An approach for malware detection on IoT systems using machine learning

MSc Research Project
Cyber Security

Jonatas Alves Fagundes
Student ID: x20144946

School of Computing
National College of Ireland

Supervisor: Niall Heffernan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Jonatas Alves Fagundes		
Student ID:	x20144946		
Programme:	MSc Cyber Security	**Year:**	2020/2021
Module:	Research Project		
Supervisor:	Niall Heffernan		
Submission Due Date:	23/08/2021		
Project Title:	An approach for malware detection on IoT systems using machine learning		
Word Count:	7075	**Page Count**	22

Signature: Jonatas Alves Fagundes

Date: 23/08/2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

An approach for malware detection on IoT systems using machine learning

Jonatas Alves Fagundes
x20144946

Abstract
Internet of things (IoT) devices communicate, collect and exchange data about our online activities and preferences via the internet. These smart devices are infiltrated into our lives in houses, workplaces, cities and it generates millions of gigabytes data every day. However, any device that shares data through the internet is vulnerable to cyberattacks. It is known that most IoT devices are built containing poor security which makes them attractive targets to cybercriminals. Malware detection has been a research area that is constantly studied due to the evolution of malwares and its vectors. Machine learning techniques have been presented as one of the most efficient and effective solutions to identify different vectors of attacks in IoT devices. This research aims to present a study of the implementation of machine learning techniques in the detection of malware to address vulnerabilities in IoT environments. This work explores ways to identify normal and anomalous behaviours in IoT systems using machine learning algorithms as classifiers that include Random Forest, Artificial Neural Network - Multilayer Perceptron, K-nearest Neighbors, and Support Vector Machine. In this work, we could identify that Random Forest presents better results identifying malicious behaviour based on tests, previous work, and other sources.

1 Introduction

With the popularization of the Internet of Things (IoT) - technology on the rise that is used in several areas such as agriculture, health, smart homes, among other applications - some concerns about this theme have been standing out. Allied to innovation, this recent technology is followed by many challenges which the main one is without a doubt, security. For the success of the IoT concept, it is essential to guarantee the security and privacy of user data. Intrusion detection is one of the techniques used to increase security, aiming to detect and identify intrusion attempts in computing environments by internal and external invaders. Intrusion Detection Systems (IDS) differ in several aspects, such as the type of detection performed, the system architecture according to the target of the detections, the location where it is arranged, among other characteristics.
Malware detection is a wide area of automated monitoring mechanisms to identify and protect devices and systems from all forms of malicious code. Considering that antivirus and other traditional mechanisms have been losing their efficiency in protecting from malwares, other security mechanisms must be used for combating these threats. IoT malware detection has been an important and urgent subject in researches by the security community as the attacks on IoT environments have skyrocketed in recent years.
Machine Learning (ML) is part of artificial intelligence (AI) and consists of programming a computer to learn from example data or past events to create improved criteria resulting in better decision-making without human intervention. Machine Learning algorithms search for patterns in data and try to make assumptions and once the algorithm starts being accurate, it

implements that knowledge to new sets of data improving accuracy and performance in that algorithm. There are two main methods of machine learning that are commonly used: Supervised and Unsupervised learning. Supervised learning trains algorithms using examples that are labelled, in other words, an input that the wanted output is known. The learning algorithm receives the correct output and a set of inputs where it learns by comparing them to find errors. After comparing, it collects that information and modifies the algorithm accordingly. Supervised learning utilises patterns to foresee the value of a label on unlabelled data by using methods such as classification, prediction, regression, and gradient boosting. Unsupervised learning, on the contrary, is used when there are no historical labels or correct outputs. The learning algorithm must identify the data being inputted and find its structure. This algorithm is commonly used to segment text topics, recommend items, and identify data anomalies by using methods such as nearest-neighbor mapping, k-means clustering, self-organizing maps, and singular value decomposition.

The detection of malware in IoT environments is a great challenge due to the massive amount of data processed. A relatively recent example is the malware botnet Mirai[1] which in 2016 compromised thousands of IoT devices and performed DDoS attacks whose traffic hit 1Tbps. With the increase in the amount of data generated by IoT devices, it became necessary to implement other technologies to assist in data processing. The heterogeneous nature of IoT systems and the complex threat landscape have been a challenge for organisations to provide adequate security. According to the most recent SonicWall threat report 2021[2], in the previous year, the number of IoT malware attacks reached 56.9 million. In addition, according to Bitglass in its Remote Workforce Security Report (2020)[3], 65% of organizations allow access from personal, unmanaged devices' which is an opened door for malicious attackers to exploit.

To be effective, it is necessary to have a security mechanism that is able to identify which characteristics are relevant to distinguish malicious from benign codes where we can use techniques such as Static analysis and Dynamic analysis. Machine learning has appeared as one of the best security mechanisms for protecting IoT environments from those powerful and destructive malware attacks. Having access to activity and behaviour data of the IoT devices connected to the network makes it possible to train Machine Learning (ML) classifiers that can be used to identify malicious behaviour. Having the ability to identify devices connected to a network also provides better control over network traffic and allows better management of devices considered to be a security risk to the network structure as a whole. Considering the increase in the number of IoT devices and its lack of built-in security features, we can notice that there is an urge for a solution that will effectively protect it against cybercriminals. Given this context, we can assume that this is an urgent matter for cybersecurity which was the motivation for this work. This work aims to present an approach for malware detection in IoT devices using machine learning techniques and analysis of device fingerprinting for the identification of malicious behaviours. After choosing the most relevant characteristics and the database with samples of the network traffic, the training of

[1] https://www.csoonline.com/article/3258748
[2] https://www.sonicwall.com/2021-cyber-threat-report/
[3] https://www.bitglass.com/

the machine learning (ML) classifiers is carried out for each device using the Random Forest, Multilayer Perceptron, K-Nearest Neighbors, and Support Vector Machine algorithms. Upon completion of the tests, the machine learning classifiers are ranked regarding their suitability in classifying between common data from IoT devices and anomalous data from devices infected by malicious agents.

1.1 Research Question

- How to improve security on IoT systems based on malware detection by using machine learning?

Considering that IoT devices have different systems and peculiarities, and that the number of malware attacks has been increasing recently, an approach on how to improve the security on IoT systems that encompasses the differences between them and curb or possibly stop those attacks is very important and needed by the time this work is being presented. By using the Random Forest algorithm and analysing the map of network behaviours in IoT systems, this work aims to create a malware detection approach to answer the research question proposed.

1.2 Objective

The objective of this work is to identify IoT devices and differentiate normal and anomalous behaviours analysing their operating patterns through network traffic samples. For this purpose, it is studied which network traffic characteristics of the devices should be chosen to make the identification of behaviours as accurate as possible. The key points we try to achieve in this work are:
• Implement Machine Learning algorithms with a high level of accuracy in the data classifications of IoT devices;
• Discover which algorithms and network traffic characteristics are best suited for classifying the behaviour of IoT devices;
• Propose an approach for malware detection on IoT systems.

1.3 Structure

The rest of this document is designed in sections as follows. In section 2, we present previous work where different approaches based on malware detection and machine learning approaches were studied. In section 3, we illustrate our research methods and specification of our approach classifying data, training, and testing machine learning classifiers. Followed by section 5, where we evaluate the classifiers. In section 6, we implement the identification of behaviours. We then present a conclusion and future work in section 7.

