





Article

A Cloud Based Optimization Method for Zero-Day Threats Detection Using Genetic Algorithm and Ensemble Learning

Mike Nkongolo ^{1,*} , Jacobus Philippus van Deventer ^{1,*} , Sydney Mambwe Kasongo ^{2,*} , Syeda Rabab Zahra ³ and Joseph Kipongo ⁴ 

¹ Department of Informatics, Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, Pretoria 0028, South Africa

² Department of Industrial Engineering, School of Data Science & Computational Thinking, Stellenbosch University, Stellenbosch 7600, South Africa

³ School of Computing, National College of Ireland, D01 K6W2 Dublin, Ireland; syedarababzahra0@gmail.com

⁴ Department of Electrical and Electronic Engineering Science, Faculty of Engineering and the Built Environment, University of Johannesburg, 5 Kingsway Ave., Rossmore, Auckland Park, Johannesburg 2092, South Africa; josephkips01@gmail.com

* Correspondence: u21629545@tuks.co.za (M.N.); phil.vandeventer@up.ac.za (J.P.v.D.); sydneybleuops@gmail.com (S.M.K.)



check for updates

Citation: Nkongolo, M.; van Deventer, J.P.; Kasongo, S.M.; Zahra, S.R.; Kipongo, J. A Cloud Based Optimization Method for Zero-Day Threats Detection Using Genetic Algorithm and Ensemble Learning. *Electronics* **2022**, *11*, 1749. <https://doi.org/10.3390/electronics11111749>

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 8 March 2022

Accepted: 24 April 2022

Published: 31 May 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: This article presents a cloud-based method to classify 0-day attacks from a novel dataset called UGRansome1819. The primary objective of the research is to classify potential unknown threats using Machine Learning (ML) algorithms and cloud services. Our study contribution uses a novel anomaly detection dataset that carries 0-day attacks to train and test ML algorithms using Amazon Web Services such as S3 bucket and SageMaker. The proposed method used Ensemble Learning with a Genetic Algorithm (GA) optimizer having three ML algorithms such as Naive Bayes (NB), Random Forest (RF), and Support Vector Machine (SVM). These algorithms analyze the dataset by combining each classifier and assessing the classification accuracy of 0-day threats. We have implemented several metrics such as Accuracy, F1-Score, Confusion Matrix, Recall, and Precision to evaluate the performance of the selected algorithms. We have then compared the UGRansome1819 performance complexity with existing datasets using the same optimization settings. The RF implementation (before and after optimization) remains constant on the UGRansome1819 that outperformed the CAIDA and UNSWNB-15 datasets. The optimization technique only improved in Accuracy on the UNSWNB-15 and CAIDA datasets but sufficient performance was achieved in terms of F1-Score with UGRansome1819 using a multi-class classification scheme. The experimental results demonstrate a UGRansome1819 classification ratio of 1% before and after optimization. When compared to the UNSWNB-15 and CAIDA datasets, UGRansome1819 attains the highest accuracy value of 99.6% (prior optimization). The Genetic Algorithm was used as a feature selector and dropped five attributes of the UGRansome1819 causing a decrease in the computational time and over-fitting. The straightforward way to improve the model performance to increase its accuracy after optimization is to add more data samples to the training data. Doing so will add more details to the data and fine-tune the model will result in a more accurate and optimized performance. The experiments demonstrate the instability of single classifiers such as SVM and NB and suggest the proposed optimized validation technique which can aggregate weak classifiers (e.g., SVM and NB) into an ensemble of the genetic optimizer to enhance the classification performance. The UGRansome1819 model's specificity and sensitivity were estimated to be 100% with three predictors of threatening classes (Signature, Synthetic Signature, and Anomaly). Lastly, the test classification accuracy of the SVM model improved by 6% after optimization.

Keywords: UGRansome1819; zero-day attacks; cloud computing; machine learning

1. Introduction

Cloud computing and network services are rapidly growing in popularity nowadays. Services and key operations of different businesses are migrating towards the cloud environment. Mobile computing, the Internet of Things (IoT), and emerging peer-to-peer networks represent trending technologies that are transforming how we communicate. Furthermore, the volume and diversity of unwanted traffic are also increasing [1]. Different approaches are used by intruders, professional developers, and computer hackers to attack cloud networking. The burden of identifying, managing, and safeguarding invasive traffic is becoming more challenging and expensive as these networks attempt to comply with organization-specific standards, quality regulations, and compliance obligations [2,3]. The majority of the front-line tools in a secure network architecture consist of Uniform Resource Locator (URL) filtering, firewalls, Intrusion Detection System (IDS), and access control. With advanced network threats and novel intrusion strategies, it is difficult to protect relevant data completely from attackers, Denial of Service (DoS) attacks, and data leakage [1–3]. In line with this, current research in the IDS field is aimed at improving AIDS using ML techniques. Intrusion detection can be divided into the anomaly and signature-based approaches. A signature-based approach is often used as a technique to detect misuse based on pre-defined parameters [4,5]. The report provides detailed information about predefined attacks and intrusions [6]. On the other hand, the anomaly detection method begins by getting to learn from a normal model and searching for anomalous patterns permitting the detection of unknown attacks [4,5]. The underpinning philosophy of the anomaly method is to identify abnormal pattern behavior that is detected in the normal model [2,5]. Cyclostationary threats being within the network traffic without the network administrator's awareness are generally known as a 0-day intrusion [7]. Various ML tactics along with GA and Information Theory, are utilized to optimize and improve the detection of network anomalies [2–4,8]. The AIDS assesses and analyzes network and host traffic: In a Host-based Intrusion Detection System (HIDS) network features including system or device logs are monitored [5], while the Network Intrusion Detection System (NIDS) controls different types of traffic including Internet Protocol (IP) addresses, firewalls, logs, and network ports [8,9]. The security setups will focus on the whole network to detect various forms of abnormalities. Generally, enterprises use their cloud networks, scan their architecture, and analyze intrusions to decide whether or not the traffic is normal or malicious. This protection protocol is most useful in terms of securing cloud networks from possible threats [10]. We have opted for a cloud-based ML optimization technique using a novel AIDS dataset [7], which incorporates 14 attributes to achieve an efficient classification accuracy from the range of 90% to 99%. The objective of this research is to categorize and come across 0-day threats integrated into the UGRansome1819 dataset by using various ML algorithms and two cloud services. The UGRansome1819 dataset has three predictive classes of threatening behavior which include Anomaly (A), Synthetic Signature (SS), and Signature (S) threats [7]. This dataset used the RF, NB, SVM, Ensemble Learning model, and GA with a mixture of all the sub-cited ML algorithms to get the surest evaluation results of the AIDS dataset. The interested reader should refer to Nkongolo et al. [7] for the production methodology of the UGRansome1819 dataset. However, the feature selection used in the construction of UGRansome1819 utilized Principal Component Analysis on the UGR'16 and ransomware datasets to detect the most relevant features. Next, these features were manually merged and combined using a fuzzy merging technique to build the final dataset. The fuzzy merging is similar to an intuitionistic data fusion technique where data retrieved from various sources are mixed to come up with one less redundant dataset. Only the features that shared the same characteristics were combined and categorized into normal and abnormal classes [7], but some updated and highly cited datasets features such as CAIDA and Cambridge Lab were neglected. Choosing supervised algorithms for the sake of anomaly detection needs strong justification. Because supervised algorithms have better performance in terms of accuracy and a low rate of False Positive in detecting known attacks rather than zero-day ones. Anomaly detection can be deployed with supervised

ML models and automate the process of determining whether the data that is currently being observed differs from typical data observed historically. This goes beyond the simple threshold of data. Anomaly detection models can look across multiple sensor streams to identify multi-dimensional patterns over time that are not typically seen. The supervised learning model can recognize patterns in the dataset that indicate a high likelihood of 0-day threatening behavior such as cyclostationarity. Anomaly detection models can also alert the analyst to patterns that require closer investigation. This may force the engineer to perform a sanity check on the network to detect the threats. In the most efficient implementation with supervised learning, labeled anomalies are fed into the supervised learning model to enhance and expand the predictive models. A threat with similar data patterns as labeled anomalies might help in the detection of 0-day threats. Anomaly detection models coupled with supervised learning learn efficiently from any dataset where each feature will have its own normal or abnormal data patterns and it is still the gold standard of predictive and classification experiments. So, anomaly detection is an important complement to supervised learning. However, the challenges of labeling and collecting the necessary data limit the construction of a robust solution based on supervised learning alone.

1.1. Research Question

The foremost research question may be stated as follows:

- Is there any classification performance difference in terms of optimization using the UGRansome1819 dataset?

1.2. Aim and Objectives

The primary objective of the research is to detect potential 0-day (unknown) threats using ML algorithms and cloud services. To obtain the favored outcomes, the research study has the following targets:

1. Use SVM, NB, and RF classifiers to determine if the classifier's assumptions are exact. We have examined and trained those classifiers using SageMaker.
2. Compare the classifiers in terms of spotting 0-days threats and apply the GA on the UGRansome1819 dataset to extract salient features.
3. Use the GA optimizer with each classifier to evaluate the improved performance.
4. Use Ensemble Learning to combine the results of classifiers and determine if this method enhances the classification performance.
5. Lastly, compare the outcomes and the UGRansome1819 performance and complexity with existing datasets with the same algorithms and optimization settings by computing them on the CAIDA and UNSWNB-15 datasets.

Researchers in the IDS panorama are using legacy datasets that encompass obsolete network threats that have changed and developed. As a result, cutting-edge NIDS cannot discover novel network attacks. Additionally, 0-day threats datasets are not published to investigate their behavior [7]. Hence, our study contribution is the use of a novel anomaly detection dataset [7] that carries 0-day threats to train and test ML algorithms. We have used two cloud services such as S3 bucket and SageMaker to achieve this goal. This article is structured as follows: The research problem, aim, as well as research question was discussed in Section 1. Section 2 will concentrate on the literature review. In Sections 3 and 4, the research methodology and the proposed cloud optimization method are presented: we have also introduced a schematic representation of the cloud optimization method and discussed the computing environment that was utilized. Further, the results of the implementation before and after optimization are illustrated in Section 5 and the discussion in Section 6. Finally, the future works and conclusions are offered in Section 7.

