International Journal of P2P Network Trends and Technology
ISSN: 2249–2615 / https://doi.org/10.14445/22492615/IJPTT-V14I1P401
Volume 14 Issue 1, 1-9, Jan-Apr 2024

Original Article

A Comprehensive Investigation on the Identification of Real and Encrypted Synthetic Network Attacks using Machine Learning Algorithms

Swati Chaudhari[1,2*], Pratyush Shukla[3], Archana Thakur[4]

[1]*Institute of Engineering and Technology, Devi Ahilya Vishwavidyalaya, Indore, India.*
[2]*Department of Atomic Energy, Raja Ramanna Centre for Advanced Technology, Govt. of India, Indore, India.*
[3]*Department of Computer Science and Engineering, Jaypee University of Engineering and Technology, Guna, India.*
[4]*School of Computer Science and Information Technology, Devi Ahilya Vishwavidyalaya, Indore, India.*

[*]*Corresponding Author : snc711@gmail.com*

Received: 02 February 2024 Revised: 06 March 2024 Accepted: 20 March 2024 Published: 08 April 2024

Abstract *- Network Intrusion Detection Systems (NIDS) are enhanced and updated consistently, but at the same the, network intruders and hackers are also modernizing and renovating their methodologies. Hence, it is very important to develop novel Intrusion Detection Systems which is constructive to deal with heterogeneous network attacks. Recent research indicates that the Intrusion Detection Systems powered by Machine Learning techniques are capable of curbing these issues up to a great extent but still, there is a long way to go. There are several distinguished models and algorithms exist which are capable of detecting network attacks. Most of the existing research is focused on building a robust system against common and prevalent network attack categories. These approaches do not extend to some peculiar and menacing network attacks, which are often encrypted to spoof the Intrusion Detection Systems. Hence, we have proposed an effective Decision Tree Model which is capable of detecting such attacks with nearly 100% accuracy. We have also investigated and presented a comparative study of more than 10 machine learning models using one of the latest datasets, the HIKARI-2021 [1] dataset. Moreover, the existing research work, particularly dealing with encrypted attacks, does not explicitly indicate the detection accuracy of the encrypted network attack category. Hence, we have also worked on individual network attack categories for various machine-learning approaches.*

Keywords *- Encrypted Network Attack, Network Intrusion Detection System (NIDS), Decision Tree Algorithm, Machine Learning, Cyber Security.*

1. Introduction

A Network Intrusion Detection System (NIDS) is a security mechanism that observes and analyses the traffic flowing through a network for any indications of unauthorized or harmful activity. By scrutinizing the network packets, NIDS can recognize patterns or actions that may suggest a security breach. NIDS is designed to recognize and prevent unauthorized entry, data breaches, malware attacks, and other forms of cyber-attacks that could potentially compromise the network and its assets. In Sekar et al.'s [2] research paper, they indicate that an efficient NIDS must have the capacity to detect various types of attacks with a high degree of accuracy in real time while minimizing false positives. Furthermore, the system should be scalable and able to manage high volumes of network traffic. Shun and Malki [3] propose a unique method which uses neural networks to boost the NIDS's accuracy and speed. Meanwhile, Sultana et al. [4] recommend machine learning techniques in SDN-based NIDS to improve its ability to detect and react to network threats. Moreover, a well-designed NIDS should have the ability to detect an extensive range of attacks, reduce false positives, scale well, and incorporate advanced technologies such as neural networks and machine learning to enhance its performance.

Machine learning has a crucial role in improving the efficacy of NIDS. Sinclair et al. [5] demonstrate an early example of applying machine learning techniques to NIDS through a Decision Tree algorithm used to identify network connections and classify them based on their behavior. Supervised machine learning algorithms have become increasingly popular in recent years for NIDS, with Taher et al. [6] using a supervised machine learning algorithm with feature selection to enhance the accuracy of NIDS. The

authors used a mutual information-based approach to select the most relevant features and then applied a Random Forest classifier to classify network traffic. Furthermore, Sommer and Paxson [7] proposed using unsupervised machine-learning techniques to detect network anomalies. The authors suggested that unsupervised machine learning algorithms could overcome the limitations of signature-based detection methods. In conclusion, machine learning in NIDS has the potential to improve intrusion detection accuracy efficiency and provide better protection against evolving network threats.

Encrypted network attacks involve the use of encryption to hide malicious activities in network traffic, making them challenging to detect and prevent. As noted in Al-Hababi and Tokgoz's [8] research paper, these attacks are often carried out using man-in-the-middle techniques, which involve intercepting and altering network traffic. These attacks are difficult to detect due to their use of encryption to conceal payloads, obfuscation of network traffic patterns, and the need to analyze large volumes of encrypted traffic in real time. To address these challenges, researchers have proposed using machine learning algorithms for encrypted network traffic analysis. For instance, Conti et al. [9] developed a machine learning-based framework for analysing Android encrypted network traffic to identify user actions. Shen et al. [10] conducted a comprehensive survey of machine learning-powered encrypted network traffic analysis, which identified the benefits and limitations of different techniques and suggested avenues for future research.

