Leveraging Artificial Intelligence to Secure Wireless Network: Exploring Threats, Existing Approaches, and Proposed Mitigation Strategies

Alice Lee Nan Xin
Department of Computing and Information Systems
Sunway University
Selangor, Malaysia
21067848@imail.sunway.edu.my

Athirah Mohd Ramly
School of Architecture, Computing and Engineering, UEL
London, United Kingdom
a.mohd-ramly@uel.ac.uk

Mhd Saeed Sharif
Intelligent Technologies Research Group, School of Architecture, Computing and Engineering, UEL
London, United Kingdom
s.sharif@uel.ac.uk

Mehran Behjati
Department of Computing and Information Systems
Sunway University
Selangor, Malaysia
mehranb@sunway.edu.my

Abstract— **The exponential growth of network has introduced new Internet-of-Things (IoT) use cases that has enabling us convenience and comfort. The surge of IoT devices due to the capabilities brought by fifth generation (5G) have given rise to security threats and attacks, particularly malware attacks IoT botnets have been an alarming issue, where smart devices can be manipulated by malicious actors to commence subsequent attacks such as Denial of Service (DoS). Traditional and complex security techniques may not be a viable solution towards these resource-constrained devices with limited processing power. Machine Learning techniques (ML) are the rising trend, and it is often used in Intrusion Detection Systems and Network Anomaly Detection. This paper emphasizes on analyzing and comparing various ML models on the IoT-23 dataset. It aims to predict anomalies and conclude the model with optimal performance and least computational time cost that can be used for network anomaly detection systems with real-time data in future works. The ML models used in this paper are Decision Trees (DT), K-nearest neighbours (KNN), Random Forest (RF), Naïve Bayes (NB) and Histogram Gradient Boosting (HGB). DT displayed the best performance with an accuracy score of 73% and F1 score of 0.49 with a time cost of 28.22 seconds.**

Keywords—5G, Artificial Intelligence, Cybersecurity, Internet-of-Things, Wireless Network

I. INTRODUCTION

As technology advances throughout the years, the world is transitioning into the fifth generation (5G) of wireless mobile communication technology. According to [1], a significant increase in number of devices in the network has been recorded and mobile traffic is expected to increase about 27% from the year 2019 to 2025 annually. With the increase of tremendous amount of data traffic, previous mobile communication technologies such as fourth generation/ Long Term Evolution (4G/4G LTE) are not capable of handling such requirements.

5G wireless systems not only provide traditional voice and data communication, but also it introduces new uses cases and industrial applications. With the roll out of 5G technology, high speed connectivity within a household allows electronic appliances to be connected and for devices to communicate between another which is a major application called Smart Homes. Additionally, another popular application of Internet of Things (IoT) in vehicular communications also known as Vehicular Networks can prevent accidents and collisions by transmitting information between cars through sensors [2].

New technologies, use cases and applications introduces new security issues and vulnerabilities which are critical to resolve. IoT devices [16], [17] are normally dependent on the internet and more devices are interconnected throughout the years especially with 5G. There exists various security and privacy issues as majority of IoT devices could store personal information of users such as wearables like smart watches potentially storing the location information and smart home devices being hacked where cyber attackers can spy or monitor on user's activities.

Moreover, many new advanced radio schemes that are realistic over 5G networks, which one of them are 5G New Radio (NR) developed by the 3GPP. The new technologies introduce new uses cases such as Device to Device (D2D), Machine to Machine (M2M), Internet of Vehicles (IoV) and Internet of Things (IoT) [2]. 5G is developed in a way to better implement the IoT with lower latency and energy consumption.

5G wireless systems not only provide traditional voice and data communication, but also it introduces new uses cases and industrial applications. The application of IoT and 5G is transforming the technological era and is used in broadly in different fields of intelligent applications. With the roll out of 5G technology, high speed connectivity within a household allows electronic appliances to be connected and for devices to communicate between another which is a major application called Smart Homes. Additionally, another popular application of IoT in vehicular communications also known as Vehicular Networks can prevent accidents and collisions by transmitting information between cars through sensors [2].

Furthermore, the obtained data produced by IoT devices are enormous in numbers and they have unique characteristics where traditional security approaches such as network segmentation may not be able to tackle the security of such devices [3]. Another concern is the resourceconstraint requirements of IoT (low energy consumption and limited computational power) that makes it challenging for the adoption of advanced and complex security technologies. Traditional security mechanisms often involve encryption techniques and methods with the use of complex mathematical operations which requires powerful computation. Due to such computational limitations, there is a need of optimized security solutions that can accommodate these requirements.

The prevalent deployment of massive numbers of devices in 5G networks increases the vulnerability footprint and given that IoT devices are commonly resource-constrained, it may not be viable to utilize advanced and complex security mechanisms that are computationally expensive to counter notable attacks. The objectives of this paper are as stated below:

i. To review and conduct concise research on existing security approaches and analyse the challenges of each approach.

ii. To investigate the threats and attacks faced by 5G IoT.