2. Related Works

Daily, attackers update themselves as well as the technology they utilize to produce new threats [11,12]. Under this assumption, IDS are being created at a growing rate to

reinforce network systems to effectively combat evolved threats. Several studies have been undertaken for this purpose, and novel studies are being carried out each day to optimize the quality of the IDS. Intrusion is typically categorized as a form of incorrect motion that involves damage to a network [11]. Thus, when the network security, confidentiality, and accessibility of services are confronted; the event would be categorized as an intrusion. Actions that make network services unusable to utilize for humans are also declared as intrusion [11]. Signature-based IDS (SIDS) and AIDS are the two types of NIDS. The SIDS makes use of pattern recognition strategies to understand common threats. Similar strategies for SIDS can also discover the earlier threatening behavior. In some stages, an alert message is activated on each intrusion when a signature of an intrusion resembles a previous incursion that already existed within the signature database [13]. The literature review consists of some research related to intrusion and anomaly detection and the usage of datasets in the NIDS landscape. In Divekar et al. [14], a study was carried out on the usage of benchmarking datasets for AIDS using the KDD CUP 99 dataset and its contrast performance is discussed. While tested with K-Means, NB, RF, Decision Tree (DT), SVM, and Neural Networks (NN) or using the SMOTE oversampling approach and random below sampling, it was proved that unbalanced centered training in KDD-99 and NSL-KDD avoid the effectiveness of classifiers on minority intrusion (User to Root and Root to Local attacks), probably posing security troubles. The UNSW-NB15 dataset was investigated to overcome patterns dispersion limitations of the KDD-99, when equating the overall results of minority classes in the corpus after and before SMOTE oversampling, the research reveals that for binary classification, a sufficient performance was achieved in terms of F1-Score with the UNSW-NB15 dataset that is similar to the NSL-KDD and KDD-99 datasets, and Divekar et al. [14] presented the UNSW-NB15 as a palliative for legacy datasets. A thorough analysis of IDS was made by Ring et al. [15], the robustness of NIDS had been tested to evaluate the usability of datasets. The dataset from the Defense Advanced Research Projects Agency (DARPA) has been used for studies purposes over the past years. This is a four-gigabyte collection transformed into binary and containing simulated data packets with seven million network patterns [16]. However, the dataset is out of date. The KDD99 dataset is constituted of forty-nine million hyperlink characteristics having forty-one parameters, and it is been acquired from the DARPA98 dataset. Such features have been categorized as either network data vulnerabilities or in any other case. The DARPA dataset has analytical problems, making it unable to correctly discover malware with excessive precision. The Network Security area became advocated to assemble the NSL-KDD dataset as an upgraded model of the DARPA and KDD99 datasets because of various drawbacks. When compared to the KDD98, KDD99, NSL-KDD, and DARPA datasets, the NSL-KDD dataset attains fewer redundancy [10]. Over the last few years, researchers have proposed different AIDS solutions including fog and parallel computing amongst many others. AIDS has been utilized in a wide range of fields, not just in Network Security, but also in Network Traffic Management [17–19]. Extensive research showcasing novel anomalous intrusion is presented by Yu et al. [20].

2.1. Anomaly Intrusion Detection System (AIDS)

The key to AIDS has been the ability to discover 0-day threats. Whenever the examined traffic varies from ordinary, the AIDS will send out an alert message. AIDS offers several advantages. To start with, if the computer system has been attacked, this will raise an alarm if such a threat seems to be identifiable [21]. Furthermore, there are many principal advantages of using AIDS that help protect the network from potential cyber-attacks. It can supply an intrusion signature [22] that facilitates the recognition of network attacks. Lawal et al. [22] used a unique hybrid anomaly method based on fog computing for IoT networks. The anomaly and signature-based detection approach was used. The paper furnished a comparative analysis of anomaly detection solutions within the IoT. Shapoorifard and Shamsinejad [23] reported anomalies or undesirable entry in the IoT object that process data. In Software-Defined Networking (SDN), numerous approaches had been used to

perceive network attacks. The two critical standards to discover intrusions are a deviation from ordinary network traffic in addition to a low or high traffic flow [7]. Normally, IDS continuously analyzes network traffic or time-series data and becomes computationally expensive. With the advent of 5G, the accuracy of network attacks detection has become a serious concern. SDN centralized networks and allows the examination of their traffic. As, such, the time complexity for processing data in SDN has to be considered due to the delay that could affect the execution time.

2.2. UGRansome1819 Dataset

This dataset is created by getting critical features of the UGR'16 and ransomware datasets [7,24]. It is an anomaly detection dataset that consists of both normal and anomalous network activities. The regular characteristic collection makes up 41% of the dataset, while irregularity makes up 44% and the prediction is 15%. The dataset has been made publicly available in [7]. Regular class threats such as User Datagram Protocol (UDP) Scan and Bonet provide approximately 9% as well as anomalous category threats [7]. IP addresses represent only 18% of the dataset with 19% of connection signals. A percentage distinction exists between network protocol (14%) and seed or expended addresses (16%) [7]. According to Nkongolo et al. [7], a substantial proportion of the dataset can be categorized by the following factors:

- The Transmission Control Protocol (TCP) carries the most characteristics in terms of network protocols (92,157).
- The AF flag has the most features (72,814).
- Locky has the most features in terms of ransomware family (33,870).
- Addressing of class 1DA11mPS has the most properties (82,048).
- In comparison to classes B, A, and D, class C of IP addresses has much more characteristics (95,508).
- The Secure Shell (SSH) threat has the most attributes (34,972) compared to UDP Scan, Blacklisted, Spamming, DoS, Scan, and Bonet threats.
- The Signature (S) threat category has by far the most properties compared to the Synthetic Signature (SS), and Anomaly (A) threats predictions.

All feature attributes of the UGRansome1819 can be summed up into the following prediction categories [7]:

1. The A category represents unknown threats without signature keys. This prediction includes the most 0-day threats.
2. The SS category includes unknown and known threats with and without signature keys. This prediction exhibits both, regularity and irregularity.
3. The S category depicts well-known threats with updated signature keys. This prediction portrays regularity or normality.

2.3. Support Vector Machine Algorithm

The SVM utilizes a multi-dimensional space configuration to stratify one category from the other [25]. It is a technique that uses a kernel as a function that can be polynomial, radial basis, linear, or other. The SVM properties determine a hyperplane given in Equation (1). This function maps the original input space in a new space with a precise classification rate and accurately distinguishes two categories with a maximum margin. That said, the performance of SVM is determined by the type of function its kernel utilizes:

$$g(x) = W^T x + b. \quad (1)$$

Computationally speaking, x_i is a set of data points belonging to a binary category. The separability of observations from the hyperplane is equivalent to Equation (2):

$$\frac{g(x)}{\|w\|}. \quad (2)$$

SVM attempts to discover the weights w and bias b , for inputs $g(x) = 1$ given the nearest x_i from the binary category. This can be represented by the following margin given in Equation (3):

$$\frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|}. \quad (3)$$

This algorithm has been utilized by Dong [26] to resolve the multi-class classification problem for Network Traffic Management. However, the RF, NB, GA, and Ensemble Learning algorithms were not used to reduce the computational time and enhance the classification accuracy to resolve the imbalance issue when compared to the SVM.

2.4. Naive Bayes Algorithm

The NB uses Bayes' theorem and posits "naive" independence between features given a specific dataset. X is a set with n features to be stratified. This can be mathematically written by using a vector $X = (x_1, \dots, x_n)$. This algorithm implements the following probability (P) computation to specify the category C_k given x :

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}, \quad (4)$$

The following equation is used to assign the class for X :

$$y = P(C_k) \sum_{i=1}^n P(X_i|C_k) \quad (5)$$

where the predicted label is denoted by y and the probability by P . The NB algorithm was used with Laplace smoothing to help tackle the problem of zero probability. While testing the NB algorithm, the ML model could face a new observation \hat{x} for which it was not trained on. Then, P would be null for \hat{x} representing a specific data point. The Laplace smoothing is then used to solve this problem by ensuring that P will never be zero for \hat{x} . It uses a smoothing parameter α that is not equal to zero:

$$P(C_k|\hat{x}) = \frac{\sum_{k=1}^K \hat{x}_k * y_k + \alpha}{N + \alpha * K}, \quad (6)$$

K represents the number of features in the dataset and N is the number of observations on which the NB was trained. NB algorithm has been used by Guezzaz et al. [27] to evaluate the probability of specific categories and enhance the classification model for the Network Intrusion Detection Problem (NIDP). Nevertheless, in the worst-case scenario, the training phase of the NB takes a similar amount of time as the testing phase [28] and Guezzaz et al. [27] did not come up with an ensemble or optimization technique to improve the classification model.

2.5. Random Forest

The RF is a classification tree and pattern recognition algorithm [29]. With RF, the trees are grown adaptively to remove bias. The intuition in RF is to enhance the variance by reducing the correlation between various trees, without increasing the variance. This is computed in the tree-growing phase via random selection of the inputs (Algorithm 1). RF was used by Disha and Waheed [30] for the NIDP. The algorithm creates a tree with different training sets from a given dataset. The RF is used by Disha and Waheed [30] and performed well on the UNSW-NB 15 dataset using Gini impurity as the splitting parameter of trees that adjusted weights of the binary classification. The RF was then utilized to assess features of the most significant relevance and did not have any problem with overfitting or numeric variables [31]. However, the GA was not used to tackle the feature extraction problem of imbalanced datasets.

Algorithm 1 Random Forest Algorithm**Require:** Features sampled f_s **Ensure:** $b = 1$ to B Draw a bootstrap sample Z of size N from f_s Draw a Random Forest tree T_n to bootstrapped dataRecursively repeat the process for each node in T_n Stop when the minimum size n_{min} is reached**while** $f_s \neq 0$ **do** **if** $N \neq 0$ **then** Select m variables at random from the p variables Pick the best split point among the m ▷ or pick the best variables

Split the node into two daughter nodes

 Output the ensemble of trees T_b^B **else if** $N > 0$ **then** Make a prediction at a new point x Regression: $f_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$ Classification: Let $C_b(x)$ be the class prediction of the Random Forest tree $C_{rf}^B(x) = \text{Majority vote } [C_{rf}^B]_1^B$ *2.6. Ensemble Model Technique*

The merit of ML classifiers is to extract meaningful patterns from a given dataset. The Ensemble Learning model of classification combines different classifiers and averages the accuracy to provide a high classification rate having a low false classification ratio. Ensembling refers to a particular method of training a model of weak classifiers. The model classifiers are in the form:

$$F_T(x) = \sum_{t=1}^T f_t(x), \quad (7)$$

each f_t is a weak classifier that takes an observation x as input and returns a value indicating the class of the observation. The T th sample classifier is positive if the sample is in a positive cluster and negative otherwise. Each weak classifier produces an output $h(x_i)$ or a hypothesis, for each sample in the training set. At each t th iteration, a weak classifier is selected and assigned a coefficient α_t such that the sum of the empirical or training error E_t of the resulting ensemble classifiers is optimized:

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t h(x_i)], \quad (8)$$

$F_{t-1}(x)$ is the ensemble classifiers that have been built up to the previous stage of training, $E(F)$ is the error function and

$$f_t(x) = \alpha_t h(x), \quad (9)$$

is the weak learner that is being considered for addition to the final classifier. Abu Al-Haija and Al-Badawi [32] used an ensemble method on the NSL-KDD dataset in order to implement a NIDS with quick model building time, minimal false positive rate, and high classification accuracy [33]. However, the ensemble model was not used with GA to produce a lower error and better accuracy rates.