2. Related Work

The authors in [11] proposed a new approach introduced for network anomaly intrusion detection by utilizing a Dual Intrusion Detection System (Dual-IDS) that combines the outputs jof two machine learning models. The first model uses Gradient Boosting Decision Trees (GBDT) trained on raw network packet data. In contrast, the second model employs a bagging ensemble of GBDT models trained on statistical features extracted from the packet data. The authors propose that their Dual-IDS approach can offer better accuracy and robustness than individual models, particularly when confronted with adversarial attacks. To evaluate the performance of their proposed model, the authors used the Hikari-2021 [1] dataset. The proposed model achieved an accuracy, precision, recall, and F1-score of 99.91%.

There are various limitations of the proposed model in detecting the Hikari-2021 dataset classification. The model focuses mainly on binary classification, which categorizes normal and attack data only. However, the Hikari-2021 dataset comprises various types of attacks, and this limitation may affect the model's accuracy and effectiveness in detecting different types of attacks. Secondly, the proposed model uses feature selection techniques that may not be optimal for the Hikari-2021 dataset's encrypted traffic data. As a result, the model may not extract relevant features from the encrypted data, leading to reduced accuracy. Also, the model's performance may be impacted by the dataset's imbalance, with certain attack types being significantly underrepresented.

The other prominent work is done in [12], where the authors analyzed the performance of various machine learning algorithms in detecting network intrusions using the HIKARI- 2021 dataset. According to the study [12], the Random Forest algorithm exhibited superior performance in terms of accuracy, precision, and F1-score for detecting attacks when compared to other algorithms, including the K-Nearest Neighbors Algorithm, Multilayer perceptron and Support Vector Machine. Interestingly, the study discovered that utilizing all 86 features in the dataset did not necessarily result in optimal performance. Instead, feature selection techniques were found to be beneficial in reducing the number of features while maintaining high detection accuracy. Additionally, the study revealed that the algorithm's performance varied depending on the type of attack, with certain algorithms proving more effective for specific types of attacks than others. The authors emphasized the importance of using feature selection techniques and selecting appropriate machine learning algorithms for different types of attacks to achieve optimal detection accuracy.

However, the study [12] focused solely on the performance of four machine learning algorithms, which may not provide a comprehensive representation of all available algorithms. Moreover, the study failed to investigate the potential impact of varying hyperparameters on the performance of each algorithm or to examine their interpretability. Interpretability is crucial in understanding how algorithms arrive at their predictions and enhancing transparency and trustworthiness. Additionally, the study neglected to examine the influence of different preprocessing techniques on the algorithm's performance.

The current research on detecting network attacks in the Hikari-2021 dataset has certain limitations, highlighting the need for a new approach. Decision Tree models are a promising solution due to their effectiveness in other anomaly detection applications. These models can handle large feature sets, making them suitable for comprehensive network traffic analysis. Additionally, they can detect patterns across multiple layers of network traffic, which is particularly useful for complex attacks. Moreover, Decision Tree models can handle both static and dynamic data, including encrypted traffic, further enhancing their ability to detect anomalies. A Decision Tree-based IDS approach can potentially address the limitations of existing studies and provide a more comprehensive and accurate detection of network attacks in the Hikari-2021 dataset.

3. HIKARI-2021 Dataset

The Hikari-2021 dataset consists of two parts: real network traffic and synthetic encrypted traffic, each having a specific function in the area of network intrusion detection. The real traffic data were obtained from the Japan Coast Guard network during their regular maritime patrol, resulting in roughly 3 million network packets captured in a month. On the other hand, the synthetic encrypted traffic was produced using the BotNET simulation tool, resulting in almost 1 million network packets. Both datasets are annotated with information regarding the type of attack, such as DDoS, brute force, and SQL injection attacks, that is either simulated or observed. The dataset is publicly accessible and can be used for designing and assessing intrusion detection systems, particularly those with the ability to handle encrypted traffic.

The Hikari-2021 dataset offers several advantages over existing network intrusion detection datasets. Firstly, it combines real and synthetic encrypted traffic data, providing a more precise representation of actual network traffic. This is in contrast to other datasets that rely solely on synthetic data, which may not accurately capture the complexity and diversity of real-world network traffic. Secondly, the dataset is labeled with attack information, making it easier to evaluate intrusion detection systems and refine them accordingly. Thirdly, it is publicly available, promoting transparency and encouraging collaboration among researchers and developers. Lastly, the Hikari-2021 dataset is specifically designed for encrypted traffic, a vital feature given the increasing prevalence of encryption in modern networks. Overall, the Hikari-2021 dataset is a comprehensive, representative, and specifically designed intrusion detection dataset with labeled information, which can assist in the development of more accurate and effective intrusion detection systems.

