A Comprehensive Analysis of Machine Learning based Intrusion Detection System for IoT-23 Dataset

Yang G. Kim[1], Kazi J. Ahmed[1], Myung J. Lee[1], and Kazuya Tsukamoto[2]
[1] The City College of New York, ykim2@ccny.cuny.edu, kahmed06@citymail.cuny.edu, mlee@ccny.cuny.edu
[2] Kyutech, Japan, tsukamoto@cse.kyutech.ac.jp

Abstract. With the proliferation of IoT devices, securing the IoT-based network is of paramount importance. IoT-based networks consist of diversely purposed IoT devices. This diversity of IoT devices necessitates diverse dataset analysis to ensure effective implementation of Machine Learning (ML)-based cybersecurity. However, much-demanded real-world IoT data is still in short supply for active ML-based IoT data analysis. This paper presents an in-depth analysis of the real-time IoT-23 dataset [9]. Exhaustive analysis of all 20 scenarios of the IoT-23 dataset reveals the consistency between feature selection methods and detection algorithms. The proposed ML-based intrusion detection system (ML-IDS) achieves significant improvement in detection accuracy, which in some scenarios reaches 100%. Our analysis also demonstrates that the required number of features for a high detection rate of greater than 99% remains small, i.e., 2 or 3, enabling ML-IDS implementation even with resource-constrained IoT processors.

1 Introduction

Today, we live in a world where there are more IoT-connected devices than humans. The projected growth of IoT devices over the next five years is expected to be ten times the growth in the past five years, reaching 55 billion, i.e., more than four devices per person on earth[1].

The IoT is transforming how we perceive and interact with our surroundings. As the number of connected IoT devices grows, the threat of cyberattacks exploiting constrained resources and the vulnerability of IoT devices also rises. Enhancing the security of IoT devices is a critical issue, as IoT security is about securing IoT devices and protecting the integrity of the data and the networks to which IoT devices are connected. Inadequate security would hinder the growth of the IoT industry. A recent increase in the frequency, extensiveness, and complexity of cyberattacks affecting thousands of organizations illustrates the persistent challenge of securing connected devices.

The intrusion Detection System (IDS) monitors network traffic flow to identify attacks to protect network infrastructure. The IDS is classified broadly as either signature or anomaly-based [1]. The signature-based IDS compares the traffic flow to the existing attack patterns to detect potential intrusions. The anomaly-based IDS creates a normal profile from the usual behavior of the network and considers any deviations from the normal profile as attacks. Although the signature-based IDS is employed [2] in the cloud at the expense of the communication cost between the cloud and IoT devices, it still cannot detect any unknown attacks, including zero-day attacks. Due to IoT devices' diversity and resource constraints, the detection based on predefined attack patterns is unsuitable for IoT networks. Therefore, recent anomaly detection research has demonstrated the promise of Machine Learning (ML) in identifying malicious Internet traffic [1][3][4]. The IDS with ML categorizes ML into supervised and unsupervised learnings [4]. However, little effort has been made to engineer ML models with features geared towards IoT device networks [1]. Adopting ML into the IDS requires high computation power; however, the required computation can be significantly reduced by selecting a minimum set of attack features.

The anomaly-based IDS can be categorized as either one-class learning or a multi-class learning method. In the one-class learning method, ML uses only benign data to learn the common behavior to build the model of the normal traffic pattern. The threshold for attacks is determined based on the common behavior and a cost (loss) function, e.g., the Mean Square Error, is adjusted to reduce the false alarm [5]. Once the decision point is established, the detection process determines an attack if a new data value is greater than the threshold. Otherwise, the new data value is normal (binary classification). While the one-class learning method uses only benign data to build the traffic pattern model, the multi-class learning method simultaneously employs normal and attack data during the training period. The multi-class learning is categorized as either binary classification or multi-class classification. As described above, the binary classification determines whether there is an attack or not, whereas the multi-class classification considers not only attack attempts but also what the most vulnerable attack profile is for further in-depth defense planning [6].