2.7. Genetic Algorithm

The genetic optimization approach is a biologically inspired optimization paradigm that utilizes evolutionary optimization techniques such as transmission, crossovers, and variation [34]. The critical steps of the GA can be listed as follows [34]:

- Use the initial dataset (population) to search for a possible solution s . The algorithm starts with a population that can be thought of as a set of individuals. Each individual

represents a solution to the optimization problem. The solution is characterized by a set of variables (parameters) called “genes” that are merged into a string to produce a solution (chromosome). We have used ternary values (0, 1, and 2) to represent a set of genes. This can be viewed as encoding genes into a chromosome.

- Calculate the fitness function. It determines how to fit a solution by comparing it to other solutions and providing a fitness score for each solution. The probability that a solution will be selected is based on the fitness score. The probability of selecting a solution is denoted by $P[s]$. We posit that α and σ are constant:

$$P[s] = \alpha + \sigma s. \quad (10)$$

- Selection. This phase selects the fittest solutions and lets them pass their genes to the next iteration. This is like how it is the fittest individuals that pass their genes to the next generation in Darwinism. The parents of individuals are selected using the fitness scores. To be selected for reproduction, individuals should have high fitness. It utilizes a probability distribution for selection where the given string should have a selection probability proportional to its fitness. The mathematical formulation of the selection process probability (*Prob.*) is:

$$s = \frac{\text{Prob} * [\text{fittest string selection}]}{\text{Prob} * [\text{average string selection}]}. \quad (11)$$

- Compute the crossover process. A crossover point is selected for each pair of parents to be mated. This is a random process within genes where offspring are created by merging the parents’ genes among themselves to reach a crossover point and new offspring are included in the population. Crossover replaces some parent’s genes by corresponding them with the genes of other parents. If we have two strings x and y , each having three variables:

$$(x_1, x_2, x_3), (y_1, y_2, y_3).$$

This represents possible solutions to the classification problem. Two cross-points are randomly selected and a new possible solution is produced by merging the string of the original parents. For example, the offspring solution would be

$$(x_1, y_2, y_3), (y_1, x_2, x_3).$$

- Implement the mutation process. Genes of offspring formed are subjected to a mutation using a minimalistic random probability: String’s bits are flipped to maintain the population diversity. The flip mutation selects random bits and flips them differently. The bold indicates the permutation or the bytes that have been changed or flipped during the process:

$$100101011 \longleftrightarrow 110101001,$$

$$10 \mathbf{0} 101011 \longleftrightarrow 11 \mathbf{1} 001011,$$

$$10 \mathbf{0} 101011 \longleftrightarrow 110001011.$$

When the population convergence is achieved, the GA terminates and provides a set of solutions to the optimization problem.

As a result, the GA establishes a classification model using the aforementioned approaches to select appropriate variables for the detection and recognition of anomaly [35]. Additionally, the GA simplifies the process of feature selection by using tuning parameters because high False Positive rates might bias the accuracy. Maseer et al. [25] analyzed past AIDS studies that utilise NIDS datasets. The results of ML-AIDS algorithms in identifying network threats are also provided. The algorithms present demerits in detecting novel and 0-day

threats [31] for the multi-classification issue. To overcome this problem, Yihunie et al. [36] provided a benchmarking approach that includes multiple classifiers using real datasets to enable accurate evaluation of AIDS. The experiments reveal that a single ML algorithm cannot detect all types of network attacks (known and unknown). Nevertheless, due to strong False Negative and False Positive alerts, the NB-AIDS, KNN-AIDS, and DT-AIDS simulations performed well but the EM-AIDS and SOM-AIDS algorithms perform badly.

2.8. Summary of Related Works

A comparative analysis of related works is given in Table 1 to summarise the literature review. Most of the research papers discussed in the current literature [37–41] did not solved the imbalance, optimization and multi-class issues.

Table 1. A Comparative Analysis of related Works.

Author	Dataset	Approach	Limitations
[37]	CAIDA	SMO	Anomaly detection
[38]	CICIDS2017	Deep Neural Network	IoT protection
[39]	NF-BoT-IoT-v2	Random Forest	Imbalance data
[40]	UNSWNB-15	GWO	Multi-classification
[41]	WSN-DS	KNN	Optimization

The literature review describes major research for the NIDP and the most used research methodologies such as Deep Neural Network, Random Forest, and clustering. The limitations of the discussed research papers include the use of legacy datasets, as much work has been conducted to detect and classify legacy network threats incorporated in these datasets. As such, novel network attacks such as 0-day threats should be the focus of future research work as suggested by Hindy et al. [10]. A similar recommendation is provided by Nkongolo et al. [7] in terms of more AIDS research that will concentrate on 0-day threats recognition and optimization. In this research, a comparison with other state-of-the-art models is achieved. We have tabulated the results obtained using the UGRansome1819 by comparing its performance and complexity with existing models. We have used the same algorithms (SVM, NB, and RF) with the optimization settings (Ensemble Learning and Genetic Algorithm) and computed them on the CAIDA and UNSWNB-15 datasets.

3. Materials and Methods

Many researchers apply various ML algorithms on NIDS datasets to evaluate trained algorithms, however, most of the time they do not succeed in accurately testing and training these algorithms to reduce false alarms [42]. There are many reasons for this failure such as data preprocessing, tuples selection, missing values occurrence, and anomaly detection that affect the result of a classifier [34,42]. Selecting all features may also cause a minimization in the assessment of a classifier because of overfitting problems [34]. ML algorithms might also be computationally expensive and take more time to execute if there are minimal features. To solve the execution and overfitting issues, it is crucial to select relevant and important features attributes from the dataset for accurate evaluation [42,43]. Different methodologies such as variance Inflation Factor, Particle Swarm Optimization (PSO), Stochastic Gradient Descent, and GA are utilized to extract and select important features from a dataset [44]. In our research, a GA is used to extract eight important features from the UGRansome1819 dataset.

3.1. The Proposed Model

The proposed methodology aims to optimize the classification of important features in the UGRansome1819 using GA and Ensemble Learning (Figure 1). The optimization with GA coupled with the DT is utilized as a feature selection that extracts relevant features from

the UGRansome1819. The NB, SVM, and RF algorithms are then applied to the dataset but assessed through the Recall, Accuracy, Confusion Matrix, and Precision evaluation metrics (Figure 1). These ML classifiers are also computed on the same dataset using the Ensemble Learning technique (Figure 1). The stacking Ensemble Learning technique is applied to the SVM, RF, and NB classifiers. The RF and SVM classifiers are utilized as member models while the NB has been used as the base model. In turn, the GA is then executed on the dataset to extract relevant features. After this feature extraction process, all the three selected ML classifiers along with the Ensemble Learning model are again evaluated and trained on the newly optimized feature set. The classification results of the ML algorithms are compared and assessed with each other. The UGRansome1819 dataset is subdivided into three classes used for prediction: S, A, and SS. Similarly, these classes include 16 attacks that represent abnormality of 0-day threats: Globe, SamSam, Flyper, DMALocker, NoobCrypt, CryptoLocker, CryptoLocker2015, CryptXXX, Cryptohitman, JigSaw, EDA2, Globev3, TowerWeb, APT, WannaCry, and Locky. Instances of normality correspond to nine well-known threats such as Bonet, DoS, Spam, SSH, Scan, Blacklist, Port Scanning, Nerisbonet, and UDP Scan [7].

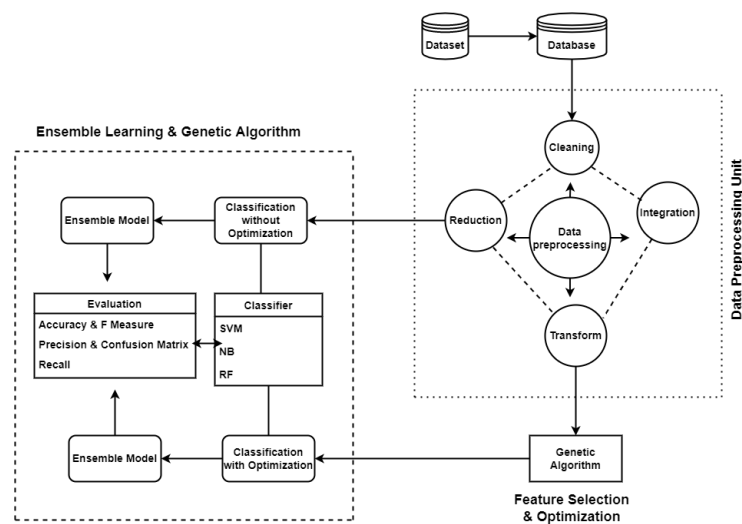


Figure 1. Proposed Methodology.

3.2. Data Pre-Processing

We have used the multi-class classification model to pre-process the UGRansome1819 CSV file with the Python programming language that was utilized for the implementation of SageMaker and S3 bucket. The first service enabled the implementation of ML in the cloud while the second assisted in storing the dataset in a cloud database. To simplify the pre-processing, each categorical feature was converted to numeric.

3.3. Specifications of Software and Hardware

The implementation of the ML classifiers is performed on the cloud using SageMaker with a laptop having the Linux Operating System (OS) installed. The specification of the employed computational environment is mentioned in Table 2.

The Python programming language is utilized to code the ML classifiers. It is widely utilized to classify, predict, and analyze quantitative and qualitative data. The predictive models are implemented with Seaborn, NumPy, Pandas, Sklearn, and Matplotlib libraries. The three selected classifiers are computed on the same dataset before and after GA.

3.4. Cloud Services

The Amazon Web Service (AWS) SageMaker is used as a cloud ML framework that assists in training, designing, executing, and optimizing ML classifiers as described by Zhang et al. [45]. The advantages of utilizing AWS SageMaker are as follows [45]:

(i) Data protection, (ii) fast and adaptable training, and (iii) continuous and enhanced operating process.

Table 2. The Computing Environment.

Component	Specification
Dataset	UGRansome1819
Processor	2.60 GHz and 2.59 GHz
System Type	64-bit
Programming Language	Python
CPU	Intel (R) Core (TM) i7-10 750
OS	Debian 11
Optimization	GA and Ensemble Learning
Cloud Services	S3 bucket and SageMaker
RAM	40 GB

4. Implementation

The suggested cloud-based optimization method execution has been supplied in Figure 2. Implementation for this research requires the Amazon Web Services (AWS) which provides various technologies for distributed services [46]. One of these services is the S3 bucket which supports object storage via an interface. The Amazon S3 bucket uses the same storage environment that Amazon.com utilizes to run its e-commerce infrastructure [46]. The Amazon S3 URL was used to store, archive, and backup the UGRansome1819 for cloud analytics (Figure 2). The AWS SageMaker assisted in building, preparing, deploying, and training the three ML algorithms [46]. We have used the UGRansome1819 dataset to compare the performance of selected classifiers that detect 0-days threats. This article offers a multi-class model approach that utilizes an optimization algorithm named GA and Ensemble Learning to discover 0-day threats from the dataset with the combination of the three algorithms. The features in the UGRansome1819 dataset have been normalized to make it more balanced in terms of normal and anomalous data [47]. The UGRansome1819 has been limited to 207,533 observations or tuples with 14 attributes or columns subdivided into 80% of training data with 20% of testing data.