KDD99 [13] has the highest number of records, followed by UNSW-NB15 [14], CICIDS-2017 [15], and Hikari-2021. However, Hikari-2021 has the highest number of features among the four datasets. KDD99 and UNSW-NB15 focus on intrusion detection for networks, while CICIDS-2017 is designed for detecting intrusions in industrial control systems, and Hikari-2021 is specifically designed for encrypted traffic. KDD99 and UNSW-NB15 have imbalanced datasets, while CICIDS-2017 and Hikari-2021 have more balanced datasets.

All four datasets have labeled data, but Hikari-2021 and CICIDS-2017 provide more detailed labeling information. Hikari-2021 is the only dataset that includes both real and synthetic data for a more accurate representation of real-world network traffic, while the other datasets rely solely on synthetic data. In summary, each dataset has its own strengths and weaknesses, but Hikari-2021 is exceptional for its detailed labeling information, the combination of real and synthetic data, and comprehensive representation of encrypted network traffic.

The Hikari-2021 dataset includes various classifications of network traffic for assessing the effectiveness of intrusion detection systems. The ”Background” classification includes ordinary network traffic, such as browsing and emailing, that is not associated with any malicious activity. The ”Benign” classification consists of non-threatening traffic, such as traffic related to software updates and network management. The ”Bruteforce” classification involves traffic linked to brute-force attacks, where an attacker tries to gain access to a system by guessing passwords. The ”Bruteforce-XML” classification specifically targets XML-based services and may involve traffic related to XML injection attacks or other forms of XML-based attacks. The ”Probing” classification pertains to traffic linked to reconnaissance or probing activities where an attacker attempts to gather information about a target network or system. The ”XMRIGCC Crypto Miner” classification is associated with network traffic related to the XMRIGCC cryptocurrency miner, a type of malware that mines cryptocurrency using a victim’s computer. By utilizing a diverse set of traffic types, the Hikari-2021 dataset offers a more inclusive and representative dataset for testing intrusion detection systems.

The dataset Hikari-2021 encompasses 86 characteristics that aid in developing intrusion detection systems and represent network traffic data. These features can be broadly classified into different groups, including statistical features (e.g., standard deviation of packet length, mean packet length, etc.), general network traffic features (e.g., number of packets, packet length, etc.), and protocol-specific features (e.g., DNS query type, HTTP request method, etc.). Some specific features of the dataset are the total bytes and packets in a flow, the ratio of incoming to outgoing packets, the protocol of the flow, the DNS query type, the HTTP request method, and many more. With this comprehensive set of data points, intrusion detection systems can be trained and assessed, and network security measures’ accuracy and effectiveness can be improved.

4. Proposed Decision Tree Model

A Decision Tree [16] is a structured model that follows a hierarchical arrangement comprising nodes and edges. Nodes in the tree represent evaluations of specific attributes of objects being classified, and edges indicate the potential outcomes of those evaluations. Each internal node in the tree corresponds to a decision point about an attribute, while each leaf node represents a class or probability distribution for that class. The decision-making process begins at the top of the tree (root node) and continues downwards until a leaf node is reached. At this point, the object being classified is given a class label. The process of constructing a Decision Tree involves creating a tree from a set of training examples that optimally divides the data into homogeneous regions based on attribute values. Decision Trees are non-parametric models since they do not require any prior assumptions about the data distribution or the functional form of the decision

boundary. Decision Trees [17] are popular due to their interpretability, ease of use, and high accuracy. Decision Trees can handle both categorical and continuous data and can be used for both classification and regression tasks. They are non-parametric models, meaning they do not require any prior assumptions about the data distribution or decision boundary. Additionally, Decision Trees can handle missing values and are resistant to outliers.

In our approach, the Decision Tree model begins by encoding the categorical variable 'Traffic Category' using LabelEncoder and then splits the data into train and test sets, with a test size of 0.25 and a random state of 42. The training data has 416458 observations, while the test data has 138820 observations. This Decision Tree model was created using the Decision Tree Classifier from the scikit-learn library. The maximum depth of the Decision Tree was controlled by setting the max depth parameter to 3. To ensure the reproducibility of the results, the random state parameter was set to 42, which set the random seed.

The model was trained on the train data using the fit function, and then predictions were generated on the test data using the predict function. These predictions were used to evaluate the performance of the Decision Tree model on the test data.

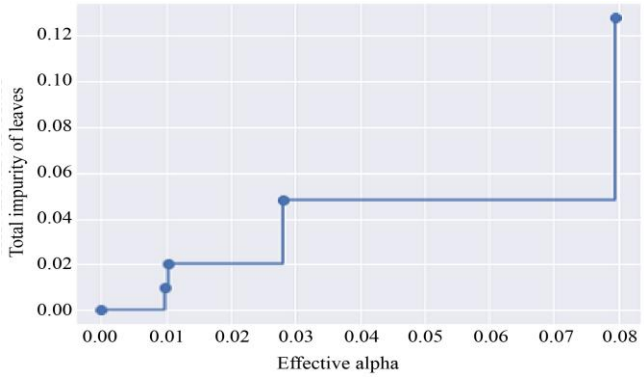

Fig. 1 Total Impurity vs Alpha for Training Set Effective Decision Tree (Our Approach)