One of the challenges of utilizing ML in cybersecurity is effective feature selection that enhances threat detection capability. The effectiveness of ML depends on the accumulation of useful data, but an excessive amount

[1]https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/non-geek-a-to-z-guide-to-internet-of-things-108846.pdf.

of unwanted data may result in undesirable outcomes [1]. As a result, ML trained with unwanted data deteriorates the quality of the detection algorithm. A feature selection algorithm can select wanted data to feed into a detection algorithm. However, a feature selection algorithm has its characteristics. For example, the correlation feature selection (CFS) refers to minimum redundancy among the features by ruling out high correlation features. Minimum Redundancy Maximum Relevance (MRMR) feature selection considers both the correlation and relevance to the parameters (e.g., importance). Note that CFS and MRMR can be either supervised or unsupervised. On the other hand, the feature selection in Bot-IoT [3] adopted unsupervised learning, while the detection algorithm was based on supervised learning. The inconsistency in feature selection and detection algorithm degrades the detection rate.

Therefore, in our proposal, the consistency of the learning method, feature selection, and detection algorithm is maintained for a high detection rate. Both feature selection and detection algorithms use supervised learning, and the character of the feature algorithm is reflected in the detection algorithm. The feature selection is performed by Information Gain (IG) in the Tree detection algorithm; IG is calculated with entropies, upon which the Tree algorithm is built. Artificial Neural Network (ANN) utilizes Sequential Forward feature Selection (SFS), in which the metrics of both ANN and SFS are misclassification.

Our main contribution is as follows. First, we align the metrics of the feature selection scheme with those of the detection algorithm to improve the overall detection accuracy. Maintaining the consistency between feature selection and detection algorithm improves the performance by using the same metrics for both. Second, the computational complexity of ML decreases as the number of features used in ML is reduced. Our comprehensive analysis of the IoT-23 data offers the minimum number of features necessary to maintain a high detection rate, e.g., 99%. Last, we analyze and generalize the uncertainty (diversity) of different attack types and time periods in large-scale IoT-23 dataset scenarios. The authors of [6] investigated only a select few individual and combined scenarios that may not support many practical IoT applications.

2 Related Work

Related work is discussed in view of whether a work maintains the consistency between feature selection and detection algorithms among the IoT datasets designed specifically for anomaly-based IDS in IoT networks. Bot-IoT [3] mimics IoT data in a testbed network while projecting Botnet malware attacks into IoT-mimicked devices in the VM-based testbed network. The feature selection is made with the correlation coefficient and the joint entropy while choosing the ten bests out of 30 features. There are a total of 44 features, in which 14 features engineered by the authors are added to 30 original flow-based analyses (e.g., Argus tool) without evaluating 44 features at once. Bot-IoT performs feature selection of both CFS and joint entropy through unsupervised learning, while the detection algorithms were evaluated through supervised learning.

The N-BaIoT [5] dataset performs an empirical evaluation with real traffic data from nine commercial IoT devices infected with authentic botnets from Mirai (virus/worm attack due to lack of update mechanism) and BASHLITE (firmware-based malware). 12 features, including 8 features for packet size (inbound and outbound), 4 for packet count, and 3 for packet jitter were extracted. Instead of applying a feature selection algorithm, the extracted feature is used directly, and the detection algorithm is based on unsupervised learning for binary classification.

The Random Forest (RF) algorithm based on a wrapper feature selection was developed by N. Moustafa [7] from 44 features using the Mean Squared Error (MSE) between the node of a tree and its children's nodes to determine how the performance is related to the MSE. The most important features were scaled in a range of [0,1], where the value 1 refers to the highest correlated attribute (e.g., src_IP) and 0 denotes the lowest correlated attribute (e.g., duration). Utilizing RF in both feature selection and detection algorithms improves detection rate due to the consistency between feature selection and detection algorithms for binary classification.

IG for the Waikato Environment for Knowledge Analysis (Weka) is calculated by Hegde *et al.* [6] to rank the most important features in both binary and multi-classification among 22 features using one-hot encoding. The IG and Multi-class Decision Forest (MDF) show consistency as both is based on entropies. However, the number of features increases due to a one-hot encoding, and that increase may offset the benefit from the consistency due to the increase in the computation complexity.

3 IoT-23 Dataset

The scarcity of available data leads to bias in AI, and an extensive dataset is required to ensure objective decision-making in AI applications. One significant research challenge in IoT-based networks is the lack of

comprehensive network-based datasets that reflect modern network traffic scenarios [8]. Nonetheless, the IoT-23 dataset [9] provides data obtained from an IoT-based network with 23 scenarios involving Windows and Linux OSs in diverse IoT-related attack types. There has not been such extensive network data collection for the real environment as the IoT-23 dataset.