2 Related Work

According to Zhou et al., IoT devices are vulnerable to attacks because of their interdependence where the authors claim the attacker could exploit other devices` behaviour that belongs to the same environment and have an interdependence relationship with the target device. It would increase the chances of bypassing the security mechanism applied to the target device. In the same work, the authors mention the diversity of IoT devices as a threat due to many different firmware, hardware, protocols, and lack of security which in most cases are common web security vulnerabilities used by attackers. It highlights the importance of monitoring malicious behaviours and protecting the environment as a whole.

2.1 IoT Malware Detection

Aslan et al. review different approaches of malware detection including signature-based detection and behaviour-based detection. The authors compare different solutions, give an evaluation of each technique used in previous work presenting pros and cons, and conclude by reinforcing that there is still a need for malware detection research. In contrary to this paper, none of the approaches presented by the authors used traffic network parameters in the detection of malicious behaviour.

In the work of Wang et al., the authors presented a study of IoT malware where features of malware samples were extracted to identify their relationship with other malware family and by using honeypots the authors collected information to design a classifier to categorise IoT malwares into families and identify the correlation between them. This work helps the research community to have a better understanding of IoT malwares, however, it does not provide a solution to protect against them.

Ni An et al. implemented semi-supervised algorithms for the detection of malwares in home routers where the authors affirm to have achieved high rates of detection of malware without false alarms. Z. D. Patel, Westyarian et al., and Hadiprakoso et al. applied dynamic-based and hybrid-based malware analysis to present a solution of malware detection in Android operation system using a machine learning model where a detection accuracy of 99% was achieved. However, the authors limited their work to a single type of device. As mentioned before, the IoT environment consists of multiple different devices with different characteristics such as hardware, firmware, and protocol.

2.2 Device Fingerprinting and Machine Learning

In Miettinen et al., a system called IoT Sentinel which is aimed at small networks such as homes and small offices is proposed. The system can identify the type of IoT device introduced into the network and impose measures to mitigate possible security flaws in devices considered to be vulnerable. IoT Sentinel controls the network traffic accessed by devices recognized as vulnerable to prevent problems to other devices connected to the network. J. Pan created a novel model using convolutional supervised machine learning based on network behavioural fingerprint to identify IoT devices efficiently. In another similar work, Bezawada et al. presented a methodology that uses extracted features from network traffic by using device fingerprinting. A machine learning model is trained to identify similar types of IoT devices where an accuracy of 99% was achieved in the identification. In Aneja et al., the objective was to identify IoT devices based on only one attribute, Inter Arrival Time (IAT), which is the time interval between two consecutive received packets. For this, it was generated graphic images from the IAT of the registered packets which served as input to a Convolutional Neural Network (CNN). In the end, the hit

rate achieved was 86.7%. In another work that uses CNN, M. Yeo et al. presented a method of malware detection in real-time using a large amount of data. The method applied for classification achieved 85% of accuracy and precision. In contrary to this paper, the authors utilised different parameters to analyse and classify data and behaviours which impacted on their results.
In the article of Shahid et al., techniques based on Machine Learning for monitoring IoT networks are presented where the authors analyse common and malicious IoT network traffic and use characteristics such as the size of the first N packets sent, the size of the first N packets received, the inter-arrival times of N-1 packets between the first N packets sent, and the inter-arrival times of N-1 packets among the first N packets received. Regarding the classifiers, the algorithms tested were Decision Tree, Random Forest, Naive Bayes, K-Nearest Neighbor, Support Vector Machine, and Artificial Neural Network, in which Random Forest presented the best results. Even though these papers presented great results, they are only focused on the identification of IoT devices to be used for other implemented solutions.
In Shaikh et al. a model for classifying network characteristics from malicious devices is presented to identify attacks that originate from compromised IoT devices using packet header information from a darknet database. The machine learning classifiers explored were Random Forest, Gradient Boost, AdaBoost, and Naive Bayes in which Gradient Boost and Random Forest obtained better results.
D. Gibert et al., P. Priyadarshan et al.; and M. Wazid provides detailed work about approaches using machine learning for the detection of malware. The authors compare and analyse a great number of previous works in different categories according to multiple different characteristics including dynamic and hybrid-based analysis, database, and objectives. In addition, the authors evaluate the performance of methods used in those papers based on common solutions used by the scientific community.
In a different approach, S. A. Roseline et al. uses a method of malware analysis by visualization converting malware samples into greyscale images. This approach combines datasets with benign and malicious samples and trains multi-layered classifiers with labelled images to try to predict malware in the data. The algorithm used was Random Forest and achieved an accuracy of 98.9% and more effectiveness than other deep learning models.
M. Mehra et al. proposed a model that combines system files, IoT application files, and Linux malware to build a Random Forest classifier. The authors provide an exhaustive analysis of machine learning models trained using parameters that were observed to present good results in the detection of zero-day attacks.
The importance of choosing the right attributes that will constitute the device fingerprinting is because of the fact that the database used for the training of machine learning classifiers is built based on these attributes.
In table 1.1 below, we present the algorithms used respectively by each author previously mentioned.

Algorithm	Author
Decision Tree	Shahid et al.; Bezawada et al.
Random Forest	Shaikh et al., Shahid et al.; Miettinen et al.; M. Yeo et al.; S. A. Roseline et al.
Gradient Boost	Shaikh et al.; Bezawada et al.
Ada Boost	Shaikh et al.
Naïve Bayes	Shaikh et al., Shahid et al.
LSTM-CNN	Wang et al.; J. Pan
Convolutional Neural Network	Aneja et al.; M. Yeo et al.
Support Vector Machine	Shahid et al.; M. Yeo et al.; M. Mehra et al.
K-Nearest Neighbor	Shahid et al.; Bezawada et al.; M. Mehra et al.
Artificial Neural Network	Shahid et al.
Majority Voting	Bezawada et al.

Table 1.1: Algorithms used in related work

In this paper, the focus is not only on one aspect of classification but encompasses both the identification of data between different types of devices and the differentiation between standard and malicious IoT network traffic.

3 Research Methodology

This section presents the methodology used to perform the classification of IoT devices. In section 3.1 the treatment of databases used in machine learning algorithms is presented, in section 3.2 the creation of training and testing databases is explained, and in section 3.3 the adjustment of attributes to be used in the classifiers is presented.

3.1 Data Processing

To get data from IoT devices, the ideal scenario would be to set up an environment with several devices and capture the traffic generated on the network. As this is not feasible for this work, the option chosen was to use a previously built database. Due to the area of IoT studies is something that there is still a lack of related materials, the number of public databases available for use is still scarce. The database that is available online[4] for research purposes was chosen for this work and also used in Miettinen et al.. This database represents the traffic emitted during the configuration of 21 IoT devices in a smart home and is divided into '.pcap' files according to the type of device. From these files, a total of ten characteristics were extracted to be used for the device fingerprinting, which can be seen in table 4.1.
Some of these features were chosen based on the related work in the previous chapter and others were chosen because of their relevance in the context of this work, for example, the TCP Flags attribute can help to indicate whether a packet is part of a SYN Flood type of

[4] The database is available on: https://research.unsw.edu.au/projects/bot-iot-dataset

DDoS attack. In addition, only primitive attributes were chosen as this paper deals with the classification of individual packages rather than the flow of packages.

