuracy, precision, recall and f-measure. The NSL-KDD dataset serves as a pivotal resource in the realm of cybersecurity, particularly in the domain of intrusion detection. Specifically designed for evaluating and benchmarking intrusion detection systems and machine learning models, the dataset is an enhanced version of the original KDD Cup 1999 dataset. Originating from a simulated military network environment, the NSL-KDD dataset includes instances of both normal network traffic and various types of attacks, encompassing Denial of Service (DoS), Probe, Unauthorized access from a remote machine (R2L), and Unauthorized access to user privileges (U2R). With features describing network connections such as protocol type, service, flags, duration, and source/destination IP addresses, the dataset provides a diverse set of examples for training and testing intrusion detection systems. The dataset is often provided in two versions, the original and a reduced version that addresses redundancy issues, offering a more balanced class distribution. Researchers and practitioners leverage the NSL-KDD dataset to explore the efficacy of machine learning algorithms in identifying and classifying network intrusions, recognizing its importance in enhancing cyber resilience and understanding the intricacies of securing network environments.

4.1. Accuracy

In cybersecurity and data privacy, accuracy is crucial to ensure that the model is making correct predictions across various classes, minimizing false positives and false negatives. It gives a general sense of the model's effectiveness. Accuracy is given as (15).

$$\text{Accuracy} = \frac{\text{True Positive (TP)} + \text{True Negative(TN)}}{\begin{array}{c}\text{True Positive(TP)} + \text{True Negative(TN)} + \\ \text{False Positive(FP)} + \text{False Negative(FN)}\end{array}} \tag{15}$$

Table 1 presents a comparison of the accuracy achieved by three different methods—NIDS, ENetRM, and ML-ENC—across several iterations (50, 100, 150, 200, and 250). The data shows that as the number of iterations increases, the performance of each method generally improves. At iteration 50, NIDS starts with an accuracy of 61%, ENetRM with 68%, and ML-ENC leading significantly at 95%. By iteration 100, NIDS rises to 74%, ENetRM to 71%, while ML-ENC maintains its superior accuracy at 96%. As iterations continue to 150, NIDS reaches 88%, ENetRM 73%, and ML-ENC achieves 97%. At iteration 200, NIDS sees a slight improvement to 89%, ENetRM rises to 75%, and ML-ENC continues to outperform with 97.5%. Finally, at iteration 250, NIDS peaks at 91%, ENetRM shows significant growth to 81%, while ML-ENC remains dominant with an accuracy of 98%.

Table 1. Comparison of Accuracy

Iteration	NIDS	ENetRM	ML-ENC
50	61	68	95
100	74	71	96
150	88	73	97
200	89	75	97.5
250	91	81	98

Figure 1 illustrates this comparison graphically, highlighting the accuracy trends of each method across the iterations. The graph shows a steady upward trajectory for NIDS, a slower but improving rise for ENetRM, and consistently superior performance by ML-ENC, whose accuracy remains higher than the other two methods at every iteration. This visual representation clearly underscores ML-ENC's consistent dominance in accuracy, while also reflecting the gradual improvements made by NIDS and ENetRM as iterations increase.

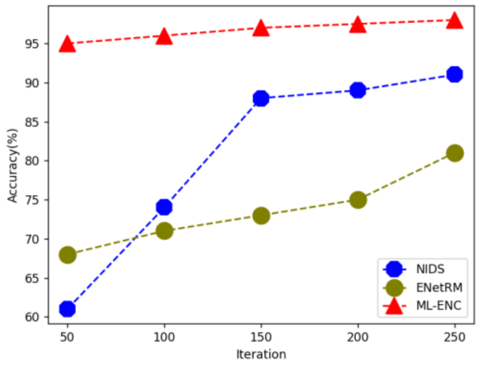

Figure 1. Comparison of Accuracy

4.2. Precision

In the context of cybersecurity, precision is important to minimize false positives. A high precision indicates that when the model predicts a positive outcome (e.g., detecting a security threat), it is likely to be correct. Precision is given as,