The IoT-23 data were collected for at least one hour and up to 112 hours depending on how quickly the *pcap* file size increased. The 16th scenario containing over 73 million records, is the largest scenario. The second-largest scenario is the 2nd scenario with 67 million records. The third-largest scenarios are the 10th and the 12th, each with 54 million records. The data size for a single scenario in the IoT-23 is larger than the size of the Bot-IoT dataset [3]. A broad range of ratios of the number of attacks to the amount of normal data exists due to diverse scenarios that occur in real-world networks. The data size for each scenario, including attacks and normal data, ranges from 237 records (3rd scenario) to over 73 million records (16th scenario). We analyzed all 20 scenarios using Matlab.

4 Feature Selection Algorithms

Feature selection is an optimization problem that can be categorized into two parts: search strategy and evaluation strategy. The two parts are further categorized into optimal and heuristic for the search and filter and wrapper methods for the evaluation. The major difference between the filter and the wrapper methods is that the filter is independent of the classification (i.e., detection) algorithm, while the wrapper is related to a classification algorithm, where the feature selection uses the same metrics as the detection algorithm. Among 18 features (normalization in Fig.1.), the smallest possible number of features is selected for efficient implementation in resource-constrained IoT devices. Refer to [9] for details attributes of 18 features used in the IoT-23 data set.

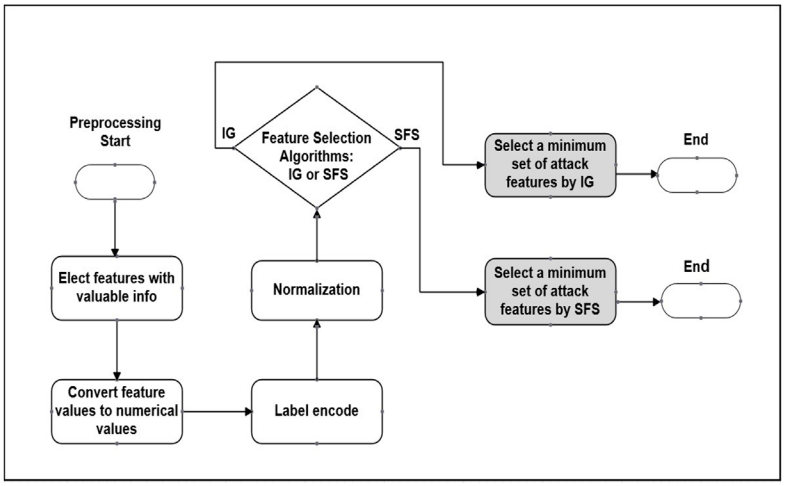

Fig. 1. The proposed framework for the feature selection process.

Two feature selection algorithms were investigated: IG and greedy-based SFS. As the Tree algorithm is built along with the entropies of features, i.e., IG, the Tree algorithm provides the best detection rate. The most important feature, i.e., the highest IG, becomes the parent of the Tree, from which the rest of the Tree is constructed based on the IG rank. SFS easily adopts the consistent feature selection and detection as it is a computationally efficient sub-optimal solution [10]. The greedy-based SFS uses ANN related to misclassification. Thus, SFS would be favorable to ANN detection algorithm as SFS performs feature selection on detection algorithm metrics. Our primary concern is the consistency of the feature selection method and detection algorithm to maximize the overall detection performance.

The Information Gain (IG) is calculated as presented in Eqs. (1-3).

$$Info\ (D) = -\sum_{j=1}^{m} \frac{freq(C_j, D)}{|D|} \log_2 \left\{ \frac{freq(C_j, D)}{|D|} \right\} \quad (1)$$

$$Info\ (T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} info(T_i) \quad (2)$$

$$IG\ (A_i) = Info(D) - Info(T) \quad (3)$$

Note that even with the same number of data records and features, the IG values for binary and multiclass classifications should be different due to the term *freq* (C_j) in Eq. (1). D is the number of data records. The IG depends on the number of attributes in the labeled class (C_j), in which each attribute (T) in the label is calculated as its probability of occurrence in Eq. (2) [11]. Furthermore, the IG is also influenced by the number of features and the number of data records. Based on the IG, those features in the top 10 to 20% of the IG values are selected above the line [12] for each scenario that has different the IG values, and the line threshold varies as well. The evaluation, starting from the highest IG feature to the lowest IG feature, shows that the performance diminishes rapidly beyond the 20% IG.