Attributes	Description
Src Port	Source port
Dst Port	Destination port
Time	Packet arrival time
Length	Size of packet
Protocol	Protocol used
Ethernet Type	Auth, Arp, 802.1X, ipv4, …
Inter-Arrival Time	Time between a packet and the previous one
TTL	Time to live
TCP Flags	SYN, ACK, FIN, …
IP Protocol	UDP, TCP, …

Table 1.2: Fingerprinting attributes

After choosing the characteristics, the data extraction is performed transforming '.pcap' files into '.csv' to be used in machine learning classifiers.

3.2 Classification Databases

To create the training and testing databases for each classifier, we adopted the following procedures. Firstly, '.csv' files containing network traffic data of each device were concatenated into a single file. Then, for each type of existing device, a separate database was created containing all data from the network packets of this specific device and added X random packets data from the other 20 devices where X is the total number of packets of the device to be classified. Thus, each database is formed with 50% of data from the device to be classified and 50% with data samples from other devices.

3.3 Attributes Adjustment

After creating the classifiers databases, the next step was to adapt the attributes to be used in classifiers. Most of the data had categorical values in the original database, for example, "Protocol" = "EAPOL" or "Ethernet Type" = "IPv4", so it was necessary to convert these attributes to numerical values which are accepted by machine learning algorithms. In addition, the Ordinal Encoding method was also applied, where for each distinct value of a characteristic a new column was created, thus, creating several new columns and increasing the size of the database according to the variety of features. These columns have numeric values whi. An example of Ordinal Encoding can be seen in Figure 1. This method was applied only to those attributes that had categorical values, with the numerical attributes remaining the same and its use was due to the improvement in the results presented by the classifiers with its application.

Original Encoding	Ordinal Encoding
Poor	1
Good	2
Very Good	3
Excellent	4

Figure 1: Ordinal Encoding

In addition, the transformation of the attribute values of "Src Port" and "DstPort" were executed. As the variation of the port number and its values is considerably large, the transformation presented in Miettinen et al. was performed in order to divide the ports into four distinct network classes. The transformation is shown in table 1.3.

Port class	Port number	Assigned value
No port	n/a	0
Known port	0 - 1023	1
Registered port	1024 - 49151	2
Dynamic port	49152 - 65535	3

Table 1.3: Port transformation

4 Design Specification

The programming language used for this work is Python with sklearn library. After processing the databases, the next step is to train and test the classifiers. For the training and testing stage, the Stratified K-Folds cross-validation method was used. The database is divided K times among testing and training databases, in K separate parts of the same size which in each of the K divisions, K-1 databases are used for training and 1 database for testing. Figure 2 illustrates the operation of the base divisions for K = 5. This method has a good distribution among the existing data classes in the database and also increases the reliability of the results obtained by the classifiers. Test configurations were performed with 4, 6, and 10 folds where the value K = 10 (Stratified 10-Folds cross-validation) presented the best results, thus being the definitive value chosen for database division. After running the tests 10 times with each classifier algorithm for each device, the accuracy, precision, and recall results obtained per device were averaged.

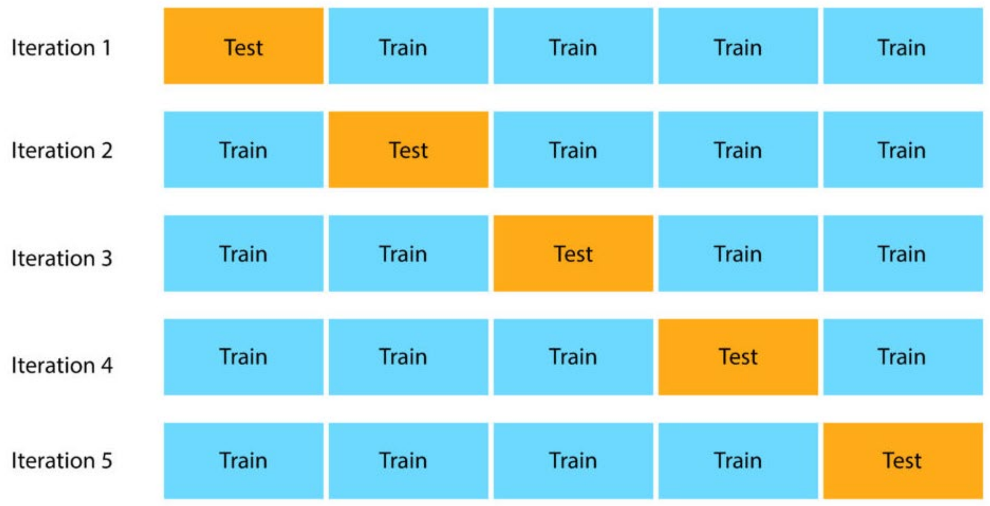


Figure 2: Stratified K-Folds Cross-Validation   Source: Great Learning[5]

A total of 4 algorithms were chosen to be classifiers, namely: Random Forest, Multilayer Perceptron, K-nearest Neighbor, and Support Vector Machine. The choice was made due to their performance in related works. All of the algorithms were implemented and tested using Python with sklearn library. Below in table 1.4, the individual settings for each algorithm are shown.

Classifiers	Configuration used in classifiers	
Random Forest	**Function:**	**Parameters:**

	from sklearn.ensemble import RandomForestClassifier	n_estimators = 40
		criterion = "entropy"

Multilayer Perceptron	**Function:**	**Parameters:**

	from sklearn.neural_network import MLPClassifier	max_iter = 1000
		tol = 0.000010
		solver = "adam"
		hidden_layer_sizes = (100)
		activation = "relu"
		batch_size = 200
		learning_rate_init =

K-nearest Neighbors	**Function:**	**Parameters:**

	from sklearn.neighbors import KNeighborsClassifier	n_neighbors = 5
		metric = "minkowski"
		p = 2

Support Vector Machine	**Function:**	**Parameters:**

	from sklearn.svm import SVC	kernel = "rbf"
		C = 2.0

Table 1.4: Configuration of classifiers in Python

[5] https://www.mygreatlearning.com/blog/cross-validation/

These parameters were established after several tests in order to achieve the best results.

5 Classifiers Evaluation

The metrics used to assess the efficiency of the algorithms are Accuracy, Precision, and Recall which definitions could be interpreted as follows:

- Accuracy: is the proportion of correctly labelled classes to the entire set of classes.
- Precision: out of all the positive class classifications that the model has made, how many are correct.
- Recall: among all positive class situations as expected value, how many are correct.