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{16}$$

Table 2 presents a comparison of the precision values for three methods—NIDS, ENetRM, and ML-ENC—across different iterations (50, 100, 150, 200, and 250). Precision, which reflects the accuracy of positive predictions by minimizing false positives, is an important metric for assessing the effectiveness of these methods. The table shows that at iteration 50, ML-ENC already outperforms the other two methods with a precision of 95%, while NIDS and ENetRM start at 54% and 56%, respectively. As the iterations increase to 100, ML-ENC slightly improves to 95.5%, with NIDS and ENetRM rising more significantly to 73% and 75%, respectively. By iteration 150, ML-ENC reaches 96%, NIDS achieves 91%, and ENetRM demonstrates strong improvement with a precision of 94%. At iteration 200, ML-ENC continues to lead with 97%, while NIDS and ENetRM reach 95% and 96%, respectively. Interestingly, at iteration 250, NIDS experiences a drop to 86%, while ENetRM peaks with 99% precision, and ML-ENC further improves to 99.6%.

Table 2. Comparison of Precision

Iteration	NIDS	EnetRM	ML-ENC
50	54	56	95
100	73	75	95.5
150	91	94	96
200	95	96	97
250	86	99	99.6

Figure 2 provides a visual representation of the precision comparison across these iterations, highlighting the steady improvements made by all three methods. ML-ENC consistently maintains higher precision than NIDS and ENetRM throughout the iterations. ENetRM, after a slower start, shows remarkable growth in precision by iteration 250, nearly matching ML-ENC. NIDS, although improving in the earlier iterations, experiences a decline in precision at iteration 250. Overall, both the table and figure emphasize ML-ENC's superior performance in terms of precision, while ENetRM demonstrates significant gains in later iterations.

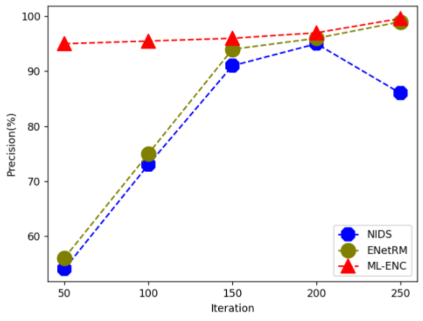

Figure 2. Comparison of Precision

4.3. Recall

In cybersecurity, recall is crucial to ensure that the model can identify and capture as many true positive instances as possible, minimizing false negatives. It is particularly important when dealing with potential security threats. The recall is given as,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{17}$$

Table 3 shows a comparison of recall values for three methods—NIDS, ENetRM, and ML-ENC—across different iterations (50, 100, 150, 200, and 250). Recall, also known as the true positive rate, measures the ability of a method to identify all relevant instances, making it crucial in contexts like cybersecurity where missing potential threats (false negatives) can have severe consequences.

At iteration 50, ML-ENC demonstrates the highest recall with 91%, significantly outperforming NIDS and ENetRM, which have recall values of 55% and 58%, respectively. As the number of iterations increases to 100, ML-ENC slightly improves to 92%, while NIDS and ENetRM see notable gains, reaching 73% and 78%. By iteration 150, ML-ENC's recall improves further to 94%, NIDS jumps to 92%, and ENetRM leads with 97%. However, at iteration 200, NIDS and ENetRM experience a sharp decline in recall to 10% and 17%, respectively, while ML-ENC continues to improve with a recall of 95%. Finally, at iteration 250, NIDS and ENetRM recover slightly to 29% and 36%, but ML-ENC maintains its superior performance with a recall of 96%.

Table 3. Comparison of Recall

Iteration	NIDS	EnetRM	ML-ENC
50	55	58	91
100	73	78	92
150	92	97	94
200	10	17	95
250	29	36	96

Figure 3 provides a graphical representation of the recall comparison between NIDS, ENetRM, and ML-ENC across different iterations. The graph clearly highlights that ML-ENC consistently achieves the highest recall throughout all iterations, maintaining a strong ability to identify relevant instances. ENetRM shows impressive gains in the early and middle iterations but suffers a significant drop in recall performance in iteration 200, along with NIDS, before partially recovering by iteration 250. NIDS follows a similar trend but remains consistently below both ENetRM and ML-ENC in terms of recall.