The SFS as a wrapper approach includes a learning/classification algorithm, under which misclassification rate can be used as performance metrics. During the feature selection in SFS, a classification algorithm to determine misclassification is used in ANN. The procedure for SFS is as follows (refer to Fig. 2). First, the best single feature is selected for the least misclassification rate among all features. Then, pairs of features are formed using one of the remaining features, and the best pair is selected. Next, triplets of features are formed using one of the remaining features, and the best triplet is selected. This procedure continues until one of the stopping conditions is met. The stopping conditions are: the iteration reaches the 1000th epoch, zero misclassification, gradient 10^{-6}, or six consecutive verification checks. From this algorithm, the minimum number of attack features is determined. For example, the least one is just one feature for scenarios 6, 7, 13, and 18, as shown in Table 1.

1. Start with the empty set $Y_0 = \{ \emptyset \}$
2. Select the next best feature
$x^+ = \arg \max_{x \notin Y_k} J(Y_k + x)$
Misclassification = J (x')

3. Update $Y_{k+1} = Y_k + x^+$; $k = k + 1$
4. Go to 2

Stopping condition: 1000 epochs, zeros misclassification gradient 10^{-6}, and six consecutive verification checks

Fig. 2. The PseudoCode of the SFS algorithm.

5 Performance Evaluation

The detection algorithms are performed after the feature selection by IG and SFS with a minimal set of attack features. The experiment scenario or setting and conditions are as follows. The Tree algorithm uses the hyperparameter-tuned Tree due to the variety of scenarios. After identifying the hyperparameters (e.g., maximum depth, minimum sample leaf, and minimum sample split) for each scenario, the Tree algorithm is performed for detection accordingly. Hyperparameters for ANN are also predetermined while using cross-entropy for the activation function. The ANN we use has two hidden layers, in which the first layer has 32 neurons and the second layer has 16 neurons. The input layer relies on the number of features. The output layer is associated with the labeled classes for each scenario. The data is divided into three groups: training, verification, and test data. We used five cross-validation folds where the validation check is completed after each epoch (iteration) to determine whether the model is appropriate. Computation time refers to training and detection time.

All 20 scenarios are analyzed via IG and SFS algorithms for feature selection, and Tree and ANN for detection, and these 20 scenarios are divided into three groups, simple, intermediate, and complex in terms of the number of records, and one scenario from each group is presented below with in-depth analysis due to space constraint.

Simple case (Scenario 8): In the 8th scenario with 4426 records and three classes, the features 9, 14, 15, 16, 17, and 18 are selected as these features exhibit exceeding the IG threshold above the line (threshold) in Fig. 3 [12]. *Resp-ip-bytes* (18) is shown to be the highest IG, as shown in Fig. 3. The Tree detection rate was 99.9% while classifying two C&C File Download as File Download and all three File Download as C&C File Download for a total of five misclassifications as shown in Fig. 4 (left). For the SFS, 3, 5, and 15 features had ANN detection accuracy of 100% without any misclassifications as shown in Fig. 4 (right). Although feature 15 is common, the feature set for IG and SFS varies wildly depending on the algorithm, and our study was motivated by the need for

careful selection of the proper algorithm through experimentation. The computation time for the Tree and ANN was about 0.4 and 1 second, respectively. Note that C&C File Download occurs at the attacker's C&C server for a secondary attack.