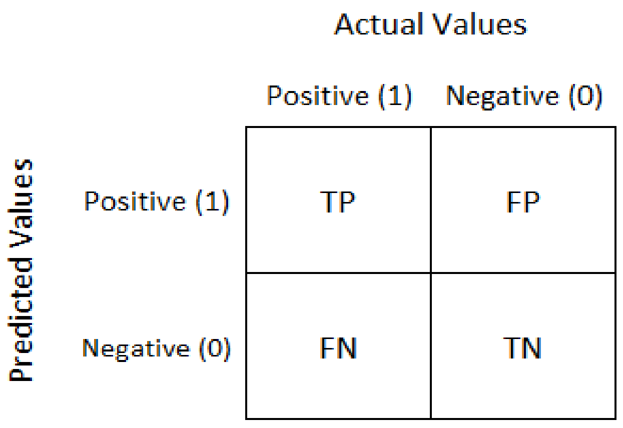
These metrics were chosen because of their common use in validating the efficiency of classifiers in machine learning. Below we present the formulas for each metric.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Figure 3: Formulas of most common metrics in machine learning

To calculate these metrics, the values of the confusion matrix generated after the classification of the databases are needed. These values are: True Negative (TN), False Negative (FN), True Positive (TP), and False Positive (FP). Taking as an example the classifier of one of the devices used in this work, Aria smart scale, TN represents the amount of data that is not Aria type and was classified as not being Aria type, FN represents the amount of data that is Aria type and have been classified as not being Aria type, TP represents the data that is Aria type and have been classified as being Aria type, and FP represents the data that is not Aria type and have been classified as being Aria type. The confusion matrix model is shown in figure 4.



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 4: Confusion matrix Source: Great Learning[6]

[6] https://www.mygreatlearning.com/blog/confusion-matrix-an-overview-with-python-and-r/

5.1 Device Classification Results

The results obtained in the classification and identification of devices are shown in tables 1.5, 1.6, and 1.7 which present the accuracy, precision, and recall values specifically achieved by each of the classifiers. Figures 5, 6, and 7 are the graphical representations of the results.
Table 1.5 presents the accuracy values for each classifier. When observing the results we verified that the Random Forest algorithm obtained an average accuracy of 87.2% among all devices, being the algorithm that obtained the best results in this metric. The algorithms MLP, SVM, and KNN obtained a mean accuracy of 67.5%, 53.2%, and 73.9% respectively.
Table 1.6 presents the precision results and table 1.7 shows the recall results obtained by the classifiers. In both metrics the Random Forest classifier showed the best results, as well as in accuracy, with averages of 87.1% and 83.9% of precision and recall, respectively. The MLP, SVM, and KNN classifiers obtained an average precision of 74.7%, 68.6%, and 73.9% and recall averages of 75.3%, 69.8%, and 77.1% respectively.

	Accuracy			
Device	**Random Forest**	**KNN**	**SVM**	**ML Perceptron**
Aria	1.0000	1.0000	0.3077	0.9231
D-LinkCam	0.7368	0.7368	0.2632	0.3158
D-LinkDoorSensor	0.7500	0.5833	0.2222	0.3889
MAXGateway	1.0000	1.0000	1.0000	1.0000
D-LinkHomeHub	0.8983	0.8220	0.7542	0.7458
WeMoLink	0.9114	0.7848	0.5696	0.6582
D-LinkSiren	0.8119	0.7228	0.5644	0.6733
WeMoSwitch	0.8302	0.7358	0.1132	0.5094
D-LinkWaterSensor	0.8554	0.7349	0.4217	0.5904
EdimaxPlug1101W	0.6952	0.2500	0.1250	0.2500
EdimaxPlug2101W	0.6857	0.1429	0.0000	0.1429
EdnetGateway	1.0000	1.0000	1.0000	1.0000
HomeMaticPlug	1.0000	1.0000	1.0000	1.0000
HueBridge	0.8412	0.6228	0.4211	0.4649
HueSwitch	0.8635	0.8189	0.9479	0.8660
D-LinkSensor	0.8901	0.8022	0.5165	0.7253
TP-LinkPlugHS110	1.0000	0.8333	0.5479	1.0000
Withings	1.0000	0.9286	0.9286	1.0000
WeMoInsightSwitch	0.8842	0.8000	0.6526	0.6842
D-LinkSwitch	0.8727	0.8000	0.4182	0.6455
TP-LinkPlugHS100	0.8000	0.4000	0.4000	0.6000
Average	0.8727	0.7390	0.5321	0.6754

Table 1.5: Accuracy results

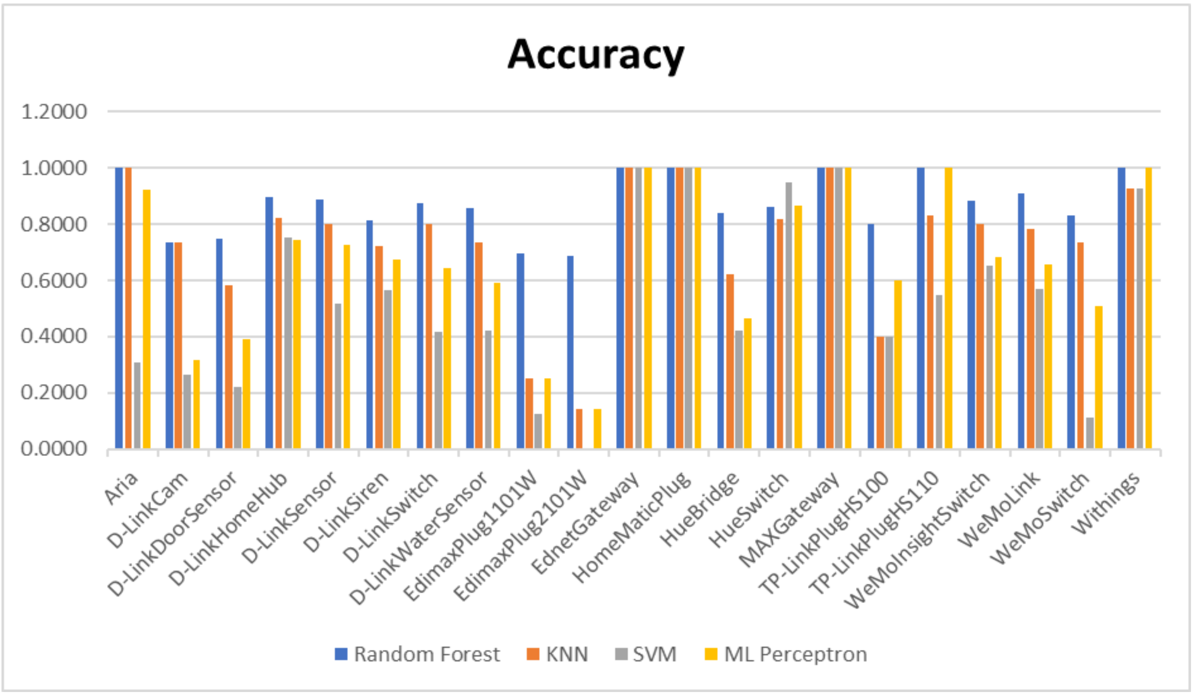


Figure 5: Accuracy chart

	Precision			
Device	**Random Forest**	**KNN**	**SVM**	**ML Perceptron**
Aria	1.0000	0.8125	0.6667	0.7500
WeMoInsightSwitch	0.8750	0.7835	0.5586	0.6633
D-LinkHomeHub	0.9217	0.7578	0.6692	0.6929
D-LinkSensor	0.8438	0.7374	0.7966	0.6286
D-LinkSiren	0.8542	0.8111	0.6129	0.6182
D-LinkSwitch	0.8889	0.8073	0.8537	0.7245
D-LinkWaterSensor	0.7978	0.7176	0.8537	0.6533
EdimaxPlug1101W	0.7874	0.3333	1.0000	0.6667
EdimaxPlug2101W	0.6687	0.2000	0.0000	0.5000
EdnetGateway	1.0000	1.0000	1.0000	1.0000
TP-LinkPlugHS110	0.7800	0.4545	0.5520	0.6000
Withings	1.0000	1.0000	0.8667	1.0000
HueSwitch	0.8447	0.7746	0.5736	0.6925
MAXGateway	1.0000	1.0000	0.7778	1.0000
TP-LinkPlugHS100	1.0000	0.8000	0.4000	1.0000
D-LinkDoorSensor	0.8182	0.6563	0.8000	0.7778
HueBridge	0.7788	0.7030	0.5963	0.7260
WeMoLink	0.8675	0.8052	0.4286	0.6190
D-LinkCam	0.7364	0.5600	0.5556	0.8571
WeMoSwitch	0.8462	0.8125	0.8571	0.5294
HomeMaticPlug	1.0000	1.0000	1.0000	1.0000
Average	0.8719	0.7394	0.6866	0.7476