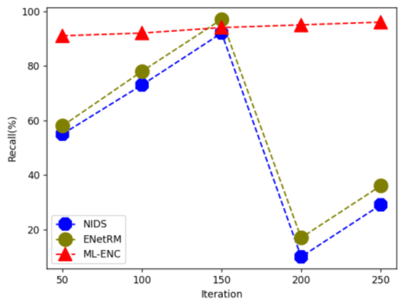

Figure 3. Comparison of Recall

This comparison emphasizes ML-ENC's consistent superiority in recall, making it highly effective at identifying potential security threats across all iterations, while NIDS and ENetRM show more variability, especially in later iterations.

4.4. F-Measure

In the context of data privacy and cybersecurity, where achieving a balance between precision and recall is often necessary, the F-measure is a valuable metric. It helps in evaluating a model's overall effectiveness, especially when the cost of false positives and false negatives needs to be balanced. The F-measure is given as,

$$F - \text{measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{18}$$

Table 4 provides a comparison of the F-measure for three methods—NIDS, ENetRM, and ML-ENC—across several iterations (50, 100, 150, 200, and 250). The F-measure, or F1-score, is a harmonic mean of precision and recall, offering a balanced evaluation of a model's ability to minimize both false positives and false negatives. This metric is particularly important in cybersecurity, where an ideal balance between these factors is critical.

At iteration 50, ML-ENC achieves the highest F-measure at 92%, significantly outperforming NIDS at 52% and ENetRM at 55%. By iteration 100, ML-ENC shows a slight improvement to 92.5%, while NIDS and ENetRM also see substantial gains, reaching 70% and 73%, respectively. At iteration 150, ML-ENC continues its upward trajectory with an F-measure of 93%, while NIDS improves to 88% and ENetRM rises further to 92%. However, by iteration 200, NIDS and ENetRM experience a sharp decline in performance, with F-measure values dropping to 5.5% and 10%, respectively, while ML-ENC continues to excel, improving its F-measure to 94%. In the final iteration (250), NIDS and ENetRM recover slightly, reaching 23% and 29%, but ML-ENC continues to dominate with an F-measure of 95%.

Table 4. Comparison of F-Measure

Iteration	NIDS	EnetRM	ML-ENC
50	52	55	92
100	70	73	92.5
150	88	92	93
200	5.5	10	94
250	23	29	95

Figure 4 visually represents the comparison of F-measure values across different iterations for NIDS, ENetRM, and ML-ENC. The graph highlights that ML-ENC consistently maintains the highest F-measure across all iterations, reflecting its ability to balance precision and recall effectively. NIDS and ENetRM follow a similar pattern, with strong improvements in early iterations but a steep drop in iteration 200, before recovering somewhat by iteration 250. ML-ENC, in contrast, shows steady improvement throughout all iterations, reinforcing its superior performance in achieving a balance between false positives and false negatives. This comparison underlines the fact that ML-ENC is more effective at maintaining a balanced and robust performance, making it highly suitable for tasks in cybersecurity where both precision and recall are crucial.