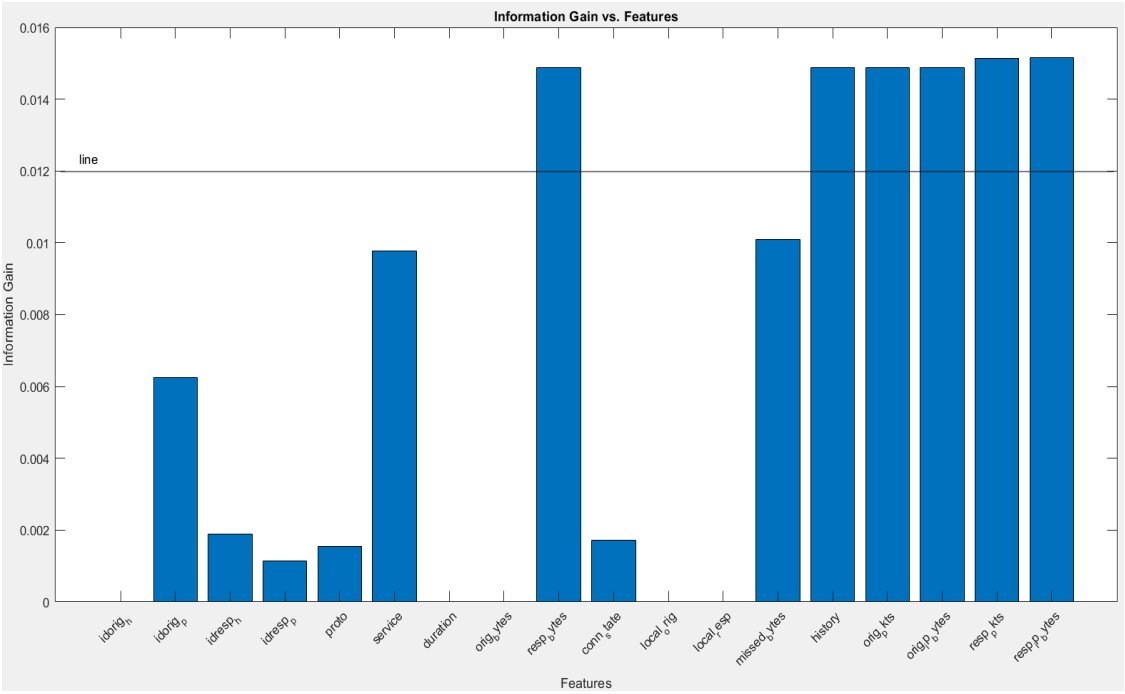

Fig. 3. Information Gain for analysis of 8th scenario.

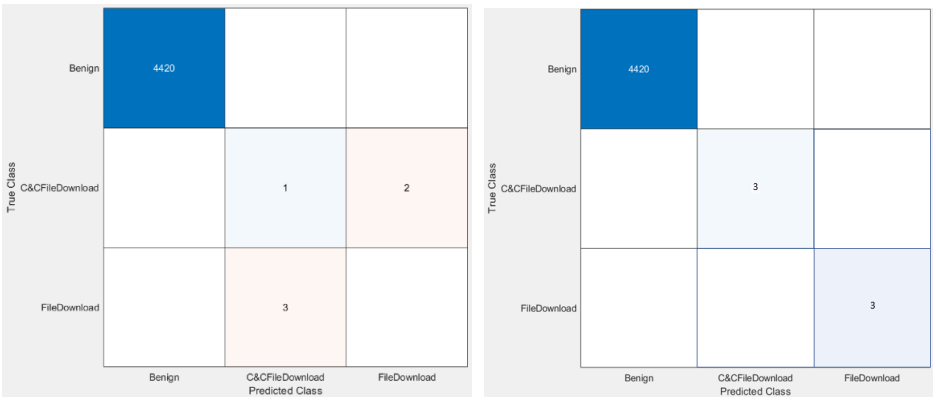

Fig. 4. Analysis of 8th scenario: the left is Tree, and the right is ANN.

Intermediate Case (Scenario 15): In the 15th scenario, with more than 3 million records and six classes, the features 2, 5, 10, and 14 are selected from IG with the top 10-20% features. *Conn-state* (10) achieved the highest IG. The Tree had 1724 misclassifications with a computation time of 2618 seconds. The detection accuracy for Tree was 99.9%. Surprisingly, for the SFS, 1, 3, 8, and 14 features had ANN detection accuracy of 100% with just two misclassifications and a computation time of 1315 seconds. Again, feature 14 is the only common one in this case. Note that the testing environment for the KDD dataset (presumably Internet router/switch) would differ from the IoT dataset. The KDD dataset essentially shares the same traits regardless of the application. However, the IoT dataset in varying domains and applications do not share the same traits, so each application of the IoT dataset needs to be considered separately. This characteristic calls for "federated learning" more often in an IoT environment than in a router/switch-based Internet. We plan to investigate federated learning in our future work.