Table 1.6: Precision results

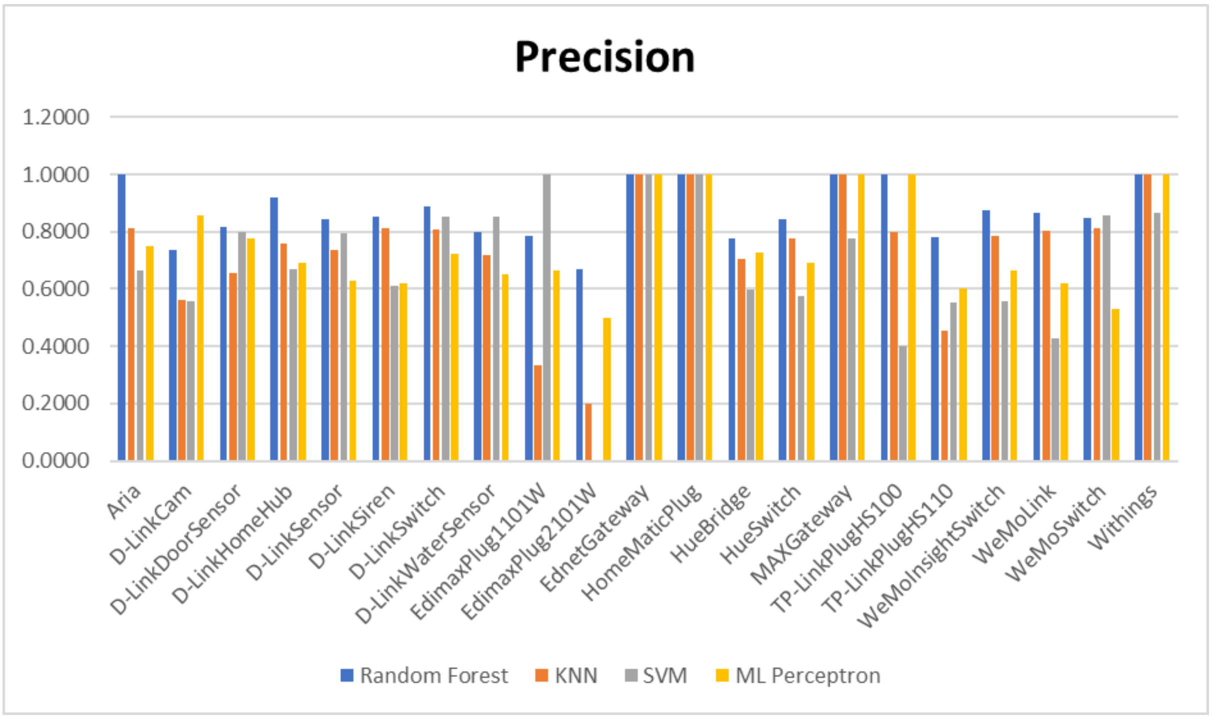


Figure 6: Precision chart

	Recall			
Device	**Random Forest**	**KNN**	**SupportVM**	**ML Perceptron**
Aria	1.0000	1.0000	0.6077	0.9231
D-LinkCam	0.7368	0.7368	0.5632	0.6158
D-LinkDoorSensor	0.7500	0.5833	0.6222	0.6989
TP-LinkPlugHS110	1.0000	0.8333	0.6781	1.0000
D-LinkSensor	0.8901	0.8022	0.5165	0.7253
WeMoLink	0.9114	0.7848	0.5696	0.6582
D-LinkSwitch	0.8727	0.8000	0.5182	0.6455
Withings	1.0000	0.9286	0.9286	1.0000
EdimaxPlug1101W	0.5000	0.5250	0.6250	0.5500
EdimaxPlug2101W	0.2857	0.5429	0.6687	0.5643
EdnetGateway	1.0000	1.0000	1.0000	1.0000
HomeMaticPlug	1.0000	1.0000	1.0000	1.0000
HueBridge	0.7412	0.6228	0.6211	0.7649
HueSwitch	0.8635	0.8189	0.9479	0.8660
MAXGateway	1.0000	1.0000	1.0000	1.0000
D-LinkHomeHub	0.8983	0.8220	0.7542	0.7458
TP-LinkPlugHS100	0.8000	0.4000	0.5940	0.6000
D-LinkSiren	0.8119	0.7228	0.5644	0.6733
WeMoInsightSwitch	0.8842	0.8000	0.6526	0.6842
D-LinkWaterSensor	0.8554	0.7349	0.6217	0.5904
WeMoSwitch	0.8302	0.7358	0.6132	0.5094
Average	0.8396	0.7712	0.6984	0.7531