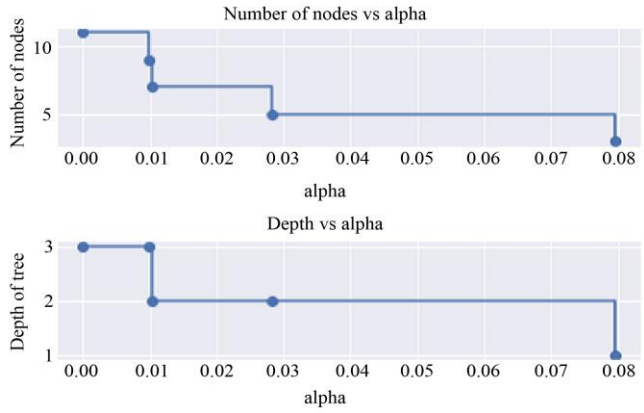

Fig. 2 Decreasing Trend of Alpha hyperparameter of Effective Decision Tree (Our Approach)

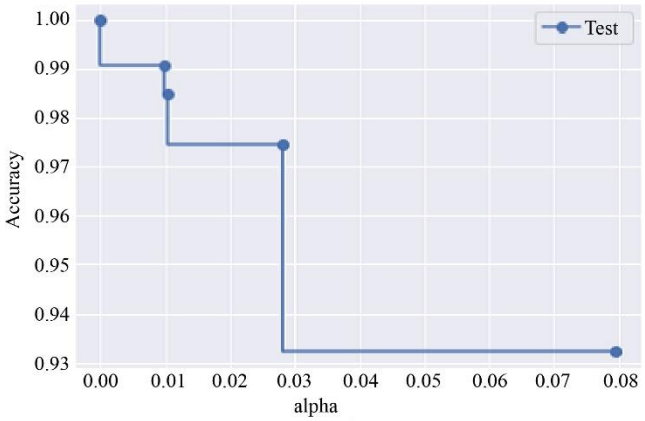

Fig. 3 Accuracy vs Alpha for Testing set of Effective Decision Tree (Our Approach)

The Decision Tree model comprises several parameters, each with its own specific role. These parameters were carefully set to ensure optimal performance of the model. The alpha parameter controls the complexity of the Decision Tree by setting the minimum value of cost-complexity pruning. In this model, the value is set to 0.0, indicating that no pruning is applied. The class weight parameter adjusts the weights of the classes in the dataset to balance the impact of the classes with fewer instances. The criterion parameter determines the function to measure the quality of the split, which in this model is set to gini. The max depth parameter controls the maximum depth of the Decision Tree and is set to 3, while the max features parameter controls the number of features to consider and is set to None.

The max-leaf nodes parameter controls the maximum number of leaf nodes that the Decision Tree can have and is set to None. The min impurity decrease parameter sets the minimum threshold for the impurity decrease of a node to split and is set to 0.0. The min samples leaf parameter sets the minimum number of samples required to be at a leaf node and is set to 1, while the min samples split parameter sets the minimum number of samples required to split an internal node and is set to 2. The min weight fraction leaf parameter sets the minimum fraction of the sum total of weights required to be at a leaf node and is set to 0.0. The random state parameter sets the random seed for the Decision Tree, ensuring reproducibility of the results. In contrast, the splitter parameter determines the strategy to choose the split at each node and is set to best. By using the above parameters in the Decision Tree model, the most optimal results were achieved compared to other hyperparameter tuning techniques.

The Decision Tree structure consists of 11 nodes, each representing a specific condition or action. The tree starts at node 0, which is a split node with two possible paths. If the condition X[:, 0] <= 181179.5 is true, the tree goes to node 1; otherwise, it goes to node 6. Node 1 is another split node, and depending on the condition X[:, 0] <= 11028.5, it goes to

either node 2 or node 5. Similarly, node 2 is also a split node, and based on the condition X[:, 0] <= 5144.0, it goes to node 3 or node 4. Nodes 3 and 4 are leaf nodes representing a final outcome or decision.

Similarly, node 5 is also a leaf node. Node 6 is another split node, and based on the condition X[:, 0] <= 528610.5, it goes to node 7 or node 8. Node 7 is a leaf node, and node 8 is another split node with two possible paths based on the condition X[:, 33] <= 13.8023. Finally, nodes 9 and 10 are leaf nodes representing the final outcome or decision. The Decision Tree algorithm utilizes the Gini index to minimize impurity while recursively searching for the optimal feature and threshold values that split the data into two subsets. The algorithm selects the feature and threshold values that can achieve maximum reduction in the Gini index by trying all possible combinations. For instance, node 0 uses the first feature (X[:, 0]) as the splitting criterion and has a threshold value of 181179.5. Nodes 1, 2, and 6 also use the first feature but have different threshold values of 11028.5, 5144.0, and 528610.5, respectively. Node 9, on the other hand, splits the data based on the 34th feature (X[:, 33]) and has a threshold value of 13.8023. The Decision Tree algorithm continues to recursively split the child nodes using the same process until it reaches the maximum depth or a stopping criterion. By selecting the feature and threshold values that minimize impurity, the algorithm constructs a tree that can predict the target variable by traversing the tree from the root to a leaf node.