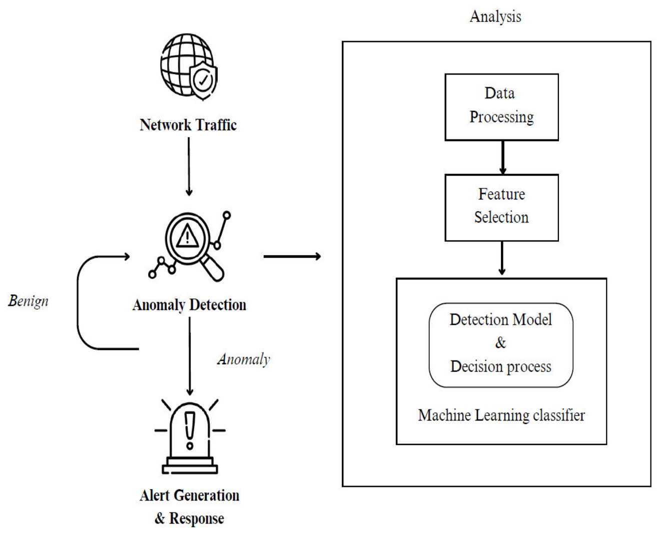
iii. To mitigate against threats and attacks faced using Machine Learning (ML) techniques

iv. To evaluate the performance metrics and the results using open-source tools.

The aim of this work is to utilize various ML models to detect network anomalies. Based on the analysed results, the ML model which has the most optimal performance and least computational cost required can be used in real world network anomaly detection systems.

II. METHODOLOGY

Fig.1 below shows a generic system architecture of a network anomaly detection system. Typically, network anomaly detection systems employ ML along with Artificial Intelligence (AI) to detect abnormal network behaviours or concealed threats such as sophisticated malware. Network attributes such as traffic intensity, protocols, bandwidth, source and destination Internet Protocol (IP) addresses, timestamps and more are tracked by these systems during real-time. An alert will be triggered in the event of malicious or abnormal network traffic being detected and this will notify relevant network security personnel to perform further mitigation. Therefore, the ML model with the best performance and least time cost in this study will be selected and be used to identify network anomalies for future works based on real network traffic captured.


Analysis
Network Traffic
Anomaly Detection
Benign
Anomaly
Alert Generation & Response
Data Processing
Feature Selection
Detection Model & Decision process
Machine Learning classifier
Fig. 1 Generic Anomaly Detection System

Supervised and Unsupervised learning are the two primary ML methods for anomaly detection however, based on the availability of a labelled dataset – IoT-23, Supervised ML will be used in this paper instead. The ML algorithms utilized are Decision Trees (DT), Random Forest (RF), K-nearest neighbours (KNN), Naïve Bayes (NB) and Histogram Gradient Boosting (HGB) which is an ensemble ML algorithm. Below displays a generic overview of how the methodology will be carried out and evaluated using the different types of ML models in this paper as shown in Fig.2.

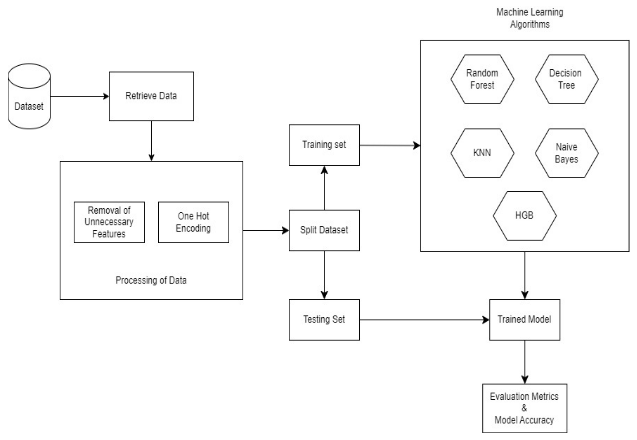


Fig. 2 Methodology Overview

The pre-processed dataset was obtained from an online community for data science and ML called Kaggle but it is based on the IoT-23 dataset obtained from [4]. The IoT-23 dataset contains 23 captures of diverse IoT network traffic from infected devices and was produced in the year 2020. Smart home devices were used during the capture and those devices are: Smart Low Emitting Diode (LED) Lamp (Philips HUE), Smart door lock (Somfy) and Amazon Echo home intelligent personal assistant, also known as “Alexa” in the modern days. The labels of the dataset were generated by the authors in [4] from the Stratosphere laboratory based on their manual evaluation of the network. The IoT-23 dataset was selected as it is a popular dataset which was publicly made available, and it comprises of numerous IoT traffic. Moreover, the dataset contains different types of common IoT threats and malware attacks which are relevant to this study such as Botnets (Mirai, Torii, Okiru) and DoS attacks which makes it a good fit for simulating and accessing the performance of different ML models.

However, the pre-processed version of the dataset obtained from Kaggle will be used instead. The reason is that, due to the constraints of current hardware settings for carrying out this project, loading all the 23 .pcap files from the original site and combining it into a .csv file requires complex work thus it would require more computational time whereas training, testing, and plotting necessary visual representation of the results of different ML models already consumes a significant amount of computational time. Thus, by using the pre-processed version of the IoT-23 dataset obtained from Kaggle, computational time can be saved, and it can be spent more on training the models, perform testing and plotting required visual representation for comparison instead.