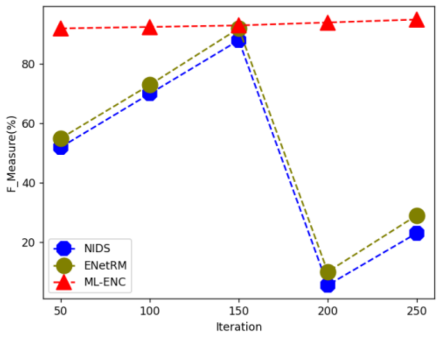

Figure 4. Comparison of F-measure

In the comprehensive comparative analysis of cybersecurity models, encompassing NIDS, ENetRM, and the innovative Machine Learning-based ENsemble Classifier (ML-ENC), key performance metrics were meticulously evaluated across multiple iterations. Accuracy, a critical measure of overall correctness, highlighted ML-ENC's consistent outperformance, with accuracy ascending from 95% to an impressive 98% over 250 iterations. Precision, pivotal for minimizing false positives, exhibited remarkable superiority in ML-ENC, consistently surpassing NIDS and ENetRM and reaching 99.6% precision at iteration 250. The assessment of recall, measuring the model's adeptness in identifying all pertinent instances, demonstrated ML-ENC's consistently higher performance, achieving 96% recall at iteration 250. Furthermore, the F-measure, a balanced metric considering precision and recall, underscored ML-ENC's efficacy, maintaining a higher F-measure compared to its counterparts and reaching 95% at iteration 250. This in-depth analysis unequivocally positions ML-ENC as a robust and reliable solution for cybersecurity applications, offering superior accuracy, precision, recall, and F-measure, and thereby providing a holistic and balanced approach to threat detection and classification. From the observation of classification outcome, it is identified that the proposed ML-ENC is highly effective.

5. Conclusion

This study demonstrates that threats from malware and digital piracy present significant risks to IoT security, with the potential to compromise sensitive data, leading to financial and reputational damage. The proposed methodology successfully detects malware-affected files on IoT devices by utilizing a stacked regression ensemble technique for data pre-processing and the Sea Lion Optimization Algorithm (sLOA) for feature extraction. The ensemble classification approach applied to malware data from an intrusion detection dataset has shown to be highly effective in identifying malicious infections.

The results underscore the effectiveness of using ensemble learning techniques combined with optimization algorithms to improve malware detection in IoT environments. This research contributes to the growing body of work aiming to enhance cybersecurity through advanced machine learning models, demonstrating their applicability to real-world IoT-based security systems. Despite its success, the study has limitations, including its reliance on a specific intrusion detection dataset. The method has not been tested across a wide variety of datasets or threat types, which could impact its generalizability to other cybersecurity scenarios.

Future research should focus on integrating additional deep learning techniques to improve the model's ability to classify the severity of threats and adapt to evolving cyber threats. Further testing on diverse datasets will also enhance the robustness of the proposed approach. Additionally, efforts should be made to refine the connectivity between IoT devices and improve the algorithmic efficiency to ensure faster and more accurate threat detection. Overall, this study highlights the potential of ensemble learning and optimization algorithms in addressing modern cybersecurity challenges, especially within IoT ecosystems. Continued innovation and adaptation to emerging threats will be crucial to maintaining robust cybersecurity defenses.

6. Declarations

6.1. Author Contributions

Conceptualization: N.M., B.L.S., S.S.M., and H.W; Methodology: N.M.; Software: B.L.S. and H.W.; Validation: N.M., B.L.S., and S.S.M.; Formal Analysis: N.M., B.L.S., and S.S.M.; Investigation: N.M.; Resources: B.L.S. and S.S.M.; Data Curation: N.M.; Writing Original Draft Preparation: N.M. and B.L.S.; Writing Review and Editing: S.S.M., N.M., and B.L.S.; Visualization: N.M. and H.W.; All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. 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.

References

[1] R. Ch, T. R. Gadekallu, M. H. Abidi, and A. Al-Ahmari, "Computational system to classify cyber crime offenses using machine learning," *Sustainability*, vol. 12, no. 10, pp. 4087-4100, May 2020.

[2] M. A. R. Khan, S. N. Shavkatovich, B. Nagpal, A. Kumar, M. A. Haq, V. J. Tharini, and M. B. Alazzam, "Optimizing hybrid metaheuristic algorithm with cluster head to improve performance metrics on the IoT," *Theor. Comput. Sci.*, vol. 2022, no. 5, pp. 1-15, May 2022.

[3] V. J. Tharini and S. Vijayarani, "IoT in healthcare: Ecosystem, pillars, design challenges, applications, vulnerabilities, privacy, and security concerns," in *Incorporating the Internet of Things in healthcare applications and wearable devices*, IGI Global, vol. 2020, no. 1, pp. 1-20, Jan. 2020.

[4] V. J. Tharini and B. L. Shivakumar, "An efficient pruned matrix aided utility tree for high utility itemset mining from transactional database," *Int. J. Intell. Syst. Appl. Eng.*, vol. 2023, no. 1, pp. 1-12, Jan. 2023.