Cost complexity pruning is a technique commonly used to avoid overfitting and enhance the generalization ability of Decision Trees. This technique involves adding a penalty term to the impurity reduction criterion, such as the Gini index or entropy, to consider the tree's complexity. The complexity parameter, also called alpha, balances the model complexity and goodness of fit to the training data. The tree is initially grown to its maximum size and then pruned back using a bottom-up approach. At each internal node, the subtree is removed if the impurity reduction is not significant or is overshadowed by the penalty term. The pruning process continues for all internal nodes until the desired level of pruning is achieved based on alpha. Cross-validation can be used to select the optimal alpha value that strikes a balance between model complexity and prediction accuracy.

In summary, cost complexity pruning is a powerful yet straightforward approach that can improve Decision Trees' generalization performance and mitigate overfitting risks. The graph in Figure 1 illustrates the correlation between the total impurity and the effective alpha parameter in the Decision Tree classifier with the cost complexity pruning technique. The graph demonstrates a decreasing pattern, which indicates that the total impurity decreases as the effective alpha increases. The effective alpha parameter acts as a regularization parameter that balances the accuracy and complexity of the model.

An increase in the effective alpha simplifies the model, thereby decreasing the total impurity. The graph also depicts an optimum point of the effective alpha parameter, beyond which the model's performance significantly decreases. This optimum value signifies the highest accuracy point of the Decision Tree model, which is neither too complex nor too simple. Hence, this graph helps determine the optimal value of the effective alpha parameter, which is essential to achieve the best performance of the model on unseen data. Similarly, the graph in Figure 2 displays a decreasing pattern, indicating that the number of nodes decreases as the effective alpha increases.

The effective alpha parameter controls the trade-off between model complexity and accuracy, resulting in a simpler model as the effective alpha increases. This graph is utilized to determine the optimal effective alpha parameter value, which is crucial for achieving the best performance of the model on new data. By selecting the optimal value of alpha, we can balance the model's complexity and accuracy and avoid overfitting or underfitting. The graph in Figure 3 illustrates the relationship between the effective alpha parameter and the accuracy of the Decision Tree model on the test data. This graph helps to determine the ideal effective alpha value, which plays a crucial role in achieving the best possible performance of the model on unseen data.

5. Experimental Outcome and Discussion

The accuracy of a machine learning model is a measure of its ability to correctly predict the target variable based on the input features. As mentioned earlier, our approach has achieved 100% accuracy, indicating that it was able to correctly classify all samples in the dataset. This can be attributed to the Decision Tree's ability to capture complex relationships between the features and the target variable. In Table I, the accuracy of various machine learning models is compared, and our approach is shown to be the most optimal. Also, Figure 4 depicts the confusion matrix of our Decision Tree approach. The K Neighbors Classifier achieved a high accuracy of 98.21%. This algorithm works by classifying a sample based on the class labels of its k nearest neighbors in the feature space. The high accuracy suggests that the samples in the dataset are well separated in the feature space and that the majority of the samples have similar features. The Extra-Tree Classifier achieved a 96.7% accuracy by building a large number of Decision Trees and selecting the best split among a random subset of features at each node.

The high accuracy indicates that the features in the dataset have strong predictive power, and the Decision Tree ensemble can capture this information effectively.

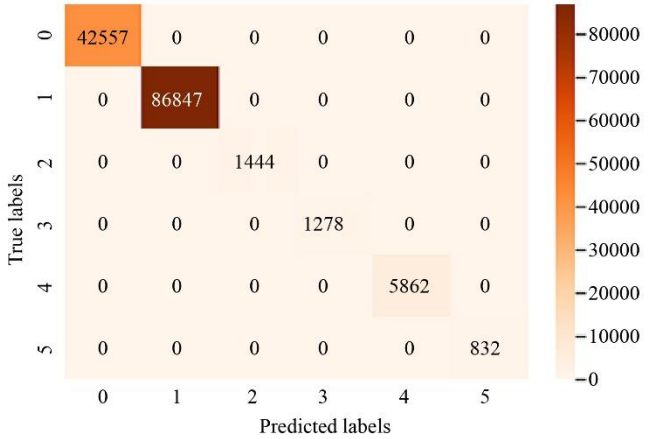

Fig. 4 Confusion Matrix of Effective Decision Tree (Our Approach)