Complex Case (Scenario 12): In the 12th scenario, with over 54 million records and four classes, features 2 and 4 are selected from the IG. *Idresp*$_p$ (4) achieved the highest IG, as shown in Fig. 5. The detection rate for Tree was 100% (due to rounding off) for a total of 8660 misclassifications as shown in Fig. 6 (left). The computation time for Tree took 55 minutes. For the SFS, features 2, 4, and 14 had ANN detection accuracy of 100% for a total of 211 misclassifications as shown in Fig. 6 (right). The computation for ANN with the three selet features (2, 4, and 14) took 1 hour and 15 minutes while that with all 18 features was 6 hours and 30 minutes.

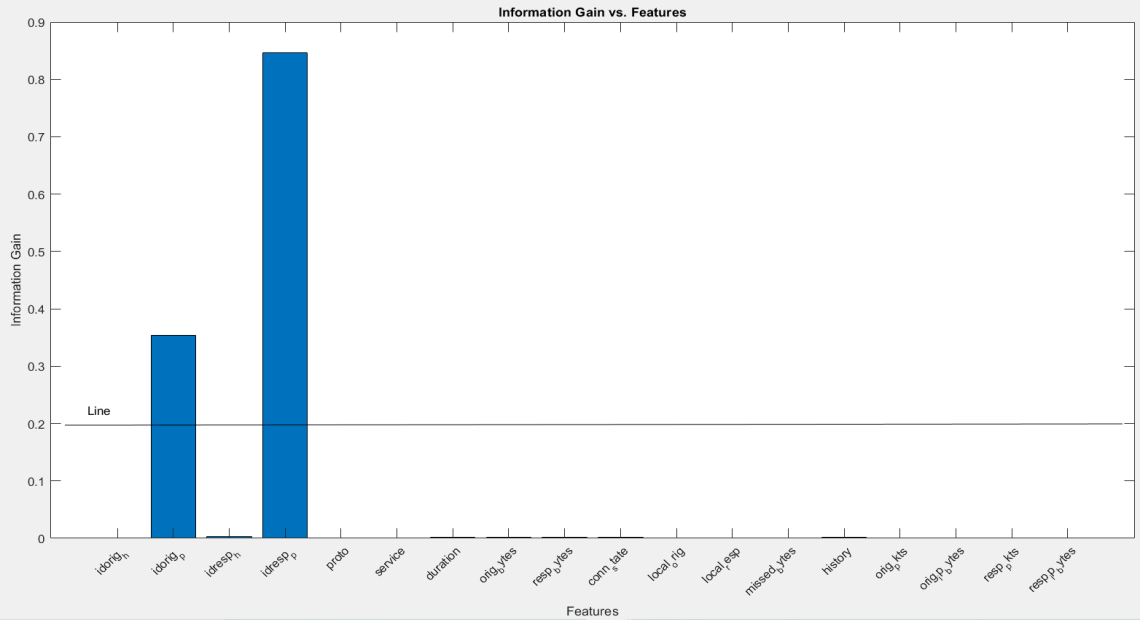

Fig. 5. Information Gain for analysis of 12th scenario.

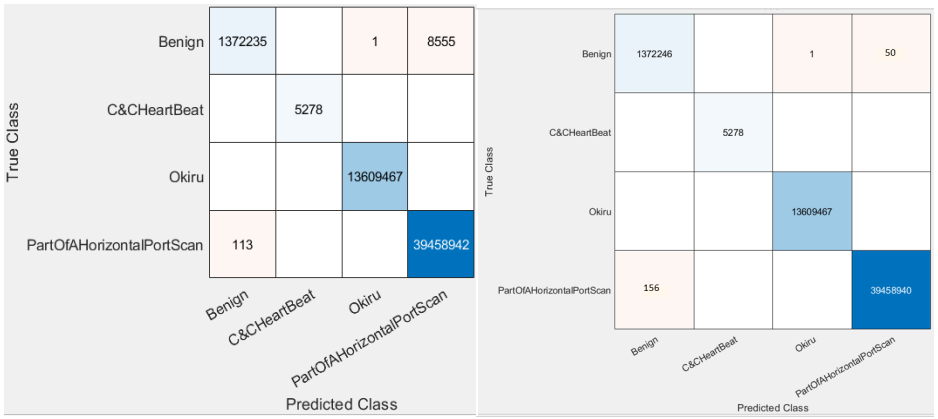

Fig. 6. Analysis of 12th scenario: the left is Tree, and the right is ANN.