Table 1.7: Recall results

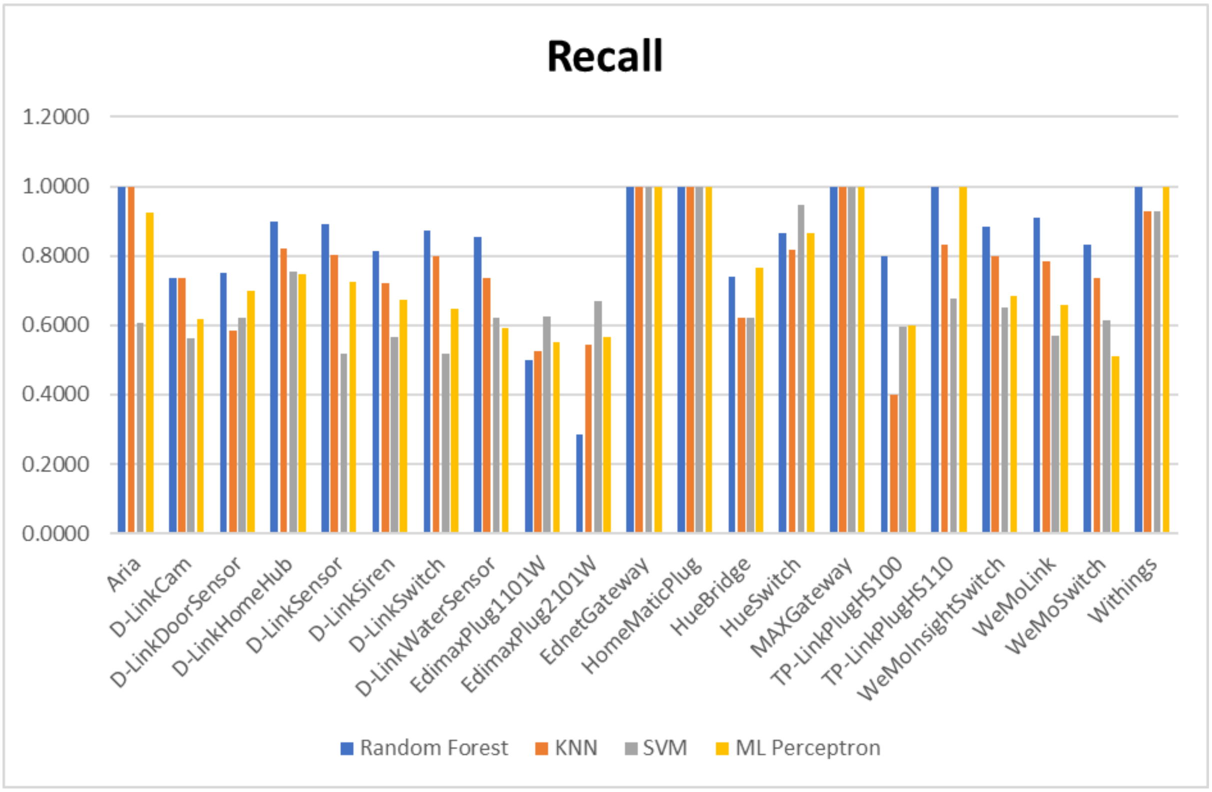


Figure 7: Recall chart

According to the results, we can observe that Random Forest had the classifiers with the best metrics for the vast majority of devices. The results obtained may have been greatly influenced by the size of the database, perhaps with larger databases and more training data the Multilayer Perceptron algorithm could have a better performance but for this specific case Random Forest stands out from the other algorithms.

5.2 Statistical Tests

To determine whether there is a statistically significant difference between the averages of the results, the Friedman and Nemenyi test was performed on the accuracy results of the classifiers. The accuracy metric was chosen in this case because it represents the total value of correct answers for all data in the database, thus, being the most important metric in this study. Firstly, the ranking of the accuracy results of the classifiers was carried out as shown in table 1.8. The ranking was performed by device getting a ranking by line. In the EdnetGateway, HomeMaticPlug, MAXGateway, TP-LinkPlugHS100, and Withings device lines the classifiers were tied so the results are not an integer.

Device	Random Forest	Support VM	KNN	ML Perceptron
WeMoLink	1	4	2	3
D-LinkCam	1	4	3	2
D-LinkDoorSensor	1	2	4	3
Withings	1.5	2	1.5	1.5
D-LinkSensor	1	2	3	4
D-LinkSiren	1	4	2	3
D-LinkSwitch	1	2	3	4
D-LinkWaterSensor	2	1	3	4
EdimaxPlug1101W	1	4	3	2
EdimaxPlug2101W	1	4	3	2
EdnetGateway	1.5	1.5	1.5	1.5
HomeMaticPlug	1.5	1.5	1.5	1.5
HueBridge	1	4	3	2
HueSwitch	1	4	2	3
MAXGateway	1.5	2	1.5	1.5
TP-LinkPlugHS100	1.5	3	2	1.5
TP-LinkPlugHS110	1	3	1	2
Aria	1	4	2	3
WeMoInsightSwitch	1	4	2	3
D-LinkHomeHub	1	4	3	2
WeMoSwitch	2	1	3	4

Table 1.8: Ranking accuracy

The values in table 1.8 were used as input to a function that performs the Friedman and Nemenyi test. The test was executed in R programming language with the nemenyi function from the tsutils library. Then, we executed the function tsutils with parameters (M, conf. level=0.95, plottype= "vline") where M is the matrix formed with the data from table 1.8. The result is presented in figure 8.

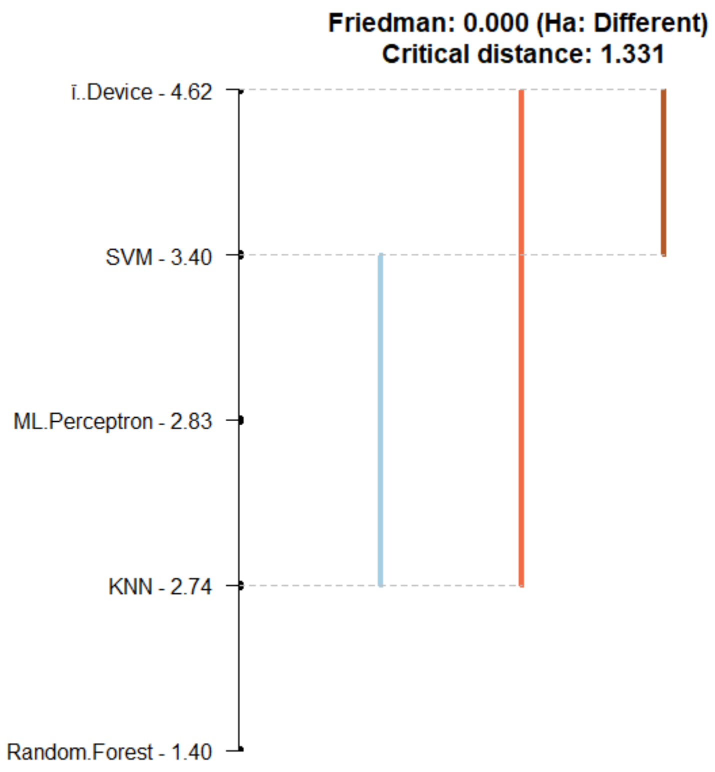


Figure 8: Friedman and Nemenyi test result

The critical distance (CD - Critical Distance) shown in figure 8 is 1.331, which means the distances between the results of the algorithms are statistically significant. The blue bar in figure 8 indicates that there is significant difference from KNN to SVM, thus, there is difference between using these two algorithms in this case. But the difference between Random Forest and Multilayer Perceptron (3.40 - 1.40 = 2.00) is greater than CD (1.33 < 2.00), in other words, statistically in this case it is better to use Random Forest rather than Multilayer Perceptron or any of the two other algorithms. Because it has the best results, the Random Forest algorithm was chosen as the classifier in the anomalous data verification part that is presented in the next section.

6 IoT Behaviour

Identifying the device type is just the beginning when you are focused on ensuring the operation and security of an IoT network. To ensure that devices do not pose a risk to the integrity of the network it is necessary to know when they are presenting anomalous behaviour. After achieving the best results in device identification using the Random Forest classifier in the previous chapter, this algorithm was chosen to identify anomalous network data.

6.1 Anomalous Database

To create a database with anomalous data, we used data from the Bot-IoT database in Koroniotis et al. which is available online[7]. The database incorporates legitimate and simulated botnet traffic and contains data from various types of known attacks to an IoT environment. For this work, we chose databases with DDoS attacks such as 'HTTP flood', 'TCP flood', and 'UDP flood'. The choice was made due to the fingerprinting attributes used in which it is possible to distinguish between common and malicious data packets. The anomalous database is composed of packets of each type of DDoS attack where each type of attack represents 1/3 of the database, having in total the same number of packets as the IoT device database used by Miettinen et al. for device identification.