The pre-processed version is already a .csv file which is extracted from the actual IoT-23 dataset, but the sampling numbers differ from the actual dataset thus the amount of data will vary. The number of labels for the pre-processed version of the dataset is shown in Table 1.

TABLE 1 Number of attacks of the pre-processed version of IoT23 dataset from Kaggle

Label	Number
Benign	3389036
C&C- Heartbeat	1313012
C&C- Torii	688812
DDoS	638506
Okiru	15286
Okiru-attack	1332
Attack	538
C&C-File Download	46
FileDownload	30
C&C	13
C&C-Mirai	8
C&C-HeartBeat-FileDownload	3
PartOfAHorizontalPortScan	1

The last step of data preparation is to split the data into training and testing to obtain a representation of the data points. The random state will be set to 42 so that the results will be the same each time it runs whereas the training and testing data will be divided to a 70:30 ratio split. The *.csv* file comprises of 6,046,623 entries in total.

Five ML models which are RF, DT, KNN, NB and HGB were selected to test and train on the IoT-23 dataset. Several libraries were used in this methodology and those are Scikit-learn, Pandas, numpy, matplotlib and seaborn. Scikit-Learn is a library in Python which provides a variety of tools used for ML tasks such as classification, clustering, and regression. Classification will be used to classify benign network behaviour amongst abnormal network behaviour. Meanwhile for the visualisation of results, it will be done using the matplotlib and seaborn libraries. The process is described as shown Fig. 3.

A. *Process Overview*

- Import the classification models (RF, SVM, DT, KNN, HGB) from scikit-learn library.
- Instantiate and fit the models upon the training data.
- Use the model's method to predict test data and determine model accuracy using Scikit-Learn's model.score () method instead of manually calculating the mean absolute error (mae). This method calculates the R^2 value.
- Determine the performance metrics by importing the classification_report() method from Scikit-Learn. This displays the relevant metrics to assess the effectiveness of each model.
- To visualise the performance of each classification model, the libraries: numpy, seaborn, matplotlib and confusion_matrix from Scikit-Learn were used.

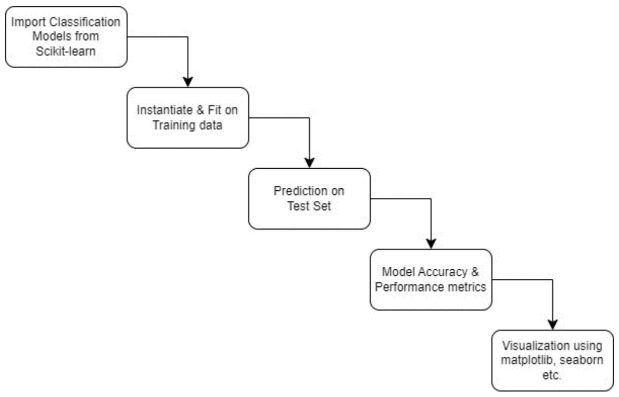

Import Classification Models from Scikit-learn
Instantiate & Fit on Training data
Prediction on Test Set
Model Accuracy & Performance metrics
Visualization using matplotlib, seaborn etc.

Fig. 3 *Process Overview*

B. *Supervised Learning*

The Supervised Learning is a learning method in ML where the models make use of labelled data to train on and make predictions. It can be categorized into two types which are classification and regression. In this paper, multiclass classification is used as there are more than two classes present. Five supervised learning models were chosen to be analysed upon the IoT-23 dataset. Due to limitations of current hardware settings, cross validation is not performed to prevent overfitting as the dataset is very large and it can be very computationally expensive.

1) *Random Forest (RF)*

Random Forest utilizes numerous DT to make decisions, and it is a flexible algorithm. It is typically applied in classification scenarios like fraud detection, anomaly detection and Distributed Denial of Services (DDoS) detection [6]. Based on [7], RF is proven to be advantageous in detecting anomalous network attacks. The algorithm tends to have a higher accuracy than DT however the trade-off is it requires more computation power and requires more time to train especially on a large dataset. Table 2 below depicts the hyperparameters configured in the RF Classifier:

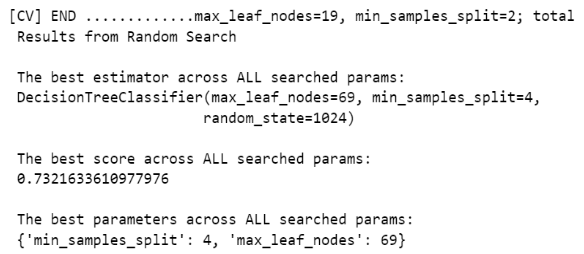
TABLE 2 *Parameters Configured for Random Forest*

Parameters	Explanation/Justification
n_estimators = 100	Refers to the number of trees. The default value 100 was used as any value more than that would cause the model to

	take longer time to train.