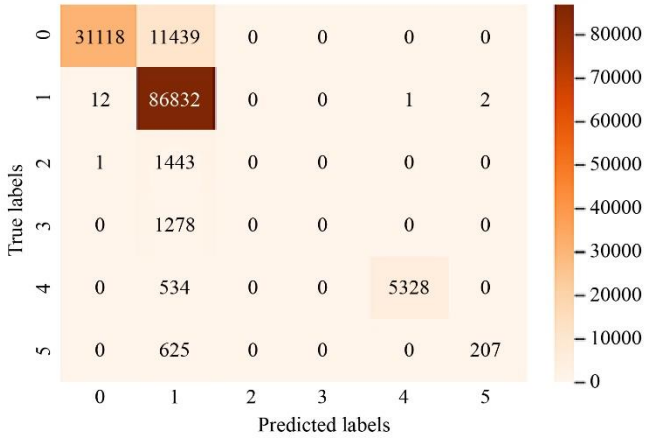

Fig. 5 Confusion Matrix of Multi Layer Perceptron Classifier

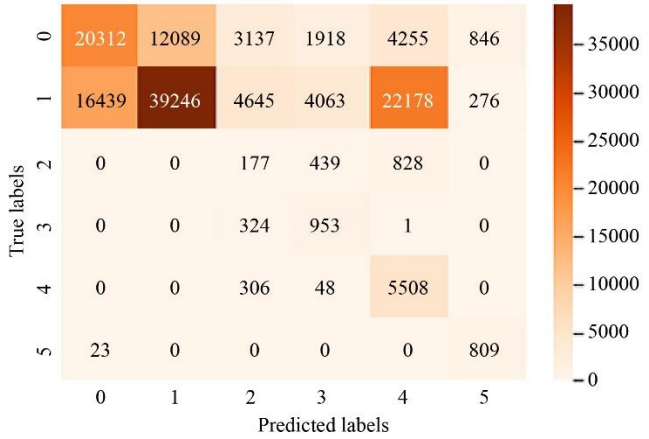

Fig. 6 Confusion Matrix of Bernoulli naive Bayes classifier

The AdaBoost Classifier achieved a 94.73% accuracy. The high accuracy suggests that the dataset contains a large number of informative features that can be used to build a strong classifier. The Ridge Classifier achieved a 91.81% accuracy by fitting a linear model to the data using L2 regularization. The high accuracy suggests that the features in the dataset have a linear relationship with the target variable. The MLP Classifier achieved an 89.58% accuracy by fitting a neural network model to the data. The relatively lower accuracy suggests that the dataset may not contain enough samples or features to train a complex neural network model effectively.

Figure 5 introduces the confusion matrix of this approach. The Passive Aggressive Classifier achieved an 86.24% accuracy by fitting a linear model to the data using an online learning algorithm. The relatively lower accuracy suggests that the dataset may contain noisy or irrelevant features that are affecting the performance of the model. The SGD Classifier achieved an 81.43% accuracy by fitting a linear model to the data using stochastic gradient descent. The lower accuracy suggests that the features in the dataset may not have a strong linear relationship with the target variable.

The Gaussian Naive Bayes Classifier achieved a 74.67% accuracy by assuming that the features are independent and have a Gaussian distribution. The lower accuracy suggests that the assumption of independence may not hold for the features in the dataset. The Complement Naive Bayes Classifier achieved a 55.81% accuracy by assuming that the complement of each feature has a multinomial distribution. The relatively low accuracy suggests that the assumption of the multinomial distribution may not hold for the features in the dataset. The Bernoulli Naive Bayes Classifier achieved a 48.26% accuracy by assuming that the features are binary and have a Bernoulli distribution.

The low accuracy suggests that the assumption of binary features may not hold for the features in the dataset. Figure 6 depicts the confusion matrix of this approach. As part of our work, we also analyzed the performance of each approach in detecting traffic categories. Performance metrics of various machine learning approaches on all network categories, including encrypted networks, are presented in Figures 7, 8 and 9. The Decision Tree classifier performs exceptionally well on all traffic categories with flawless precision, recall, and f1-score. This is because Decision Trees are adept at capturing non-linear connections between features and class labels. Hence, it can divide the data into smaller segments and make decisions based on those divisions, which ultimately results in extremely accurate predictions.

Gaussian Naive Bayes works well on benign, brute-force, brute-forcexml, and XMRIGCC Crypto Miner traffic categories but has a relatively lower f1-score on the background and probing categories. Gaussian Naive Bayes is a probabilistic model that supposes the features are autonomous of each other. Therefore, the model may not be able to comprehend complex relationships between the features, leading to lower performance in certain traffic categories. The K Neighbors Classifier performs well in detecting benign and Background traffic categories, achieving precision and recall scores of around 0.98-0.99.