Table 1 shows the results of the extensive computational analyses. The appropriate feature set by IG and SFS for each scenario contributes to the reduction of computations complexity. The performance shown in Table 1 is achieved through the consistency in feature selection and detection, without which the 100% or nearly 100% accuracy would not have been possible. As seen in Table 1, the most frequent feature selections for IG are 2, 4, and 14, while those for SFS are 3 and 14. Therefore, the most crucial feature in IG and SFS is 14, 'history'.

Table 1. Feature selections and performance comparison.

Scenario	Features by IG	Features by SFS	Tree	ANN
1	2,3,4,10,14	3,4,14	100%	100%
2	2,4	2,3,4,10,16	100%	100%
3	3,4,10,14	14,16	99.60%	100%
4	2,4,10,14	4,8,14,17	100%	100%
5	2,5,8,10,14	2,6,8,11	100%	100%
6	5,14	3	100%	100%
7	4,5,14	3	100%	100%
8	9,14,15,16,17,18	3,5,15	99.90%	100%
9	3,4,14	3,6	100%	100%
10	2,5,10,14	1,3,8,14	100%	100%
11	2,4	4,7	100%	100%
12	2,4	2,4,14	100%	100%
13	4,5,14	3	100%	100%
14	2,4,10,14	1,6,8,10,14	100%	100%
15	2,5,10,14	1,3,8,14	99.90%	100%
16	10,14	4,10,14	100%	100%
17	2,4,5,14	3,4	100%	100%
18	4,5,15	14	100%	100%
19	8,9,10,14	10,14	100%	100%
20	2,4,5,14	4,14	100%	99.90%

6 Performance Analysis

As expected, ANN outperformed Tree in all scenarios, except for the 20th scenario with over one million records and the 14th scenario with over 10 million records. During the feature selection by SFS in all scenarios, the stopping condition was reached with 'the gradient.' However, in the 20th scenario, the SFS was stopped by 'the validation check,' which means that our ANN model is inappropriate for that specific scenario. The inappropriate model means that the verification check happened six times. As a result, the detection algorithm for ANN is also stopped by 'the iteration' rather than 'the gradient' as usual. The Tree had just four misclassifications with a detection rate of 100%, whereas the ANN had 1437 misclassifications with a detection rate of 99.8575%. The computations time for Tree and ANN was 28 seconds and 938 seconds, respectively.

During the analysis of the 14th scenario, the gradient values of the selected features were almost the same and just below the limited gradient in which the contribution for each feature was equalized, resulting in decreased detection rate and increased computation time. The Tree had only 14 misclassifications with a detection rate of 100%, while the ANN had 345 misclassifications with a detection rate of 99.9967%. The computation times for Tree and ANN were 262 seconds and 1320 seconds (due to the increased number of features), respectively. The SFS algorithm was stopped by 'the gradient' as usual.

Tree performs comparatively well, achieving an accuracy rate of nearly 100%, suggesting that the data may be segmented for relatively small datasets. However, when the dataset is complex, the performance of the Tree deteriorates while the ANN continues to show an excellent detection rate for all dataset sizes. In general, ANN refers to its high complexity and computation time. Our prior study [13] supported that the NSL-KDD dataset with increased inputs/features showed the highest detection performances. For example, for the KDD dataset, the optimal number of features was 15, and the performance of ANN degraded when there were either less than or more than 15 inputs. In contrast, the detection accuracy for ANN in the IoT-23 dataset with fewer features resulted in higher performance than with all features. Note that the detection accuracy for the Tree algorithm in scenario 19 is higher with all 18 features than with the four selected features (8, 9, 10, and 14) as shown in Table 1, though the difference is insignificant. The computation time for the ANN in scenario 19 was only 12 seconds with the two selected features (10 and 14) whereas it was 64 seconds with all 18 features with the same performance. ANN with a smaller number of features requires less computation time in some scenarios than that of Tree, which can be the simplest ML classifier. Since the performance of ANN and computation time in the IoT-23 dataset rely on the number of inputs, it is feasible for ANN to be implemented into IoT devices after selecting an appropriate number of features.