2) Decision Trees (DT)

Decision Trees is one of the fundamental supervised learning algorithms that utilizes a tree-like framework and divide the data into smaller samples to make decisions. The advantages of DT are that it is simple and easy to understand, and it is relatively faster in making predictions than other algorithms like RF. However, the downside is it is susceptible to overfitting. According to [6] and [7], DT is commonly utilized in Intrusion Detection and DDoS detection as a classifier. Random Search was initially performed on DT to optimize the hyperparameter as it is less computationally intensive than when GridSearch was performed. The best estimator and score were printed out based on figure 4, however the time cost based on using the parameters after performing Random Search was slightly higher than without performing Random Search as shown in Table 3.



```
[CV] END ............max_leaf_nodes=19, min_samples_split=2; total
 Results from Random Search

 The best estimator across ALL searched params:
 DecisionTreeClassifier(max_leaf_nodes=69, min_samples_split=4,
                        random_state=1024)

 The best score across ALL searched params:
 0.7321633610977976

 The best parameters across ALL searched params:
 {'min_samples_split': 4, 'max_leaf_nodes': 69}
```

Fig. 4 Decision Tree (Random Search)

TABLE 3 Decision Trees (Random Search)

	With Random Search	Without Random Search
Accuracy	0.73	
Time Cost	61.63 seconds	25.86 seconds

According to the source [8], performing random search does not assure that the best hyperparameters will be found. Thus, this paper moves forward without Random Search and the results for this algorithm will be displayed without making any adjustments towards the hyperparameters.

3) K-nearest neighbours (KNN)

KNN is perceived as one of the least complex ML algorithms in the supervised learning group. The algorithm presumes the similarity of two close data points tend to have the same label or class. The advantage of KNN is it is easy to execute, nevertheless the downside is it requires significant computational cost, and it can be time consuming due to calculation of the distance among data points. Moreover, the model is susceptible to overfitting. KNN uses a metric to calculate the distance and one of its default metrics is the Euclidean distance.

According to [6] and [7], the common applications of KNN in security are Intrusion prevention and detection, anomaly detection, pattern recognition and ransomware detection in IoT. Gridsearch to find the optimal tuning parameters was not performed in this model as it is too computationally expensive for current hardware settings. The following table 4 presents the hyperparameters configured in the KNN classifier:

TABLE 4 Parameters configured for K-Nearest Neighbors

Parameters	**Explanation/Justification**
n_neighbors = 5	Refers to the value k, the default value 5 was used as the higher the value of k, the less accurate the model will be.
Metric = 'minkowski'	Refers to the distance metric which in this case is derived from the Euclidean distance.
P = 2	Refers to the parameter for the distance metric.

4) Naïve Bayes (NB)

Naïve Bayes is a simple and commonly used algorithm especially for large datasets. It makes fast predictions and has the assumption that the features in a class are independent of each other. NB is built on the Bayes theorem which makes it a probabilistic classifier. The advantages of this algorithm are it is fast and simple and is efficient towards multi class classification. Nevertheless, the drawback is the assumption of features are independent which may lead to misclassifications affecting the accuracy in making correct predictions. According to [6], this algorithm is often deployed in Intrusion detection at the network layer. The default parameters were used in this model.

5) Histogram Gradient Boosting (HGB)

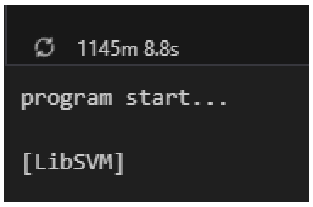
Histogram Gradient Boosting (HGB) is an ensemble ML algorithm just like the well-known AdaBoost, but it relies on histograms & integer-based data structures unlike traditional boosting methods which uses organized continuous values [9]. This allows HGB to execute faster than traditional gradient boosting methods. However, it can be computationally expensive to train on large datasets. Table 5 below represents the hyperparameters configured in the HGB classifier:

TABLE 5 Parameters configured for Histogram Gradient Boosting

Parameters	**Explanation/Justification**
learning_rate = 0.1	Refers to the parameter that manages the steps of the model in updating its predictions. A lower learning rate will slow down the model's predictions whereas a higher learning rate might cause the model to overfit.
max_depth = 5	Refers to the depth of the trees that can be built. A higher rate may cause the model to be susceptible to overfitting, but a lower rate may decrease the accuracy.

III. Analysis of Results

In this section, the performance and results acquired for each ML model used in this paper will be discussed and compared to derive the ML model with the best performance with the least computational time cost. This section is then followed by a comparison with the results of related works reviewed in this study. The algorithm: Support Vector Machine (SVM) was initially trained and tested based on the IoT-23 dataset. However, this paper moves on without the results of SVM due to the expensive computation cost it has with a time cost of 19.08 hours and no results were generated. The current hardware settings and environment could not operate beyond the time limit thus the result comparison for SVM will be carried out in future works where better hardware settings will be used. The figure below depicts the time cost when generating results for SVM.



```
1145m 8.8s

program start...

[LibSVM]
```

Fig. 5 Computation time for SVM

Below Tables 6-11 display the classification reports generated for the 5 ML models used in testing. An analysis of results will be given, followed by a comparison of results for the classification models used.