Decision Tree Classifier (Our Approach)			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	1.00	1.00	1.00
Background	1.00	1.00	1.00
Probing	1.00	1.00	1.00
Bruteforce	1.00	1.00	1.00
Bruteforce-XML	1.00	1.00	1.00
XMRIGCC Crypto Miner	1.00	1.00	1.00
K Neighbors Classifier			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	0.98	0.97	0.97
Background	0.98	0.99	0.99
Probing	0.93	0.93	0.93
Bruteforce	0.95	0.96	0.96
Bruteforce-XML	0.99	0.98	0.99
XMRIGCC Crypto Miner	1.00	1.00	1.00
Extra-Tree Classifier			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	0.98	0.98	0.98
Background	0.98	0.99	0.99
Probing	0.98	0.99	0.98
Bruteforce	1.00	1.00	1.00
Bruteforce-XML	1.00	1.00	1.00
XMRIGCC Crypto Miner	1.00	1.00	1.00
AdaBoost Classifier			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	0.97	1.00	0.98
Background	0.94	1.00	0.97
Probing	0.00	0.00	0.00
Bruteforce	1.00	1.00	1.00
Bruteforce-XML	0.00	0.00	0.00
XMRIGCC Crypto Miner	1.00	1.00	1.00

Fig. 7 Performance metrics for the decision tree (Our Approach), K Nearest Neighbors, Extra-Tree and Adaboost Classifier

Ridge Classifier			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	0.91	0.96	0.93
Background	0.92	0.98	0.95
Probing	0.00	0.00	0.00
Bruteforce	0.00	0.00	0.00
Bruteforce-XML	1.00	0.22	0.36
XMRIGCC Crypto Miner	0.00	0.00	0.00
MLP Classifier			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	1.00	0.73	0.84
Background	0.85	1.00	0.92
Probing	0.00	0.00	0.00
Bruteforce	0.00	0.00	0.00
Bruteforce-XML	1.00	0.91	0.95
XMRIGCC Crypto Miner	0.99	0.25	0.40
Passive Aggressive Classifier			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	0.98	0.79	0.87
Background	0.86	0.99	0.92
Probing	0.00	0.00	0.00
Bruteforce	0.00	0.00	0.00
Bruteforce-XML	0.95	0.71	0.81
XMRIGCC Crypto Miner	0.00	0.00	0.00

Fig. 8 Performance metrics for the ridge, MLP and passive aggressive classifier

Gaussian Naive Bayes			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	0.55	0.91	0.69
Background	0.99	0.64	0.78
Probing	0.32	1.00	0.48
Bruteforce	0.92	0.99	0.96
Bruteforce-XML	1.00	0.99	1.00
XMRIGCC Crypto Miner	1.00	1.00	1.00
Complement Naive Bayes			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	0.48	0.20	0.28
Background	0.68	0.79	0.73
Probing	0.00	0.00	0.00
Bruteforce	0.00	0.00	0.00
Bruteforce-XML	0.02	0.08	0.04
XMRIGCC Crypto Miner	0.00	0.00	0.00
Bernoulli Naive Bayes			
Traffic Category	**Precision**	**Recall**	**F1-Score**
Benign	0.55	0.48	0.51
Background	0.76	0.45	0.57
Probing	0.02	0.12	0.04
Bruteforce	0.13	0.75	0.22
Bruteforce-XML	0.17	0.94	0.29
XMRIGCC Crypto Miner	0.42	0.97	0.59

Fig. 9 Performance metrics for the gaussian, Complement and Bernoulli Naive Bayes classifiers

It also performs well in detecting Probing, Bruteforce, and Bruteforce-XML traffic categories, achieving an f1-score of 0.93-0.99. However, it has a low recall score in detecting XMRIGCC Crypto Miner traffic, indicating that the classifier is not very good at detecting this type of traffic.

Similarly, the Extra Tree Classifier performs well in detecting all traffic categories, achieving precision and recall scores of 1.00 in most cases. This indicates that the classifier is very accurate in detecting different traffic categories. The AdaBoost Classifier performs well in detecting benign and Bruteforce traffic categories, achieving a precision score of 1.00. However, it performs poorly in detecting Probing, Bruteforce-XML, and XMRIGCC Crypto Miner traffic categories, with an f1-score of 0.00, which indicates that the classifier is not able to classify these traffic categories correctly.

The Multilayer Perceptron Classifier also performs well in detecting benign traffic with a precision of 1.00 and an f1-score of 0.84. However, it performs poorly in detecting Probing and Bruteforce traffic categories with an f1-score of 0.00, which indicates that the classifier is not able to classify these traffic categories correctly. It also has low recall in detecting XMRIGCC Crypto Miner traffic. On the other hand, Stochastic Gradient Descent (SGD) performs well in the background traffic category, but it has a low f1-score in all other traffic categories. The inadequate performance can be attributed to the fact that SGD is a linear classifier and may not be able to comprehend complex relationships between the features, particularly for non-linearly separable data.