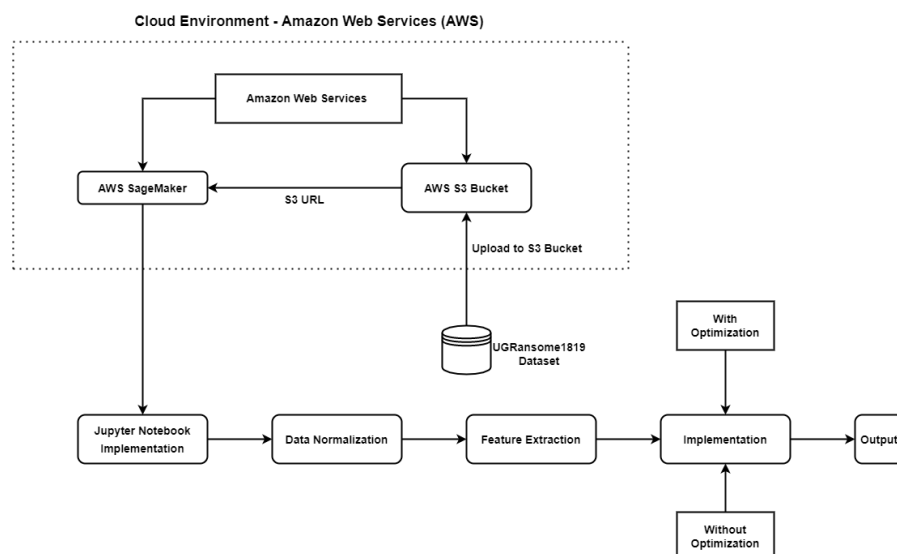


Figure 2. The Cloud Based Optimisation Method Execution.

4.1. The UGRansome1819 Dataset

Figure 3 represents the class labels of the prediction column in the dataset [7].

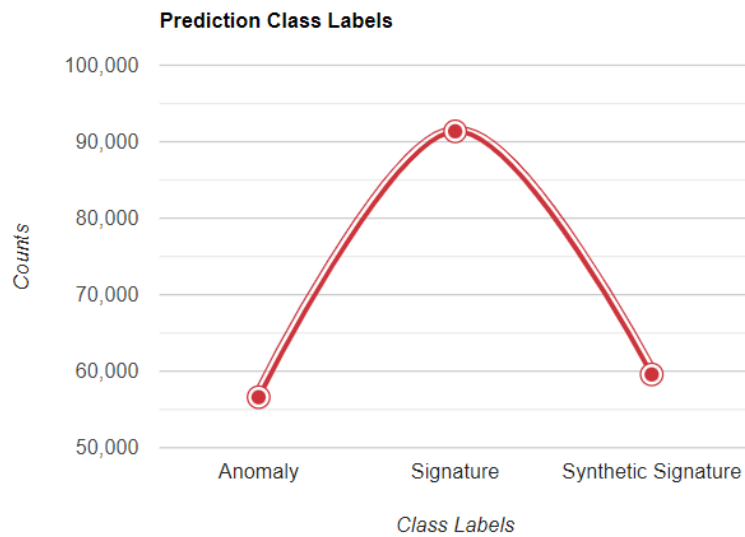


Figure 3. The Prediction Class Labels.

The ratio of the three-class labels is 56,598 counts for Anomaly, 91,360 for Signature, and 59,575 for Synthetic Signature attacks (Figure 3). This dataset is obtained from [7] and contains 0-day threats for AIDS. The 207,533 tuples with 14 attributes are depicted in Figure 4. The UGRansome1819 consists of both numeric and textual data: Six features are numeric while the remaining eight features are categorical. Various features such as Prediction, Port, Threats, IP address, Netflow, Dollars (USD), Bitcoins (BTC), Expanded address, Seed address, Clusters, Ransomware Family, Flag, Protocol, and Timestamp are recorded. The prediction column representing the three-class labels (A, S, and SS) is illustrated in Figure 3. S stands for Signature, A for Anomaly, and SS stands for Synthetic Signature in the dataset (the last columns in Figure 4). All patterns categories of the UGRansome have been stratified as follows:

	50	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1.1	500	5	A.1	Bonet	5061	SS
0	40	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	504	8	A	Bonet	5061	SS
1	30	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	508	7	A	Bonet	5061	SS
2	20	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	512	15	A	Bonet	5061	SS
3	57	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	516	9	A	Bonet	5061	SS
4	41	TCP	A	WannaCry	1	1DA11mPS	1BonuSr7	1	520	17	A	Bonet	5061	SS
...
207528	12	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1964	2986	6081	A	UDP Scan	5062	A
207529	8	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1968	2992	6092	A	UDP Scan	5062	A
207530	8	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1972	2998	6103	A	UDP Scan	5062	A
207531	8	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1976	3004	6114	A	UDP Scan	5062	A
207532	8	TCP	APSF	Flyper	8	1AEoiHYZ	1SYSTEMQ	1980	3010	6125	A	UDP Scan	5062	A

207533 rows × 14 columns

Figure 4. The Dataset Description.

1. The category of Synthetic and Signature malware (SS). It represents characteristics of well-known as well as unknown attacks. This category exhibits both, normality and abnormality.
2. The category of Signature malware (S). It depicts well-known malware having available keys that have been released and updated regularly. The signature category portrays normality.

3. The category of anomaly malware (A). It is a set of unknown malware for which signatures or keys do not yet exist. Abnormality is illustrated by this category.

Structure of the Dataset

The timestamp is just a numerical value expressing the time spent by the threatening behavior [48], the network protocol is a categorical communication protocol such as TCP or Internet Control Message Protocol (ICMP) [49]. The network flag is a categorical representation of the network status (for instance, APSF) [7]. The ransomware family is the column containing the ransomware-type (e.g., EDA2, Flyper, and WannaCry) [24], it is a categorical column (Figure 4). Every network threat is allocated to a numerical cluster, which is a numerical feature. The seed (1AEoiHYZ) and extended (1SYSTEMQ) addresses were converted during the features normalization phase; it is a categorical column that is used for malicious activities (receive and transfer BTC or USD) [24]. The financial damage caused by network threats is quantified in USD and BTC [24]. The network flow (Net-flow) is a numerical representation of the network traffic where the threatening behavior occurred [7,50]. The victimized host computer is represented by a unique categorical IP address. It was a normal address such as 198.169.18.19 that was obfuscated into classes A, B, C, and D for privacy reasons [7]. Network threats are categorical attributes such as Bonet, DoS, UDP Scan, SSH, and Spamming. The network port is numerical: It is the port number used by a threatening behavior. The prediction is categorical: It denotes the stratification scheme that the ML algorithm will utilize to predict features into A, S, and SS [7]. The construction methodology of the UGRansome1819 is discussed by Nkongolo et al. [7]. The data structure of the UGRansome1819 is presented in Table 3.

Table 3. The UGRansome1819 Data Structure.

Abnormal Threats	Example	Prediction	Total
DMALocker	Port Scanning	Anomaly	12,371
Globe	UDP Scan	Anomaly	13,200
CryptoLocker Types	Blacklist	Signature	20,527
JigSaw	Scan	Anomaly	17,544
SamSam	Spam	Anomaly	28,597
TowerWeb	Spam	Anomaly	7353
Flyper	Nerisbonet	Synthetic Signature	14,352
WannaCry	Bonet	Synthetic Signature	22,931
APT	DoS	Synthetic Signature	15,363
Locky	DoS	Synthetic Signature	33,870
Razy	Scan	Anomaly	12,535
EDA2	Blacklist	Signature	8744
Globev3	Spam	Anomaly	146

4.2. Pre-Processing

Data preprocessing plays a crucial role in ML classification. If the dataset is not well processed the classifier may not correctly work as expected [51]. It might be due to tuples that have missing values, outliers, and inconsistency. Therefore, it is also crucial to pre-process the dataset to make it suitable for ML computation [34,51]. There are no missing values in the UGRansome1819 dataset and all categorical features are translated to a numerical form using the Python label encoder function. This function assigns a numerical value to each category starting from zero. For instance, the prediction class labels having three distinct category (A, S, and SS) is transformed by the label encoder: 2

is assigned to SS, 1 to S, and 0 to A. All the categorical features presented in Figure 4 are similarly transformed into a numerical form. The new pre-processed UGRansome1819 contains only numerical features and is recorded for further AIDS research (Figure 5).

	1	2	3	4	5	6	7	8	9	10	11	12	13	output
0	40	1	0	16	1	2	2	1	504	8	0	1	5061	2
1	30	1	0	16	1	2	2	1	508	7	0	1	5061	2
2	20	1	0	16	1	2	2	1	512	15	0	1	5061	2
3	57	1	0	16	1	2	2	1	516	9	0	1	5061	2
4	41	1	0	16	1	2	2	1	520	17	0	1	5061	2
5	22	1	0	16	1	2	2	1	524	11	0	1	5061	2
6	18	1	0	16	1	2	2	1	528	19	0	1	5061	2
7	3	1	0	16	1	2	2	1	532	13	0	1	5061	2
8	26	1	0	16	1	2	2	1	536	21	0	1	5061	2
9	22	1	0	16	1	2	2	1	540	15	0	1	5061	2
10	7	1	0	16	1	2	2	1	544	23	0	1	5061	2

Figure 5. The Numerical Format of the Dataset.

4.3. Evaluation of the Machine Learning Method

To quantify the performance of the ML, data scientists train a predictive model on a specific amount of data [52]. The training requires feeding the features set with class label results to the ML classifier [4]. The model is assessed after training by using the holdout feature without the class labels results to predict the new output. Generally, the evaluation metrics of the ML classifier compare the actual class labels with the predicted class label result and quantitatively evaluate the ML predictive model [4,34,53]. In this research, five evaluation metrics are used: Confusion Matrix, Accuracy, Recall, Precision, and F1-Score. TP is the True Positive representing an outcome where the model correctly predicts the positive class. FP is the False Positive representing an outcome where the model incorrectly predicts the positive class. It is also named type-1 error [43]. FN is the False Negative representing an outcome where the model incorrectly predicts the negative class. It is also named type-2 error. TN is the True Negative representing an outcome where the model correctly predicts the negative class. The Precision refers to correctly predicted positive observations [43]. The Recall value specifies the percentage of some class correctly identified from all of the given examples [54]. The Accuracy represents the proportion of correct predictions in all predictions made [43,54]. It is a measure of the performance of our ML model. The F1-Score is the mean of Recall and Precision. The mathematical formulations of evaluation metrics are presented in Table 4. The Confusion Matrix is a n*n table that allows visualization of the performance of an algorithm where the matrix row represents instances in an actual class and each matrix column represents instances in a predicted class [4,55]. It calculates the evaluation metrics such as Accuracy, Precision, and Recall (Figure 6). In Figure 6, FN and FP represent misclassified data while TP and TN represent accurate classification [7].