TABLE 6 Classification Report 1: Random Forest

Metric	Precision	Recall	F1-score	Support
Accuracy	-	-	0.73	1813987
Macro average	0.65	0.46	0.48	1813987
Weighted average	0.74	0.73	0.65	1813987

TABLE 7 Classification Report 2: Decision Trees

Metric	Precision	Recall	F1-score	Support
Accuracy	-	-	0.73	1813987
Macro average	0.65	0.48	0.49	1813987
Weighted average	0.74	0.73	0.65	1813987

TABLE 8 Classification Report 3: K-nearest neighbours

Metric	Precision	Recall	F1-score	Support
Accuracy	-	-	0.65	1813987
Macro average	0.32	0.30	0.22	1813987
Weighted average	0.57	0.65	0.53	1813987

TABLE 9 Classification Report 4: Naïve Bayes

Metric	Precision	Recall	F1-score	Support
Accuracy	-	-	0.65	1813987
Macro average	0.32	0.30	0.22	1813987
Weighted average	0.57	0.65	0.53	1813987

TABLE 10 Classification Report 5: Histogram Gradient Boosting

Metric	Precision	Recall	F1-score	Support
Accuracy	-	-	0.71	1813987
Macro average	0.23	0.20	0.20	1813987
Weighted average	0.65	0.71	0.64	1813987

TABLE 11 Results obtained for each model

ML Model	Accuracy	Time Cost (seconds)
Random Forest	0.7325	344.09
Decision Trees	0.7325	28.22
KNN	0.7323	27933.61
Naïve Bayes	0.6493	23.49
Histogram Gradient Boosting	0.7135	137.83

According to Table 11 shown above, the algorithm with the highest accuracy cannot be derived based on the results generated as the three algorithms: Random Forest (RF), Decision Trees (DT) and K-nearest neighbours (KNN) share the same accuracy of 0.73 with KNN's decimal notation slightly lower compared to the other two classification models. The presumption is the three models are overfitting which causes them having the same accuracy. Cross validation needs to be performed to discover and investigate if the models are overfitting. Due to the limitations of current hardware settings, it is not performed for all the five models as it is computationally expensive especially for large datasets.

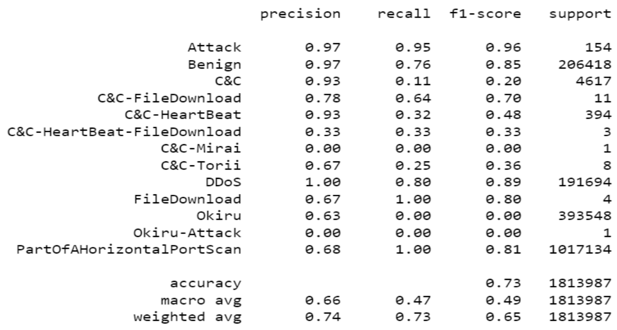
However, the three algorithms have distinct time costs where DT has the least time cost with 28.22 seconds. The algorithm with the highest computational time cost of all models is KNN with 27933.61 seconds which is equivalent to 7.7 hours. According to [10], KNN's computational complexity relies on the size of the dataset which in this case the dataset used in this project has 6046623 entries in total. Another factor that might affect the time cost is the value of *k*; where the higher the *k* value, the higher the computational time. A smaller *k* value would be more appropriate for a large dataset however it is susceptible to overfitting therefore the default *k* value 5 was used instead.

Random Forest (RF) has a time cost of 344.09 seconds which is approximately 5.7 minutes. The time cost for RF is higher than DT, however it is lower than KNN. HGB shows an accuracy of 0.71 which is the second highest in this study. HGB has a time cost of 137.83 seconds which is lower than RF and KNN. NB has the lowest accuracy compared to the other algorithms in this study with an accuracy of 0.65. Nevertheless, it has the least time cost of all the algorithms with a time cost of 23.49 seconds. As discussed in Section 3.4, NB assumes that the features are independent while the rest of the models used in this paper do not make this assumption. This factor could lead to NB being less accurate but faster to train. In the context of this analysis, DT presents the highest accuracy and has the lowest execution time among the other classification models tested in this study.

Apart from the accuracy metric to evaluate the performance of the classification models, the metrics

Precision, Recall, F1-score, and Support are significant as well in comprehending the trade-offs between mispredictions: false negatives and false positives. The accuracy metric can be misleading as it only displays the percentage of correctly predicted classes which disregards mispredictions which is essential to understand the effectiveness of the model. The dataset is considered to be imbalanced as the support score for some classes are notably lower than the score for other different classes. Figure 6 below depicts a full classification report for one of the ML models used in this analysis.

The above report indicates that the model is performing quite well towards the majority classes (PartOfAHorizontalScan, DDoS, Benign) with higher precision and recall scores. Nevertheless, the classification model is not performing too well on some of the minority classes (C&C-HeartBeat-FileDownload, C&C-Torii). Moreover, there are 0 values for F1-socre, precision, recall for the classes: C&C- Mirai & Okiru-Attack; 0 values are also present for recall and F1-score of the class: Okiru. This denotes that the dataset is possibly imbalanced.



```
                            precision    recall  f1-score   support

                    Attack       0.97      0.95      0.96       154
                    Benign       0.97      0.76      0.85    206418
                       C&C       0.93      0.11      0.20      4617
          C&C-FileDownload       0.78      0.64      0.70        11
             C&C-HeartBeat       0.93      0.32      0.48       394
C&C-HeartBeat-FileDownload       0.33      0.33      0.33         3
                 C&C-Mirai       0.00      0.00      0.00         1
                 C&C-Torii       0.67      0.25      0.36         8
                      DDoS       1.00      0.80      0.89    191694
              FileDownload       0.67      1.00      0.80         4
                     Okiru       0.63      0.00      0.00    393548
              Okiru-Attack       0.00      0.00      0.00         1
 PartOfAHorizontalPortScan       0.68      1.00      0.81   1017134

                  accuracy                           0.73   1813987
                 macro avg       0.66      0.47      0.49   1813987
              weighted avg       0.74      0.73      0.65   1813987
```