[5] K. Shaukat, S. Luo, S. Chen, and D. Liu, "Cyber threat detection using machine learning techniques: A performance evaluation perspective," in *2020 Int. Conf. Cyber Warfare Secur. (ICCWS)*, IEEE, vol. 2020, no. 10, pp. 1-10, Oct. 2020.

[6] M. Rege and R. B. K. Mbah, "Machine learning for cyber defense and attack," in *Data Analytics 2018*, vol. 2018, no. 1, pp. 83-90, Jan. 2018.

[7] N. M. Karie, V. R. Kebande, and H. S. Venter, "Diverging deep learning cognitive computing techniques into cyber forensics," *Forensic Sci. Int.: Synergy*, vol. 2019, no. 1, pp. 1-10, Jan. 2019.

[8] T. Manyumwa, P. F. Chapita, H. Wu, and S. Ji, "Towards fighting cybercrime: Malicious URL attack type detection using multiclass classification," in *2020 IEEE Int. Conf. Big Data*, vol. 2020, no. 12, pp. 1-6, Dec. 2020.

[9] I. Goni and M. Mohammad, "Machine learning approach to mobile forensics framework for cyber crime detection in Nigeria," *J. Comput. Sci. Res.*, vol. 2020, no. 1, pp. 1-10, Jan. 2020.

[10] P. U. Chinedu, W. Nwankwo, F. U. Masajuwa, and S. Imoisi, "Cybercrime detection and prevention efforts in the last decade: An overview of the possibilities of machine learning models," *Cybercrime Detection and Prevention Efforts*, vol. 2020, no. 1, pp. 1-15, Jan. 2020.

[11] N. M. Karie, V. R. Kebande, and H. S. Venter, "Diverging deep learning cognitive computing techniques into cyber forensics," *Forensic Sci. Int.: Synergy*, vol. 2019, no. 1, pp. 1-10, Jan. 2019.

[12] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in IoT networks," *IEEE Access*, vol. 2021, no. 8, pp. 1-20, Aug. 2021.

[13] C. Rupa, G. Srivastava, S. Bhattacharya, P. Reddy, and T. R. Gadekallu, "A machine learning driven threat intelligence system for malicious URL detection," in *The 16th Int. Conf. Availability, Reliability and Security*, vol. 2021, no. 8, pp. 1-10, Aug. 2021.

[14] R. O. Ogundokun, J. B. Awotunde, S. Misra, O. C. Abikoye, and O. Folarin, "Application of machine learning for ransomware detection in IoT devices," in *Artificial Intelligence for Cyber Security: Methods, Issues and Possible Horizons or Opportunities*, Springer, Cham, vol. 2021, no. 8, pp. 1-15, Aug. 2021.

[15] P. P. Ray, "A survey of IoT cloud platforms," *Future Comput. Inform. J.*, vol. 2016, no. 1, pp. 1-12, Jan. 2016.

[16] L. V. Duong, "Optimization of cyber-attack detection using the deep learning network," *Int. J. Comput. Sci. Netw. Secur.*, vol. 2021, no. 5, pp. 1-10, May 2021.

[17] A. Dastanpour and R. A. R. Mahmood, "Feature selection based on genetic algorithm and support vector machine for intrusion detection system," *Int. J. Comput. Appl.*, vol. 2013, no. 6, pp. 1-8, Jun. 2013.

[18] E. Kesavulu Reddy, "Neural networks for intrusion detection and its applications," in *Proc. World Congr. Eng.*, vol. 2013, no. 7, pp. 1-10, Jul. 2013.

[19] H. Saxena and V. Richariya, "Intrusion detection system using K-means, PSO with SVM classifier: A survey," *Comput. Sci. Eng.*, vol. 2014, no. 1, pp. 1-12, Jan. 2014.

[20] J. Wang, T. Li, and R. Ren, "A real-time IDSS based on artificial bee colony-support vector machine algorithm," in *Adv. Comput. Intell. (IWACI), Third Int. Workshop*, vol. 2014, no. 1, pp. 1-10, Jul. 2014.

[21] S. K. Palaniswamy and R. Venkatesan, "Hyperparameters tuning of ensemble model for software effort estimation," *J. Ambient Intell. Humaniz. Comput.*, vol. 2021, no. 8, pp. 1-12, Aug. 2021.