Table 4. The evaluation metrics.

Accuracy	Precision	Recall	F1-Score
$A = \frac{TP+TN}{TP+TN+FP+FN}$	$P = \frac{TP}{TP+FP}$	$R = \frac{TP}{TP+FN}$	$F = \frac{2*P*R}{P+R}$

TN	FP
FN	TP

Figure 6. The Confusion Matrix.

5. Results

5.1. Classification

We present the results of the classification model before and after optimization. Before the optimization, the selected ML classifiers are computed individually with the stacking ensemble technique, and their performance is assessed. For all classifiers, the UGRansome1819 is divided into equal ratios of 80% and 20%. Each algorithm is trained with 80% of the dataset and tested on the remaining 20%. The evaluation metrics discussed in Section 4 are assessed for each ML classifier.

5.2. Without Optimisation

The SVM is applied on the UGRansome1819 to predict the most dominant class label. However, it did not obtain satisfying results. The SVM algorithm achieved 67.7% of Accuracy. The specification of the SVM is depicted in Table 5.

Table 5. The Specification of the SVM.

Parameter	Shape
Test set	20%
Random state	42
Train set	80%
Algorithm	Linear SVC

The SVM model accuracy is presented in Figure 7.

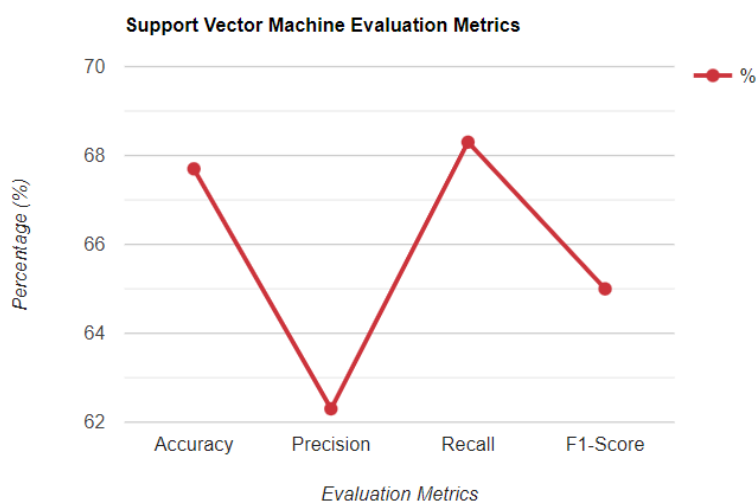


Figure 7. The SVM Classifier Prior Optimization.

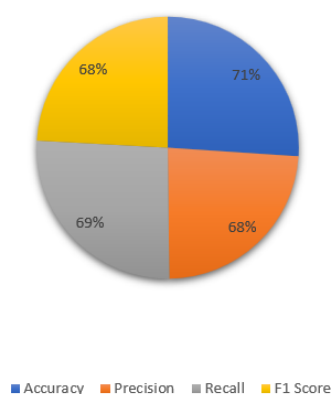
The algorithm is trained by feeding the features with the class label output. After this training, the testing data are provided to the SVM model without the class labels output. The classifier predicts the class labels output for the testing data in this manner. A comparison of the predicted data with the actual class labels output of the testing dataset is then evaluated. The Figure 7 portrays the SVM accuracy rate without optimization. This figure portrays the observed instability of the SVM classifier. The classification could not demonstrate stable predictions of a single classifier using SVM. An optimized validation technique will aggregate the weak classifier (SVM) into an ensemble and genetic optimizer to improve the performance. The NB algorithm outperformed the SVM and achieved an accuracy of 71%. Table 6 illustrates the NB specification model.

Table 6. The Specifications Model of NB.

Parameter	Shape
Test set	20%
Random state	42
Train set	80%
Algorithm	Gaussian NB

The same dataset is used to train and test the NB algorithm. Similarly, the evaluation metrics values obtained are not impressive. Figure 8 presents the accuracy ratio of the NB. This figure confirms that the Laplace smoothing help tackles the problem of zero probability and can be used to enhance single classification. The Laplace smoothing improved the NB performance compared to SVM by optimizing the classification probability. The optimized validation technique that aggregates the weak classifiers into an ensemble and genetic optimizer to improve the performance will include both SVM and NB.

NB Evaluation Metrics

**Figure 8.** The Naive Bayes Evaluation Metrics.

The RF uses Ensemble Learning to combine multiple algorithms and enhance the classification model [56]. In the enhancement of Accuracy and the other evaluation metrics, the number of trees plays a crucial role [56]. The higher the number of trees, the better the optimal values of assessment metrics can be obtained. We have used the Confusion Matrix to obtain the presented results. However, the stability of the model improvement was achieved only with 100 trees. It was useful to use this number during the training phase. If this number increased; the evaluation metrics values also improved. Details of the RF specification are shown in Table 7.

Table 7. Specifications Model of RF.

Parameter	Shape
Test set	20%
Random state	42
Train set	80%
Algorithm	RF
Trees	100

Figure 9 shows the evaluation metrics of the RF classifier. We have trained this model to compare results obtained by Nkongolo et al. [7] who achieved a similar Accuracy rate using RStudio. Our RF model achieved an Accuracy of 99% as also with the ensemble model. Figure 9 represents the accuracy ratio of the RF algorithm. The random state in the ensemble model technique is 42. This number has been utilized to configure the core random numbers generator, that controls the division of the UGRansome1819 into testing and training sets. The test classification accuracy of the RF model was 99.6% (before optimization). An improvement was not obtained after optimization because the classification accuracy remained 99.5%. The sensitivity and specificity reached 100%. The three most important predictors of 0-day threats were the Signature, Synthetic Signature, and Anomaly category of threatening behavior. The Table 8 illustrates the ensemble specification model.

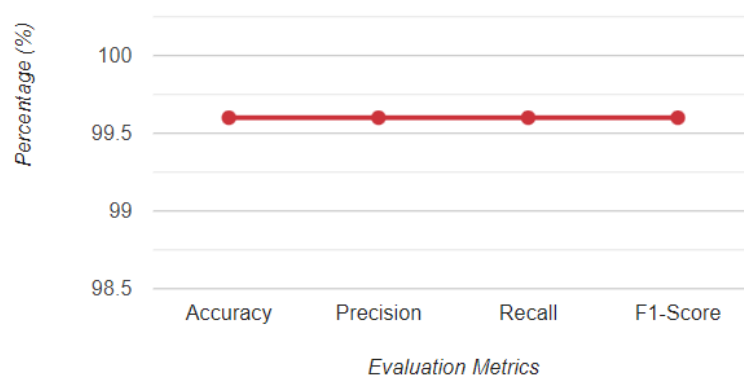


Figure 9. The Random Forest Classifier Model Performance.

Table 8. Ensemble Model Specifications.

Parameter	Shape
Test set	20%
Base model	NB
Train set	80%
Random state	42
Member model 1	SVM
Member model 2	RF
Algorithm	Stacking

This model produces impressive results when the members and base models such as SVM and NB do not have substantial results (Figures 7 and 8). The outcome of ensemble learning produces an efficient classification with a combination of weak classifiers such as SVM and NB. As a result, the model classification improved by achieving 100% of Precision and F1-Score with 99% of Recall and Accuracy. In this research, the RF algorithm performance was impressive and achieved sufficient results as in [7] and the Ensemble Learning also attained almost similar results. Figure 10 shows the accuracy ratio of the Ensemble Learning. Figure 10 has the overall results of the three selected ML classifiers using Ensemble Learning. All four evaluation metrics of all algorithms are illustrated. The figure shows a maximal Precision value of all three classifiers among the metrics.

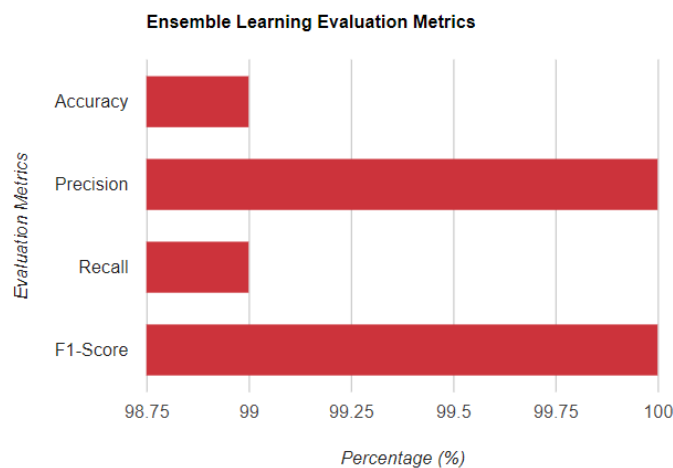


Figure 10. The Ensemble Learning Evaluation Metrics.

Overall, the SVM and NB classifier does not perform very well. Ensemble Learning and RF have almost the same performance. A GA is an optimization technique used to extract relevant information from the UGRansome1819 dataset. It is a search-based algorithm that used the concept of genetics and natural selection [34]. In a GA, all possible solutions represent a population subset while a single possible solution to a problem is a chromosome. The key factor of a GA is the fitness function that collects the input features and generates possible output or solution for a given classification problem. In this research, the GA is coded to retrieve the top eight attributes (columns) from the UGRansome1819 (Figure 11). The critical steps of the GA can be listed as follows [34]:

- Use the initial dataset (population) to search
- Calculate the fitness function
- Assess the efficiency of a candidate solution using the fitness function
- Compute the crossover process
- Implement the mutation process

```
"estimator = DecisionTreeClassifier() #genetic estimator to start GA\n",
"model = GeneticSelectionCV(\n",
"    estimator, cv=5, verbose=0, \n",
"    scoring=\"accuracy\", max_features=8,\n",
"    n_population=100, crossover_proba=0.5,\n",
"    mutation_proba=0.2, n_generations=50,\n",
"    crossover_independent_proba=0.5,\n",
"    mutation_independent_proba=0.04,\n",
"    tournament_size=3, n_gen_no_change=10,\n",
"    caching=True, n_jobs=-1)\n",
"model = model.fit(X, y)\n",
"print('Features:', X.columns[model.support_])"
```

Figure 11. The GA Specification. The verbose is used to produce logging information details and the tournament to select individual attributes from a population dataset. The tournament is executed three times and the winner of each tournament having the best fitness is selected.