Fig. 6 Full Classification Report: Decision Trees

As the dataset is imbalanced, Macro Average is more preferred to evaluate the classification models' performances. The reason is that Macro Average treats all classes as uniformly important to evaluate the performance of the model. In contrast, Micro Average (accuracy) provides equal importance to each class where in the case of an imbalanced dataset, classes with higher observations will greatly affect the results [11]. According to multiple sources such as [11], [12], and [7], Macro Average scores are preferred for imbalanced datasets as it will indicate the model's true performance even though the classes are skewed. Macro average scores are the unbiased mean of the performance metric scores (Precision, Recall, F1-score) calculated for each class [12]. The Macro Average of these performance metrics will be compared and evaluated instead of the individual performance metric scores for each class.

The algorithm which has the highest Macro precision score is KNN with a macro score of 0.73 in which refers to the predictions that were correctly identified. The Macro Precision scores for RF and DT both have the same score of 0.65. NB has a Macro precision score of 0.32 whereas HGB has the least score of all algorithms with a score of 0.23. Moreover, DT has the highest Macro Recall score of 0.48 where this score refers to as the positive instances the classifier can identify. KNN shares a close Macro Recall score with RF with scores of 0.47 and 0.45 correspondingly. HGB has the least score as well for Macro recall score with a score of 0.20 whereas NB has a score of 0.30.

In the context of this analysis, the Macro F1-score will be more focused on to evaluate the performance of each classification model as it considers both the two metrics: Precision and Recall. The reason is that, both false positives and false negatives are critical when detecting network anomalies. For instance, regarding false positives, incorrectly identifying benign or authorized network activity as anomalous or malicious may result in unnecessary interruptions such as segregating the infected IoT device from the network. Moreover, pertinent to false negatives, unable to identify or predict network anomalies and malicious network activities when in fact is present may cause malicious attackers gaining unauthorized access to IoT devices such as compromising the devices to launch further attacks such as DoS/DDoS. Additionally, the Macro F1-score also provides a comprehensive measure of the classification model's performance than just the accuracy metric.

A higher Macro F1-score demonstrates that the classification model is performing well. Both DT and KNN have a Macro F1-score of 0.49 which is the highest among the other classification models in this study. RF has a score of 0.47 making it the second highest whereas HGB and NB have a similar score of 0.20 and 0.22 respectively, which is the lowest in this study. The factor that might cause a low score and poor performance of these two models is the imbalance of classes present in the dataset which causes it to be biased towards the majority class. Moreover, another factor is that the parameters of the models are not tuned properly.

In comparison with the accuracy score, both KNN and DT have the highest accuracy with a score of 0.73 and macro average F1 score of 0.49. This indicates that these two classification models performed better than the rest of the models in this study. However, this also implies that the two model's performance is poor in predicting on the minority classes as the difference between the two-performance metrics: accuracy and Macro F1-score are notably large. Further interpretation on the performances will be done by evaluating the confusion matrices of each classification model.

A summary table 12 of the count of correctly identified classes by each model are as presented below.

TABLE 12 Summary results of confusion matrix

	RF	***DT***	***KNN***	***NB***	***HGB***
Attack	150	146	145	6	0
Benign	156673	156675	156702	5994	150968
C&C	527	528	519	472	483
C&C-FileDownload	8	7	10	8	0
C&C-HeartBeat	128	128	131	135	0
C&C-HeartBeat-FileDownload	1	1	1	1	0
C&C-Mirai	0	0	0	0	0
C&C-Torii	2	2	1	0	0
DDoS	154110	154112	154211	154094	154084
FileDownload	3	4	3	2	0
Okiru	155	155	193	0	7
Okiru-Attack	0	0	0	0	0
PartOfA Horizontal	1017070	1017066	1016614	1017112	988884

PortScan					
Performance Ranking	*Third*	*Second*	*First*	*Fourth*	*Last*

Intriguingly, none of the ML models are able classify the C&C- Mirai and Okiru-Attack class. This could be due to the fact that there is only 1 occurrence of that class in the entire dataset. Similarly with the C&C-HeartBeat-FileDownload class, however, some models managed to classify and display some results. For the majority classes such as Benign, DDoS, Okiru, and PartOfAHorizontalPortScan class, the five ML models are able to classify most of them which indicates the models have problems in classifying the minority classes because of the proportion in the dataset. Moreover, there exists a large number of misclassifications for the Okiru and C&C class for all ML models used in this study. An assumption is that for the captures of these attacks might not have the features that can properly differentiate them from the others thus it categorizes it as the class that resembles it the most [13].