The major claims of our study are as follows. One, the ANN outperforms the Tree in view of detection rate, but its complexity hinders its implementation in IoT devices. The complexity of the ANN may be reduced by selecting a minimal set of attack features. Two, we show the superiority of the ANN over the Tree, especially when applying the consistent metrics in feature selection and detection algorithms. Finally, in IoT-23 data analysis, selecting a fewer number of features achieves improved detection performance in the ANN algorithm due to the use of the same metrics for feature selection and detection. This result is contrary to the KDD data in

the Internet router environment. Clearly, fewer features require less computation time to the point that even IoT devices are fully capable of handling the required ANN computation, opening an opportunity to implement it in IoT devices.

7 Conclusion and Future Work

This article analyzed the several-gigabyte IoT-23 dataset comprised of real environment-based data. With in-depth analysis, we applied Machine Learning in Intrusion Detection systems for securing IoT-based networks while ensuring consistency between feature selection and detection algorithm. This consistency enhanced the detection accuracy to nearly 100%. Furthermore, the computation time is reduced by selecting a minimal set of attack features in the computation of the complex ANN. All 20 scenarios in the IoT-23 are analyzed for feature selection and detection. One of the scenarios behaves differently than others; the Tree with 18 features in scenario 19 outperforms that with a fewer number of features while the performance of the ANN remains the same. However, even in that scenario, the computation time for the ANN with a few select features drops to about five times less than when all 18 features are selected. Due to the heterogeneity of IoT data inherent to diverse IoT sectors, applying the domain knowledge to Artificial Intelligence during feature engineering and detection will enhance overall detection performance. Our in-depth analysis of the IoT-23 dataset demonstrated the feasibility of an ML-based Intrusion Detection System for securing a heterogeneous IoT-device-connected home or enterprise network. We will further study Federated Learning based on distributed model training using local datasets from large-scale nodes without uploading the raw training data.

Acknowledgments

The authors acknowledge CUNY high performance computing Center resources (csi.cuny.edu/cunyhpc) available for conducting the research reported in this paper. This research is supported in part by NSF IRNC Grant No. 2029295.

References

[1] N. Moustafa *et al.*, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1-6.
[2] M. Hammoudeh *et al.*, "Network Traffic Analysis for Threat Detection in the Internet of Things," in IEEE Internet of Things Magazine, vol. 3, no. 4, pp. 40-45, December 2020.
[3] N. Koroniotis *et al.*, "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset". Future Generation Computer Systems 100 (2019) 779–796.
[4] Tim M. Booij *et al.*, "ToN IoT-The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion datasets." IEEE Internet of Things Journal (2021).
[5] Y. Meidan *et al.*, "N-BaIoT-network-based detection of IoT botnet attacks using deep autoencoders". IEEE Pervasive Computing, 17, 12–22 (2018).
[6] M. Hegde *et al.*, "Identification of Botnet Activity in IoT Network Traffic Using Machine Learning," 2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), 2020, pp. 21-27.
[7] N. Moustafa. "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets." Sustainable Cities and Society (2021).
[8] MR Anawar *et al.*, "Fog computing: An overview of big IoT data analytics." Mobile Computing, 2018.
[9] Sebastian Garcia, Agustin Parmisano, & Maria Jose Erquiaga. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set].
[10] FJ Ferri *et al.*, " Comparative study of techniques for large-scale feature selection". Machine Intelligence and Pattern, 1994 – Elsevier.
[11] H. Kayacik *et al.*, "Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets." Proceedings of the third annual conference on privacy, security and trust. Vol. 94. 2005.
[12] O. Igbe *et al.*, "Distributed Network Intrusion Detection Systems: An Artificial Immune System Approach," 2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2016, pp. 101-106, doi: 10.1109/CHASE.2016.36.
[13] Yang G. Kim *et al.*, "Proportional Voting based Semi-Unsupervised Machine Learning Intrusion Detection System". Journal of Computer Science and Information Technology, 8(2), pp. 1-9. December 2020.