The algorithm extracted the eight attributes namely, “Flag”, “Timestamp”, “Ransomware Family”, “USD”, “IP address”, “Network Flow”, clusters, and “Port number”. The three ML classifiers along with the Ensemble Learning are again computed to the newly pre-processed UGRansome1819. The normalized dataset consists of all tuples as the original dataset, however, the 14 attributes have been reduced to nine. Dropping five attributes caused a decrease in the computational time and the over-fitting chances are also minimized because the UGRansome1819 has been reduced. If a model performance remains the same, then it is important to re-optimize the dataset as this will reduce over-fitting as well computational time. However, when a dataset is unbiased with a standard deviation approaching one or a mean of all columns within a common range; it is not a

good option to apply feature extraction. In the UGRansome1819, the standard deviation (std) ranges from two to 26,435 and the mean from two to 14,179 (Figure 12).

	50	1	1.1	500	5	5061
count	207533.000000	207533.000000	207533.000000	207533.000000	207533.000000	207533.000000
mean	21.520009	2.377930	35.497988	14179.514265	2283.817509	5064.014696
std	15.863390	2.883349	116.785406	26435.795386	2667.948833	2.722092
min	-10.000000	1.000000	1.000000	1.000000	1.000000	5061.000000

Figure 12. The Standard Deviation and the Mean of the UGRansome1819 Dataset.

It was important to normalize UGRansome1819 and optimize it using a GA. The GA shown in Figure 13 has been applied in the optimization of selected ML algorithms and it is posited that this will increase the chances of improving the classification accuracy rate.

```
%%time
import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split

from sklearn.ensemble import StackingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc, roc_auc_score

from genetic_selection import GeneticSelectionCV
from sklearn.tree import DecisionTreeClassifier
import numpy as np

import warnings
warnings.filterwarnings('ignore')

CPU times: user 158 µs, sys: 17 µs, total: 175 µs
Wall time: 184 µs

df = pd.read_csv(r's3://sagemaker-studio-h1yxcgdssqe/new_file_data.csv')
```

Figure 13. Genetic Algorithm Implementation Improving Classification.

5.3. With Optimization

Figure 14 shows evaluation metrics of the SVM algorithm after optimization. The SVM has an improved Accuracy of 72.4% which was 66% before optimization. The test classification accuracy of the SVM model improved by 6%. The NB classifier is applied to the pre-processed dataset. This algorithm performed well by improving the computational time with reduced features, the model has similar results as to prior the optimization. It is then a best practice to utilize the pre-processed dataset as it will optimize the computational time of the classifier. The Figure 15 mimics a decrease of one percent in the accuracy ratio of the NB classifier with optimization. The test classification accuracy of the NB model decreased by 1% after optimization. The straightforward way to improve the NB performance to increase its accuracy after optimization is to add more data samples to the training data. Doing so will add more details to the data and fine-tune the model can result in a more accurate and optimized performance. The RF outperformed all the ML classifiers used in this research. It achieved optimal accuracy before and after optimization. Almost similar results have been obtained without and with optimization. As such, the optimization of the RF classifier is also suitable as it minimizes the computational time and reduces overfitting chances. Figure 16 portrays the evaluation metrics of the RF classifier after optimization. It depicts an Accuracy of 99% with optimization. The test classification accuracy of the RF model did not improve (99.5% after optimization and 99.6% before optimization). The closer the decision is to the leave, the more noise we have in the decision. The entire dataset

is taken from the top to the bottom and divided into two parts. In the end, the training sample will only include a few samples. Pruning was performed to avoid the model from overfitting the data. This figure shows that pruning provides a more compact tree and a better model in terms of accuracy (before and after optimization). Figure 17 illustrates the evaluation metrics of the Ensemble Learning, it is an accuracy ratio of 98% of ensembling with an optimization that has been presented.

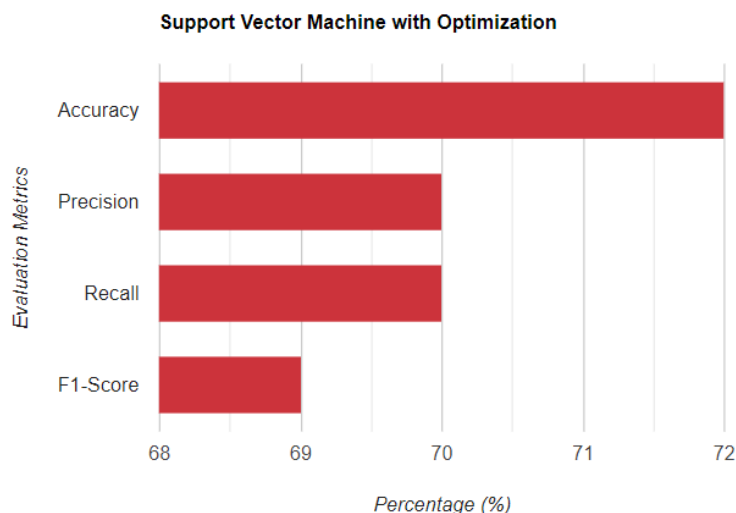


Figure 14. The SVM Classifier Performance After Optimization.

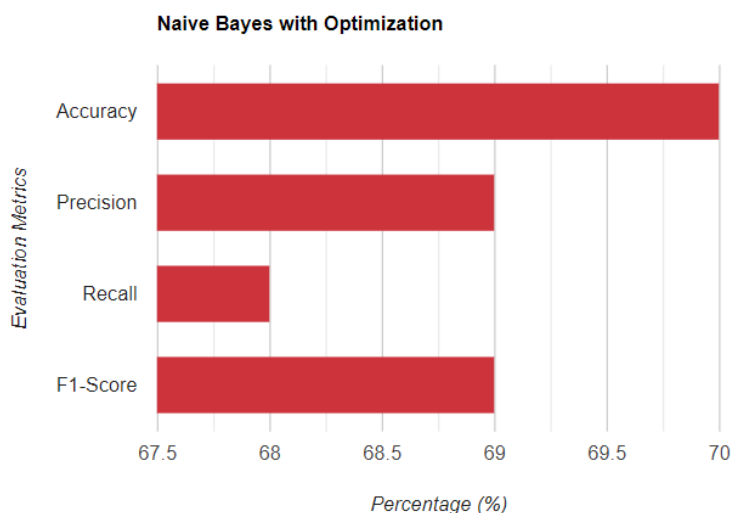


Figure 15. Naive Bayes Algorithm Performance After Optimization.

The same members and base models (SVM and NB) are used in the optimized validation model. The predictive capability of ensembling after optimization decreased by 1% for the Precision and F1-Score which was 100% before optimization. The Recall value remains at 99% (before and after optimization) while the Accuracy also decreased by 1% which was 99% before optimization. This result proves that the UGRansome1819 classification ratio before and after optimization is 1%.

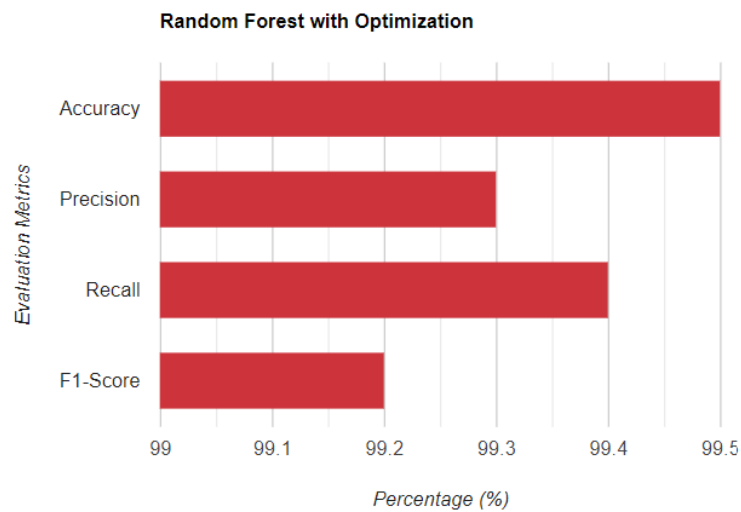


Figure 16. Random Forest Algorithm Performance After Optimization.

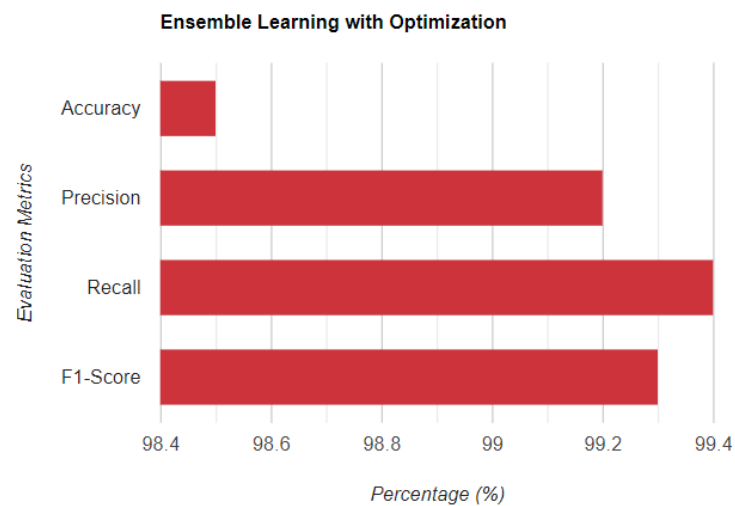


Figure 17. Ensemble Learning After Optimization.

6. Discussion

The substantial overall results of the implementation come with optimization of GA: 99% of Accuracy is achieved by the RF model, Ensemble Learning 98.90%, NB 70.40%, and SVM 72.40%. Before optimization, the RF reached a 99.60% of Accuracy, Ensemble Learning 99.60%, NB 71%, and SVM 66.4%. With this, the SVM classifier enhanced by six percent reaching 72% of Accuracy but the Ensemble Learning, and NB reduced one percent of Accuracy after optimization. The Recall, Precision, and F1-Score before and after optimization are shown in Figures 18 and 19. According to Figures 18 and 19 which summarize the evaluation results, it can be seen that the optimization part of the UGRansome1819 model has a slight improvement in the model performance. All metrics remain almost the same.

The CAIDA and UNSWNB-15 datasets' performance are compared to the UGRansome1819 with SVM, NB, and RF using the same optimization settings. The model's accuracy is presented in Table 9 before optimization. The UGRansome1819 outperformed the CAIDA and UNSWNB-15 datasets in terms of accuracy values (Figure 20). The model's accuracy is also illustrated with optimization in Table 10. It can be observed that the optimization improved in Accuracy on the CAIDA and UNSWNB-15 datasets (Figure 21).