According to the comparisons and analysis made for Macro Average scores of the performance: Precision, Recall and F1 score, the comparison of the accuracy metrics and computational time costs of each classification model and the generic interpretation of confusion matrices, it is evident that the model with the most exceptional performance along with the least computational time cost is DT. Although KNN displayed better performance in classifying the classes accurately than the rest of the models but the computational cost of KNN is 27933.61 (approximately 7.7 hours). Conversely, DT displayed the 2nd best performance, and its performance is similar to of KNN but with a significantly lower computational time cost of 28.22 seconds.

This section compares the performance metrics obtained in this paper against related works using similar models on the same dataset (IoT-23). Two of the authors mentioned in this section [13] and [14] did not include computational time costs for each model in their study, which is an important metric in the context of IoT devices and there is no method in identifying how computationally fast their testing is as the devices usually have resource constraints requirements. The accuracy metric will be used for comparison against related works as majority of the papers reviewed in this study do not provide sufficient information about the Macro Average scores. HGB is not discussed in this section as the related works using the same dataset did not incorporate HGB in their work but instead incorporated a different ensemble learning method: AdaBoost. Thus, the remaining four models will be discussed instead.

TABLE 13 Result comparison with related works: [13], [14] and [15]

Model [paper]	**Accuracy**
RF [paper 13]	1.00
RF [paper 14]	0.96
RF [*this paper*]	**0.73**
NB [paper 13]	0.23
NB [paper 14]	0.63
NB [paper 15]	0.30
NB [*this paper*]	**0.65**
KNN [paper 14]	0.65
KNN [*this paper*]	**0.73**
DT [paper 14]	0.76
DT [paper 15]	0.73
DT [*this paper*]	**0.73**

Based on related works reviewed using the same dataset – IoT-23, Stoain et.al [13] obtained an accuracy of 1.00 for RF where the authors segregated the dataset into four parts of the same size and the accuracy was obtained from each segregated parts which might be the cause of obtaining a very high accuracy. Piragash et. al. [14] utilized the same ML models in their work on the same dataset as well where the accuracy score, they acquired for RF is 0.96. During the data pre-processing phase, the authors [14] mentioned that they omitted several file captures in which not all file captures were converted to a .csv which makes their dataset smaller with fewer rows. In comparison to this study, the accuracy score obtained for RF is 0.73 which is lower than the two papers. The reason for that could be because the pre-processed .csv dataset obtained from Kaggle has 6,046,623 records which is noticeably larger than the two papers [13] & [14].

Stoain et.al [13] obtained an accuracy score of 0.23 for NB which is the lowest compared to other related works and the results in this study. Yue et.al. [15] obtained a score of 0.30 whereas Piragash et. al. [14] acquired an accuracy score of 0.63 which is similar with this study. This study obtained an accuracy score of 0.65 for NB. A possible assumption is that NB is usually more suited for large datasets, and it is prone to overfitting with limited data where [13] opted for the lighter version of the .pcap files. According to the authors [13], the lighter version does not contain complete captures whereas the authors of [15] stated that their combined .csv file has a total of 1,444,674 entries. In contrast to this study where there are a total of 6,046,623 records in the dataset used.

Furthermore, Piragash et. al. [14] obtained an accuracy score of 0.65 where this study obtained a higher accuracy score of 0.73 for KNN. This could be because of the variation in data processing techniques or the selection of n_neighbours (K) which may affect the accuracy score. Piragash et. al. [14] did not provide a thorough description of methods involved in data processing or the value of parameters used when training the model. As for DT, Yue et.al. [15] acquired the same accuracy score as obtained in this study with a score of 0.73. In contrast, the authors of [14] achieved the highest accuracy score of 0.76 for DT. The possible reason that the accuracy score obtained in this study is lower than of [34] could be due to different data processing methods and hyperparameters tuning differences where no hyperparameters were tuned for Decision Trees in this study. Similarly, to the justification made above, Piragash et. al. [14] did not provide sufficient information about the parameters used in their analysis.

As compared to the papers mentioned in this section, this paper provides a more detailed discussion of results and a concise explanation and possible assumptions that affect the results of each ML model. From the comparative discussion above, although the accuracy score of NB in this study is higher, it is evident that NB has the lowest accuracy compared to the other models. On the other hand, RF displayed the highest accuracy although this study achieved a lower result than the two papers. Nevertheless, the combined dataset obtained by the authors are not as large compared to the one obtained in this study which affect the results obtained. To sum up, the comparison with paper [13], [14] and [15] demonstrates that the results obtained in this study is precise.

IV. Conclusion

To conclude, out of the five ML models evaluated, DT presented the best performance along with a computational time cost of 28.22 seconds whereas HGB displayed the worst performance in the context of detecting and classifying network anomalies upon the IoT-23 dataset. Although NB has the least time cost generated in this study (23.49 seconds), its performance is not as substantial as DT. Additionally, the imbalance dataset is also a factor that affects the accuracy and performance of each model, causing it to be bias to the majority classes. The dataset used in this paper (IoT-23) does not perfectly represent 5G IoT scenarios and all of its encountered attacks. Nevertheless, it still exhibits the effectiveness of the ML models in classifying anomalous network behavior to present a proof of concept for general anomaly detection that can be implemented in future works. The apparent limitation of this paper is the technical drawbacks of hardware settings. Cross validation and Hyperparameter tuning for all classification models were not performed due to limitations of current hardware settings, and it is computationally expensive especially for a large dataset like IoT-23. Moreover, the results for SVM could not be generated as the current hardware could not operate beyond the time limit. Thus, cross validation and hyperparameter tuning will be performed in future works with upgraded hardware to prevent overfitting and to improve the performance of the ML models. With upgraded hardware in future works, the results for SVM can be generated as well.