6.2 Anomaly Identification by Device

We classified the network packets into normal or anomalous data using the Random Forest classifier with the same settings presented in section 3.2.
The databases for each classifier are composed of 50% of packets from the device to be tested and 50% of DDoS attack packets from the anomalous database. For the division between training and testing database, cross-validation method and accuracy, precision, and recall evaluation metrics were used again. The results are presented in table 1.9 and their graphical representation in figure 9. The results were very satisfactory with accuracy, precision, and recall with some values reaching 100%.

Device	Accuracy	Precision	Recall
Aria	1.0000	1.0000	1.0000
D-LinkCam	0.6840	0.8615	0.4457
D-LinkDoorSensor	0.7619	0.7273	0.7619
D-LinkHomeHub	0.8961	0.8734	0.8961
D-LinkSensor	0.8955	0.9091	0.8955
D-LinkSiren	0.8500	0.8500	0.8500
D-LinkSwitch	0.8493	0.9254	0.8493
D-LinkWaterSensor	0.9054	0.8272	0.9054
HueBridge	0.7692	0.7627	0.7692
EdimaxPlug1101W	0.6502	0.6500	0.7950
EdimaxPlug2101W	0.6890	1.0000	0.6650
EdnetGateway	1.0000	1.0000	1.0000
HomeMaticPlug	1.0000	1.0000	1.0000
Withings	1.0000	1.0000	1.0000
HueSwitch	0.8438	0.8438	0.8438
MAXGateway	1.0000	1.0000	1.0000
TP-LinkPlugHS100	0.8333	0.8333	0.8333
TP-LinkPlugHS110	0.7880	0.7500	0.6950
WeMoInsightSwitch	0.9342	0.9861	0.9342
WeMoLink	0.9825	0.9180	0.9825
WeMoSwitch	0.9070	0.9286	0.9070
Average	0.8685	0.8879	0.8585

Table 1.9: Identification by device

[7] Bot-IoT database is available on https://research.unsw.edu.au/projects/bot-iot-dataset

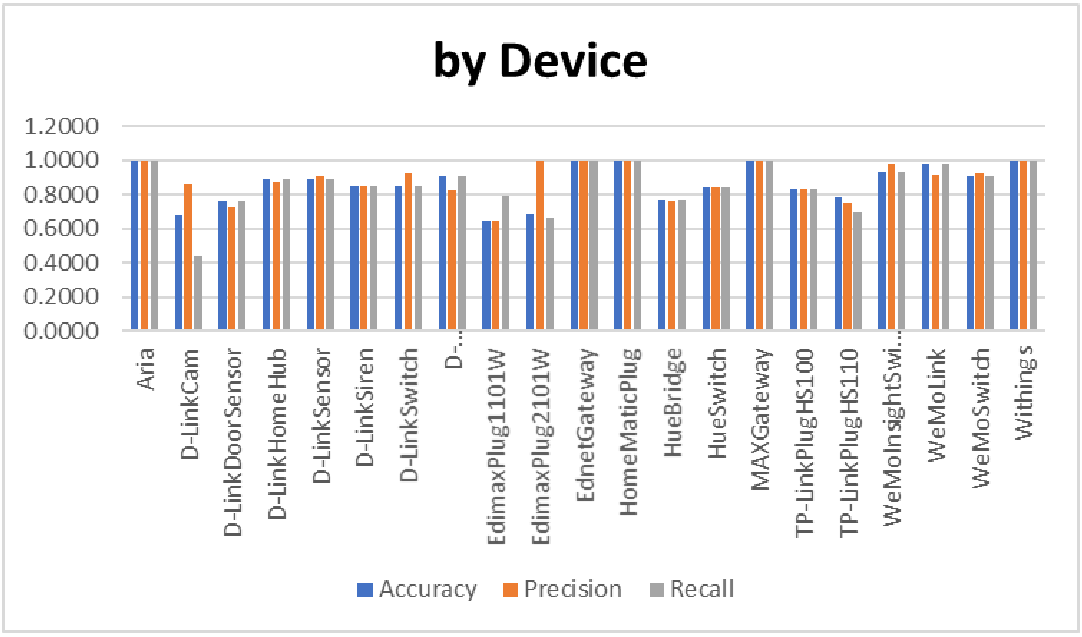


Figure 9: Results by device

6.3 Anomaly Identification by Category

After getting good results in differentiating between common and anomalous IoT data presented by single device classifiers, the final stage is to group the devices that belong to the same category and verify if the Random Forest classifier maintains good performance and results. The 21 devices are grouped into 7 different categories that are presented in table 2.0. With the categories determined, eight new databases were set up in which 50% consisting of data packets from their respective devices and 50% consisting of anomalous database packets. The training and testing methods and metrics used were the same as for single device rating. The results obtained by the classifiers in the tests are presented in table 2.1 and its graphical representation in figure 10.

Category	**Accuracy**	**Precision**	**Recall**
Smart sensor	0.8543	0.8212	0.8543
Smart plug	0.7921	0.8467	0.7977
Smart hub	0.9151	0.9329	0.9151
Smart scale	1.0000	1.0000	1.0000
Smart switch	0.9168	0.9191	0.9168