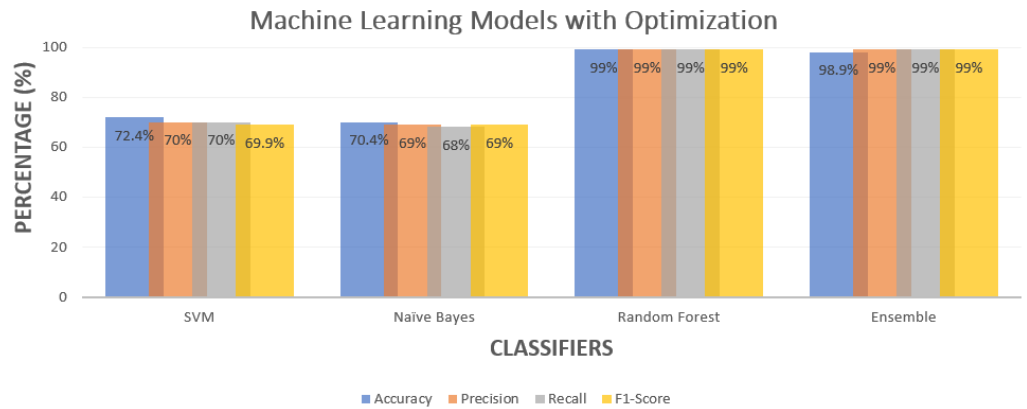


Figure 18. The Classification Results After Optimization.

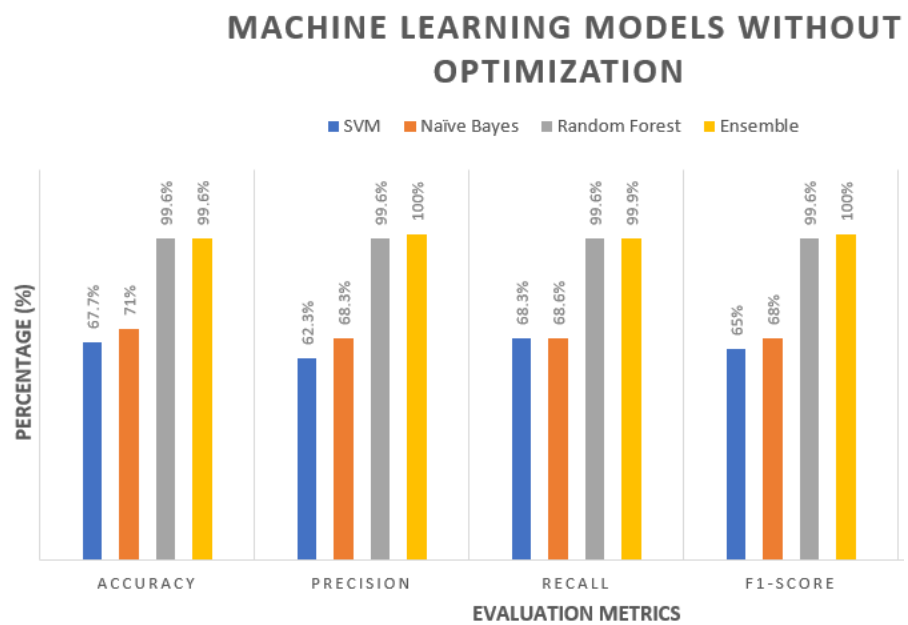


Figure 19. The Classification Results Prior Optimization.

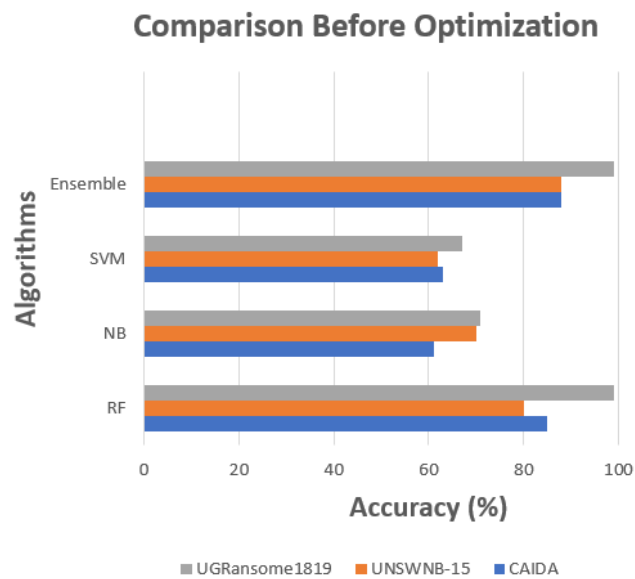


Figure 20. Comparing the UGRansome1819’s Accuracy to existing datasets (before optimization).

Table 9. Comparing the UGRansome1819’s Accuracy Prior Optimization.

Dataset	RF	NB	SVM	Ensemble
CAIDA	85%	61.8%	63.2%	88%
UNSWNB-15	80%	70%	62.4%	88.2%
UGRansome1819	99.6%	71%	67%	99.6%

Table 10. Comparing the UGRansome1819’s Accuracy After Optimization.

Dataset	RF	NB	SVM	Ensemble
CAIDA	95.6%	71.2%	73.7%	92.5%
UNSWNB-15	90.3%	75.1%	72.4%	84%
UGRansome1819	99%	70.4%	72.4%	99%

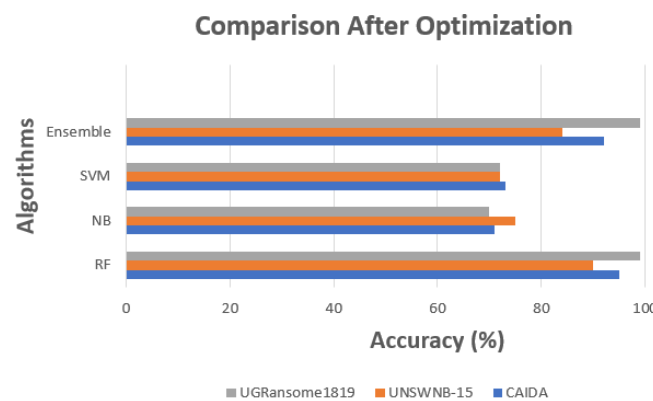


Figure 21. The UGRansome1819’s Accuracy compared to existing datasets (after optimization).

7. Conclusions

We have analyzed the classification of 0-days threats and anomalous intrusion using a novel AIDS dataset with cloud services. This research indicates that the UGRansome1819 dataset uses a multi-class prediction space. Relevant features are detected using this dataset and three ML algorithms are computed before and after optimization of the GA. The Ensemble Learning analyzed the overall performance of 0-day threats detection. Accuracy, F1-Score, Recall, and Precision values were used to evaluate the classification model of the UGRansome1819. To compare results, the NB, RF, and SVM are used with GA after optimization and Ensemble Learning before optimization to test and train the dataset accordingly. A Genetic Optimizer with each selected ML classifier was utilized as a feature extraction technique that used a Decision Tree classifier. Our research findings suggest the optimization with RF and Ensemble Learning to obtain accurate classification rates of 0-day threats. Additionally, the UGRansome1819 outperformed the UNSWNB-15 and CAIDA datasets in terms of accuracy values. It was also observed that the optimization technique improved in accuracy on the CAIDA and UNSWNB-15 datasets. The experiments demonstrate the instability of single classifiers such as SVM and NB and suggest optimized validation techniques which can aggregate weak classifiers into an ensemble of the genetic optimizer to enhance the classification performance. The Laplace smoothing improved the NB performance compared to SVM by optimizing the classification probability. The UGRansome1819 model’s sensitivity and specificity were estimated to be 100% with three predictors of 0-day threats as Signature, Synthetic Signature, and Anomaly. The results of ensemble learning produce a strong classification by combining weak classifiers such as SVM and NB. With this, the predictive capability of ensemble learning could be improved by achieving 100% of Precision and F1-Score. The test classification accuracy of the SVM

model after optimization improved by 6%. To improve the performance and increase the accuracy, one should add more data samples to the training data and fine-tune the model to obtain more accurate and optimized performance. Pruning was used to avoid overfitting of data and provided a better model in terms of accuracy. Similar research should be conducted on the UGRansome1819 dataset using Deep Learning and analyzing the feature extraction to determine if Deep Learning also works on this dataset. One can also attempt to enhance the SVM and NB Accuracy on the UGRansome1819 dataset and plot the Receiver Operating Characteristic curves (ROC) for the models. It is always great to have such graphs in the evaluation to quickly assess the classification quality of the ML models. Lastly, unsupervised algorithms with specific feature selection methods such as Boruta should be applied to the UGRansome1819 and compare the results with the current research outcome.

Author Contributions: Conceptualization, M.N., J.P.v.D., S.M.K. and S.R.Z.; writing—original draft preparation, M.N. and S.R.Z.; supervision, J.P.v.D. and S.M.K.; writing—review and editing, S.M.K., S.R.Z. and J.K.; experiments and testing, M.N., S.R.Z. and J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset and code used can be obtained upon request or downloaded at https://www.researchgate.net/publication/342200905_An_Ensemble_Learning_Framework_for_Anomaly_Detection_in_the_Existing_Network_Intrusion_Detection_Landscape (Public Files/Ugransome.zip), accessed on 20 April 2022. The experimental Python code used in this research can also be found (Public Files/Experimental Code Python.ipynb).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Abdelrahman, A.M.; Rodrigues, J.J.; Mahmoud, M.M.; Saleem, K.; Das, A.K.; Korotaev, V.; Kozlov, S.A. Software-defined networking security for private data center networks and clouds: Vulnerabilities, attacks, countermeasures, and solutions. *Int. J. Commun. Syst.* **2021**, *34*, e4706. [CrossRef]
2. Sharafaldin, I.; Gharib, A.; Lashkari, A.H.; Ghorbani, A.A. Towards a reliable intrusion detection benchmark dataset. *Softw. Netw.* **2018**, *2018*, 177–200. [CrossRef]
3. Cordero, C.G.; Vasilomanolakis, E.; Wainakh, A.; Mühlhäuser, M.; Nadjm-Tehrani, S. On generating network traffic datasets with synthetic attacks for intrusion detection. *ACM Trans. Priv. Secur. (TOPS)* **2021**, *24*, 1–39. [CrossRef]
4. Kasongo, S.M.; Sun, Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. *Comput. Secur.* **2020**, *92*, 101752. [CrossRef]
5. Dang, Q.V.; Vo, T.H. Studying the Reinforcement Learning techniques for the problem of intrusion detection. In Proceedings of the 2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 28–31 May 2021; pp. 87–91.
6. Vigna, G.; Kemmerer, R.A. NetSTAT: A network-based intrusion detection approach. In Proceedings of the 14th Annual Computer Security Applications Conference (Cat. No. 98EX217), Phoenix, AZ, USA, 7–11 December 1998; pp. 25–34.
7. Nkongolo, M.; van Deventer, J.P.; Kasongo, S.M. UGRansome1819: A Novel Dataset for Anomaly Detection and Zero-Day Threats. *Information* **2021**, *12*, 405. [CrossRef]
8. Otoum, Y.; Nayak, A. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. *J. Netw. Syst. Manag.* **2021**, *29*, 1–26. [CrossRef]
9. Ashoor, A.S.; Gore, S. Importance of intrusion detection system (IDS). *Int. J. Sci. Eng. Res.* **2011**, *2*, 1–4.
10. Hindy, H.; Brosset, D.; Bayne, E.; Seeam, A.K.; Tachtatzis, C.; Atkinson, R.; Bellekens, X. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access* **2020**, *8*, 104650–104675. [CrossRef]
11. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
12. Zoppi, T.; Ceccarelli, A.; Bondavalli, A. Unsupervised algorithms to detect zero-day attacks: Strategy and application. *IEEE Access* **2021**, *9*, 90603–90615. [CrossRef]