V. References

[1] C. Suraci, S. Pizzi, A. Molinaro, A. Iera and G. Araniti, "An RSA-based Algorithm for Secure D2D-aided Multicast Delivery of Multimedia Services," IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1-6, 2021. Available: https://doi.org/10.1109/BMSB49480.2020.9379851

[2] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Towards 5G Wireless Systems," IEEE Internet of Things Journal, vol. 7, no. 1, pp. 16-32, 2020. Available: https://doi.org/10.1109/JIOT.2019.2948888

[3] F. Hussain, R. Hussain, S. A. Hassan and E. Hossain, "Machine Learning in IoT Security: Current Solutions and Future Challenges," IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1686-1721, 2020. Available: https://doi.org/10.1109/COMST.2020.2986444

[4] A. P. &. M. J. E. Sebastian Garcia, "IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set]," Zenodo, 2020. Available: http://doi.org/10.5281/zenodo.4743746R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[5] L. McNulty and V. G. Vassilakis, "IoT Botnets: Characteristics, Exploits, Attack Capabilities, and Targets," 13th International Symposium on Communication Systems, Networks and Digital Signal Processing, pp. 350-355, 2022. Available: http://dx.doi.org/10.1109/CSNDSP54353.2022.9908039

[6] Tahsien, Syeda & Karimipour, Hadis & Spachos "Machine learning based solutions for security of Internet of Things (IoT): A survey," Journal of Network and Computer Applications, vol. 161, pp. 1084-8045, 2020. Available: http://dx.doi.org/10.1016/j.jnca.2020.102630

[7] P. Sharma, S.Jain, S.Gupta, V.Chamola "Role of Machine Learning and Deep Learning in Securing 5G-Driven Industrial IoT Applications," Ad Hoc Networks, vol. Volume 123, p. 102685, 2021. Available: https://doi.org/10.1016/j.adhoc.2021.102685

[8] A. Bonnet, "Fine-tuning Models: Hyperparameter Optimization," Encord, 22 August 2023. [Online]. Available: https://encord.com/blog/fine-tuning-models-hyperparameter-optimization/. [Accessed 10 September 2023].

[9] A. Pius, "A Faster Ensemble Model Method in Sklearn: Histogram-Based Gradient Boosting," Medium, 22 September 2020. [Online]. Available: https://medium.com/chat-gpt-now-writes-all-my-articles/a-faster-ensemble-model-method-in-sklearn-histogram-based-gradient-boosting-7033ff170bc0. [Accessed 3 October 2023].

[10] T. LaViale, "Deep Dive on KNN: Understanding and Implementing the K-Nearest Neighbors Algorithm," Arize, 16 March 2023. [Online]. Available: https://arize.com/blog-course/knn-algorithm-k-nearest-neighbor/. [Accessed 3 October 2023].

[11] A. M. Sefidian, "Understanding Micro, Macro, and Weighted Averages for Scikit-Learn metrics in multi-class classification with example," Sefidian Academy, [Online]. Available: https://iamirmasoud.com/2022/06/19/understanding-micro-macro-and-weighted-averages-for-scikit-learn-metrics-in-multi-class-classification-with-example/. [Accessed 4 October 2023].

[12] S. Allwright, "Micro vs Macro F1 score, what’s the difference?" 20 July 2022. [Online]. Available: https://stephenallwright.com/micro-vs-macro-f1-score/. [Accessed 4 October 2023].

[13] N. A. Stoain, "Machine Learning for anomaly detection in IoT networks: Malware analysis on the IoT-23 data set," 2020. Available: https://www.semanticscholar.org/paper/Machine-Learning-for-Anomaly-Detection-in-IoT-oStoian/a1dcd833360298a806c67e5622da56f6cfd1cfde

[14] Piragash Maran, Timothy Tzen Vun Yap, Ji Jian Chin, Hu Ng, Vik Tor Goh, Thiam Yong Kuek, "Comparison of Machine Learning Models for IoT Malware Classification," Atlantis Press, pp. 15-28, 2022. Available: https://doi.org/10.2991/978-94-6463-094-7_3

[15] N. V. Yue Liang, "Machine Learning and Deep Learning Methods for Better Anomaly Detection in IoT-23 Dataset Cybersecurity," 2021.

[16] A. Ghasempour, “Internet of Things in Smart Grid: Architecture, Applications, Services, Key Technologies, and Challenges,” Inventions Journal, vol. 4, no. 1, pp. 1-12, 2019, https://doi.org/10.3390/inventions4010022.

[17] “What is the IoT?,” https://www.ibm.com/topics/internet-of-things.