Moreover, the Passive Aggressive Classifier works relatively well on benign and background traffic categories but has a low f1-score on all other traffic categories. This classifier is a type of online learning algorithm which can adapt quickly to new data. However, this classifier may not be appropriate for this dataset since the dataset is not continuously changing. Gaussian Naive Bayes works well on benign, brute-force, brute-force-xml, and XMRIGCC Crypto Miner traffic categories but has a relatively lower f1-score on the background and probing categories. Gaussian Naive Bayes is a probabilistic model that supposes the features are autonomous of each other. Therefore, the model may not be able to comprehend complex relationships between the features, leading to lower performance in certain traffic categories. Complement Naive Bayes performs poorly on all traffic categories, with low precision, recall, and f1-score. This is because the Complement Naive Bayes algorithm is intended to work well with imbalanced datasets where the number of samples in one class is significantly larger than in the other classes. In this case, the dataset is not imbalanced, and the algorithm is not suitable for this type of dataset. Bernoulli Naive Bayes works well on brute-force-xml and XMRIGCC Crypto Miner traffic categories, but it has a low f1-score on other traffic categories, particularly benign and background. Bernoulli Naive Bayes assumes binary features and is typically used for text classification tasks. This classifier may not be appropriate for this type of dataset since the features are not binary.

6. Conclusion

We assessed and studied various machine learning classifiers, including Decision Tree, Gaussian Naive Bayes, Complement Naive Bayes, Bernoulli Naive Bayes, Stochastic Gradient Descent, and Ridge Classifier, to detect network attacks from the Hikari-2021 dataset. Our Decision Tree classifier outperforms all other classifiers, demonstrating flawless precision, recall, and f1-score on all traffic categories. As such, we can conclude that the Decision Tree classifier is one of the most superior classifiers for detecting network attacks from the Hikari-2021 dataset, owing to its ability to capture non-linear connections between features and class labels and segment data to make decisions, leading to highly accurate predictions. In future work, we can consider studying and exploring more complex machine learning models, like neural networks, to further enhance the accuracy of our approach. Moreover, combining multiple classifiers could improve overall system performance. We could also evaluate the performance of these classifiers on other datasets to assess their generalizability. Finally, we could investigate the feasibility of implementing these classifiers in real-time network monitoring systems for identifying and mitigating network attacks in real-time.

References

[1] Andrey Ferriyan et al., "Generating Network Intrusion Detection Dataset Based on Real and Encrypted Synthetic Attack Traffic," *Applied Sciences*, vol. 11, no. 17, pp. 1-17, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] R. Sekar et al., "A High-Performance Network Intrusion Detection System," *Proceedings of the 6th ACM conference on Computer and Communications Security*, pp. 8-17, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[3] Jimmy Shun, and Heidar A. Malki, "Network Intrusion Detection System Using Neural Networks," *2008 Fourth International Conference on Natural Computation*, Jinan, China, pp. 242-246, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[4] Nasrin Sultana et al., "Survey on SDN Based Network Intrusion Detection System Using Machine Learning Approaches," *Peer-to-Peer Networking and Applications*, vol. 12, pp. 493-501, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[5] C. Sinclair, L. Pierce, and S. Matzner, "An Application of Machine Learning to Network Intrusion Detection," *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*, Phoenix, AZ, USA, pp. 371-377, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[6] Kazi Abu Taher, Billal Mohammed Yasin Jisan, and Mahbubur Rahman, "Network Intrusion Detection Using Supervised Machine Learning Technique with Feature Selection," *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, Dhaka, Bangladesh, pp. 643-646, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Robin Sommer, and Vern Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," *2010 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, pp. 305-316, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[8] Abdulrahman Al-Hababi, and Sezer C. Tokgoz, "Man-in-the-Middle Attacks to Detect and Identify Services in Encrypted Network Flows Using Machine Learning," *2020 3rd International Conference on Advanced Communication Technologies and Networking (CommNet)*, Marrakech, Morocco, pp. 1-5, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[9] Mauro Conti et al., "Analyzing Android Encrypted Network Traffic to Identify User Actions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114-125, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[10] Meng Shen et al., "Machine Learning-Powered Encrypted Network Traffic Analysis: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 791-824, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Maya Hilda Lestari Louk, and Bayu Adhi Tama, "Dual-IDS: A Bagging-Based Gradient Boosting Decision Tree Model for Network Anomaly Intrusion Detection System," *Expert Systems with Applications*, vol. 213, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Rui Fernandes, and Nuno Lopes, "Network Intrusion Detection Packet Classification with the HIKARI-2021 Dataset: A Study on ML Algorithms," *2022 10th International Symposium on Digital Forensics and Security (ISDFS)*, Istanbul, Turkey, pp. 1-5, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Salvatore Stolfo et al., *KDD Cup 1999 Data*, UCI Machine Learning Repository, 1999. [CrossRef] [Publisher Link]

[14] Nour Moustafa, and Jill Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)," *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, ACT, Australia, pp. 1-6, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[15] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy ICISSP*, Funchal, Madeira, Portugal, vol. 1, pp. 108-116, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[16] S.R. Safavian, and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660-674, 1991. [CrossRef] [Google Scholar] [Publisher Link]

[17] Philip H. Swain, and Hans Hauska, "The Decision Tree Classifier: Design and Potential," *IEEE Transactions on Geoscience Electronics*, vol. 15, no. 3, pp. 142-147, 1977. [CrossRef] [Google Scholar] [Publisher Link]

[18] Zeeshan Ahmad et al., "Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, pp. 1-29, 2021. [CrossRef] [Google Scholar] [Publisher Link]