13. Khraisat, A.; Gondal, I.; Vamplew, P. An anomaly intrusion detection system using C5 decision tree classifier. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Melbourne, Australia, 3–6 June 2018; Springer: Cham, Switzerland, 2018; pp. 149–155.
14. Divekar, A.; Parekh, M.; Savla, V.; Mishra, R.; Shirole, M. Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives. In Proceedings of the 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), Kathmandu, Nepal, 25–27 October 2018; pp. 1–8.
15. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A survey of network-based intrusion detection data sets. *Comput. Secur.* **2019**, *86*, 147–167. [\[CrossRef\]](#)
16. Kilincer, I.F.; Ertam, F.; Sengur, A. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Comput. Netw.* **2021**, *188*, 107840. [\[CrossRef\]](#)
17. Lu, S.; Wei, X.; Li, Y.; Wang, L. Detecting anomaly in big data system logs using convolutional neural network. In Proceedings of the 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, 16th International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Athens, Greece, 12–15 August 2018; pp. 151–158.
18. Du, M.; Li, F.; Zheng, G.; Srikumar, V. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1285–1298.
19. Nkongolo, M.; van Deventer, J.P.; Kasongo, S.M.; and van der Walt, W. Classifying Social Media Using Deep Packet Inspection Data. In Proceedings of the 6th International Conference on Inventive Communication and Computational Technologies, Namakkal, India, 12–13 May 2022; Springer: Singapore, 2022; pp. 729–884.
20. Yu, X.; Joshi, P.; Xu, J.; Jin, G.; Zhang, H.; Jiang, G. Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 489–502. [\[CrossRef\]](#)
21. Alazab, A.; Hobbs, M.; Abawajy, J.; Alazab, M. Using feature selection for intrusion detection system. In Proceedings of the 2012 International Symposium on Communications and Information Technologies (ISCIT), Gold Coast, Australia, 2–5 October 2012; pp. 296–301.
22. Lawal, M.A.; Shaikh, R.A.; Hassan, S.R. An anomaly mitigation framework for iot using fog computing. *Electronics* **2020**, *9*, 1565. [\[CrossRef\]](#)
23. Shapoorifard, H.; Shamsinejad, P. Intrusion detection using a novel hybrid method incorporating an improved KNN. *Int. J. Comput. Appl* **2017**, *173*, 5–9. [\[CrossRef\]](#)
24. Paquet-Clouston, M.; Haslhofer, B.; Dupont, B. Ransomware payments in the bitcoin ecosystem. *J. Cybersecur.* **2019**, *5*, tyz003. [\[CrossRef\]](#)
25. Maseer, Z.K.; Yusof, R.; Bahaman, N.; Mostafa, S.A.; Foozy, C.F.M. Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. *IEEE Access* **2021**, *9*, 22351–22370. [\[CrossRef\]](#)
26. Dong, S. Multi class SVM algorithm with active learning for network traffic classification. *Expert Syst. Appl.* **2021**, *176*, 114885. [\[CrossRef\]](#)
27. Guezzaz, A.; Benkirane, S.; Azrou, M. A Novel Anomaly Network Intrusion Detection System for Internet of Things Security. In *IoT and Smart Devices for Sustainable Environment*; Springer: Cham, Switzerland, 2022; pp. 129–138.
28. Li, W.; Li, Q. Using naive Bayes with AdaBoost to enhance network anomaly intrusion detection. In Proceedings of the 2010 Third International Conference on Intelligent Networks and Intelligent Systems, Shenyang, China, 1–3 November 2010; pp. 486–489.
29. Gattineni, P.; Dharan, G.S. Intrusion Detection Mechanisms: SVM, random forest, and extreme learning machine (ELM). In Proceedings of the 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), Shenyang, China, 1–3 November 2010; pp. 273–276.
30. Disha, R.A.; Waheed, S. Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique. *Cybersecurity* **2022**, *5*, 1–22. [\[CrossRef\]](#)
31. Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **2017**, *5*, 21954–21961. [\[CrossRef\]](#)
32. Abu Al-Haija, Q.; Al-Badawi, A. Attack-Aware IoT Network Traffic Routing Leveraging Ensemble Learning. *Sensors* **2022**, *22*, 241. [\[CrossRef\]](#)
33. Gaikwad, D.; Thool, R.C. Intrusion detection system using bagging ensemble method of machine learning. In Proceedings of the 2015 International Conference on Computing Communication Control and Automation, Pune, India, 26–27 February 2015; pp. 291–295.
34. Kasongo, S.M. An Advanced Intrusion Detection System for IIoT Based on GA and Tree Based Algorithms. *IEEE Access* **2021**, *9*, 113199–113212. [\[CrossRef\]](#)
35. Onah, J.O.; Abdulhamid, S.M.; Misra, S.; Sharma, M.M.; Rana, N.; Oluranti, J. Genetic Search Wrapper-Based Naïve Bayes Anomaly Detection Model for Fog Computing Environment. In Proceedings of the International Conference on Intelligent Systems Design and Applications, Salem, India, 5 February 2021; Springer: Cham, Switzerland, 2020; pp. 1371–1382.
36. Yihunie, F.; Abdelfattah, E.; Regmi, A. Applying machine learning to anomaly-based intrusion detection systems. In Proceedings of the 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, 3 May 2019; pp. 1–5.

37. Xu, L.; Xiong, W.; Zhou, M.; Chen, L. A Continuous Terminal Sliding-Mode Observer-Based Anomaly Detection Approach for Industrial Communication Networks. *Symmetry* **2022**, *14*, 124. [[CrossRef](#)]
38. Ahmad, T.; Truscan, D.; Vain, J.; Porres, I. Early Detection of Network Attacks Using Deep Learning. *arXiv* **2022**, arXiv:2201.11628.
39. Le, T.T.H.; Kim, H.; Kang, H.; Kim, H. Classification and Explanation for Intrusion Detection System Based on Ensemble Trees and SHAP Method. *Sensors* **2022**, *22*, 1154. [[CrossRef](#)]
40. Alzaqebah, A.; Aljarah, I.; Al-Kadi, O.; Damaševičius, R. A Modified Grey Wolf Optimization Algorithm for an Intrusion Detection System. *Mathematics* **2022**, *10*, 999. [[CrossRef](#)]
41. Liu, G.; Zhao, H.; Fan, F.; Liu, G.; Xu, Q.; Nazir, S. An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs. *Sensors* **2022**, *22*, 1407. [[CrossRef](#)]
42. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150. [[CrossRef](#)]
43. Ahmad, A.; Harjula, E.; Ylianttila, M.; Ahmad, I. Evaluation of machine learning techniques for security in SDN. In Proceedings of the 2020 IEEE Globecom Workshops (GC Wkshps), Taipei, Taiwan, 7–11 December 2020; pp. 1–6. [[CrossRef](#)]
44. Jiang, H.; He, Z.; Ye, G.; Zhang, H. Network intrusion detection based on PSO-XGBoost model. *IEEE Access* **2020**, *8*, 58392–58401. [[CrossRef](#)]
45. Zhang, C.; Yu, M.; Wang, W.; Yan, F. MARk: Exploiting Cloud Services for Cost-Effective, SLO-Aware Machine Learning Inference Serving. In Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC 19), Renton, WA, USA, 10–12 July 2019; pp. 1049–1062.
46. Rauschmayr, N.; Kumar, V.; Huilgol, R.; Olgiati, A.; Bhattacharjee, S.; Harish, N.; Kannan, V.; Lele, A.; Acharya, A.; Nielsen, J.; et al. Amazon SageMaker debugger: A system for real-time insights into machine learning model training. *Proc. Mach. Learn. Syst.* **2021**, *3*, 770–782.
47. Sahu, S.K.; Mohapatra, D.P.; Rout, J.K.; Sahoo, K.S.; Pham, Q.V.; Dao, N.N. A LSTM-FCNN based multi-class intrusion detection using scalable framework. *Comput. Electr. Eng.* **2022**, *99*, 107720. [[CrossRef](#)]
48. Bevis Jinila, Y.; Prayla Shyry, S.; Christy, A. A Multi-component-Based Zero Trust Model to Mitigate the Threats in Internet of Medical Things. In *Data Engineering for Smart Systems*; Springer: Singapore, 2022; pp. 605–613.
49. Leng, F.; Zhang, C.-L.; Chen, W.-Y.; Zeng, Y. Attack analysis based on protocol information of Snort rules. *J. Comput. Appl.* **2022**, *178*, 14–19. [[CrossRef](#)]
50. Nkongolo, M.; van Deventer, J.P.; and Kasongo, S.M. The Application of Cyclostationary Malware Detection Using Boruta and PCA. In Proceedings of the 5th International Conference on Computer Networks and Inventive Communication Technologies (ICCNCT 2022), Coimbatore, India, 1–2 April 2022; Springer: Singapore, 2022; pp. 631–646.
51. Xiao, H.H.; Yang, W.K.; Hu, J.; Zhang, Y.P.; Jing, L.J.; Chen, Z.Y. Significance and methodology: Preprocessing the big data for machine learning on TBM performance. *Undergr. Space* **2022**, *in press*. [[CrossRef](#)]
52. Nkongolo, M. Classifying search results using neural networks and anomaly detection. *Educator Multidiscip. J.* **2018**, *2*, 102–127.
53. Kasongo, S.M.; Sun, Y. Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. *J. Big Data* **2020**, *7*, 1–20. [[CrossRef](#)]
54. Ranga, V. *On Evaluation of Network Intrusion Detection Systems: Statistical Analysis of CIDD5-001 Dataset Using Machine Le...*; Universiti Putra Malaysia Press: Serdang, Malaysia, 2018; pp. 1–35.
55. Pacheco, Y.; Sun, W. Adversarial Machine Learning: A Comparative Study on Contemporary Intrusion Detection Datasets. In Proceedings of the ICISSP, Toledo, OH, USA, 11–13 February 2021; pp. 160–171. [[CrossRef](#)]
56. Sajja, G.S.; Mustafa, M.; Ponnusamy, R.; Abdufattokhov, S. Machine Learning Algorithms in Intrusion Detection and Classification. *Ann. Rom. Soc. Cell Biol.* **2021**, *25*, 12211–12219.