Table 2.1: Identification by category

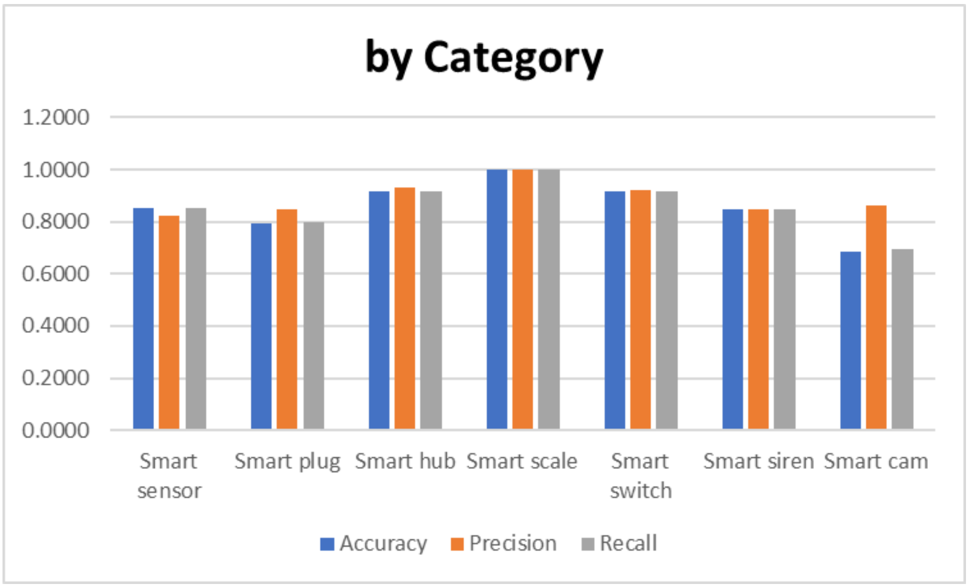


Figure 10: Identification by category

6.4 Discussion

Starting with the results of the first step in classifying the type of device in which the package being analysed belongs (chapter 5), Random Forest classifiers presented the best metrics almost unanimously. When we take the accuracy metric, in table 1.5 it is possible to observe that apart from two devices Random Forest was in the first place, being in second place for D-LinkWaterSensor and WemoSwitch devices where SVM classifier had better results for both devices. For some devices, Random Forest were giving equal results to other classifiers. For the rest of the devices, Random Forest was in first place with the best accuracy among all classifiers being therefore used in the classification between common and anomalous IoT data. Comparing to the work of Miettinen et al., the authors used the same database and Random Forest classifier to identify devices connected to the network through device fingerprinting, however, with another set of attributes. The minimum accuracy results in this work were higher using the classifier for the EdimaxPlug1101W device that presented accuracy of 69.5% being the worst result for the Random Forest while for Miettinen the worse accuracy results for his results showed below 40%. For some of the other devices, Miettinen achieved 100% accuracy but for these same devices, our work also obtained good results of accuracy.
In the second part of the work, the results of the metrics both in the classification of data by individual device and by device category were very similar for all metrics (accuracy, precision, and recall). Random Forest turned out to be a great classifier for differentiating between common and anomalous IoT data with far superior results compared to the device type classification presented in chapter 5. This shows that there is a big difference between common and anomalous data packets and that it is easier for Random Forest classifiers to differentiate between these packets than between different device packets. Many factors contributed to these results, from the size of the database available for each classifier to the treatment of attributes. Perhaps what most impacted the results were the database attributes chosen for the device fingerprinting. That is the reason we believe that the characteristics chosen for device fingerprinting were the great differential for the results obtained.

7 Conclusion and Future Work

This work aimed to use machine learning techniques to correctly classify and identify IoT network traffic data. We explored both the classification of traffic between different devices and between common and anomalous traffic packets. For this, the Device Fingerprint (DFP) technique was used to identify IoT devices through network packet samples. To classify traffic, we performed analyses on which network characteristics and machine learning algorithms are most commonly used in the area of classification of IoT devices and identification of malicious traffic.

In the device classification section, Multilayer Perceptron, K-nearest Neighbors, Support Vector Machine, and Random Forest algorithms were tested and ranked according to Accuracy, Precision, and Recall metrics in addition to the Friedman and Nemenyi statistical test. After the tests, the Random Forest classifier presented the best results and was chosen to perform the classification between common and anomalous traffic. For the classification between common and malicious traffic data, we performed separate tests with classifiers by device and by device category where both presented a high rate of success in differentiating between common and anomalous traffic packets of IoT devices.

One of the ideas for future work is to use derived attributes by developing an analysis of the packet flow in a temporal order instead of sorting only single packets in random order.

Furthermore, it would be interesting to implement Deep Learning algorithms, such as Convolutional Neural Network and Recurrent Neural Network, which showed good results in correlated works to classify and identify anomalous data differentiating the type of attack detected.

References

Aljawarneh, S., Aldwairi, M. and Bani Yassein, M. (2017) ‘Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model’, *Journal of Computational Science*. doi: 10.1016/j.jocs.2017.04.009.

An Evolutionary Study of IoT Malware (no date). Available at: https://ieeexplore.ieee.org/document/9369383 (Accessed: 16 August 2021).

An, N. *et al.* (2017) ‘Behavioral anomaly detection of malware on home routers’, in *2017 12th International Conference on Malicious and Unwanted Software (MALWARE). 2017 12th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 47–54. doi: 10.1109/MALWARE.2017.8323956.

Aneja, S., Aneja, N. and Islam, M. S. (2018) ‘IoT Device Fingerprint using Deep Learning’, in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS). 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, pp. 174–179. doi: 10.1109/IOTAIS.2018.8600824.

Aslan, Ö. A. and Samet, R. (2020) ‘A Comprehensive Review on Malware Detection Approaches’, *IEEE Access*, 8, pp. 6249–6271. doi: 10.1109/ACCESS.2019.2963724.

Bai, L. *et al.* (2018) ‘Automatic Device Classification from Network Traffic Streams of Internet of Things’, *arXiv:1812.09882 [cs]*. Available at: http://arxiv.org/abs/1812.09882 (Accessed: 5 August 2021).

Bezawada, B. *et al.* (2018a) 'Behavioral Fingerprinting of IoT Devices', in *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security*. New York, NY, USA: Association for Computing Machinery (ASHES '18), pp. 41–50. doi: 10.1145/3266444.3266452.

Bezawada, B. *et al.* (2018b) 'IoTSense: Behavioral Fingerprinting of IoT Devices', *arXiv:1804.03852 [cs]*. Available at: http://arxiv.org/abs/1804.03852 (Accessed: 23 August 2021).

Garrett, T. *et al.* (2018) 'Traffic Differentiation on Internet of Things', in *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pp. 142–151. doi: 10.1109/SOSE.2018.00026.

Gibert, D., Mateu, C. and Planes, J. (2020) 'The rise of machine learning for detection and classification of malware: Research developments, trends and challenges', *Journal of Network and Computer Applications*, 153, p. 102526. doi: 10.1016/j.jnca.2019.102526.

Hadiprakoso, R. B., Kabetta, H. and Buana, I. K. S. (2020) 'Hybrid-Based Malware Analysis for Effective and Efficiency Android Malware Detection', in *2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*. *2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, pp. 8–12. doi: 10.1109/ICIMCIS51567.2020.9354315.

Hussain, F. *et al.* (2020) 'Machine Learning in IoT Security: Current Solutions and Future Challenges', *IEEE Communications Surveys Tutorials*, 22(3), pp. 1686–1721. doi: 10.1109/COMST.2020.2986444.

Koroniotis, N. *et al.* (2018) 'Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset', *arXiv:1811.00701 [cs]*. Available at: http://arxiv.org/abs/1811.00701 (Accessed: 19 August 2021).

Mehra, M., Paranjape, J. N. and Ribeiro, V. J. (2021) 'Improving ML Detection of IoT Botnets using Comprehensive Data and Feature Sets', in *2021 International Conference on COMmunication Systems NETworkS (COMSNETS)*. *2021 International Conference on COMmunication Systems NETworkS (COMSNETS)*, pp. 438–446. doi: 10.1109/COMSNETS51098.2021.9352943.

Miettinen, M. *et al.* (2017) 'IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT', in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2177–2184. doi: 10.1109/ICDCS.2017.283.

PAN, J. (2021) 'Iot Network Behavioral Fingerprint Inference With Limited Network Traces For Cyber Investigation', in *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 263–268. doi: 10.1109/ICAIIC51459.2021.9415273.

Patel, Z. D. (2018) 'Malware Detection in Android Operating System', in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. *2018 International Conference on Advances in Computing, Communication*

Control and Networking (ICACCCN), pp. 366–370. doi: 10.1109/ICACCCN.2018.8748512.

Priyadarshan, P. *et al.* (2021) ‘Machine Learning Based Improved Malware Detection Schemes’, in *2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence). 2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pp. 925–931. doi: 10.1109/Confluence51648.2021.9377123.

Ray, S. (2019) ‘A Quick Review of Machine Learning Algorithms’, in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 35–39. doi: 10.1109/COMITCon.2019.8862451.

Roseline, S. A. *et al.* (2019) ‘Towards Efficient Malware Detection and Classification using Multilayered Random Forest Ensemble Technique’, in *2019 International Carnahan Conference on Security Technology (ICCST). 2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–6. doi: 10.1109/CCST.2019.8888406.

Shahid, M. R. *et al.* (no date) ‘Machine Learning for IoT Network Monitoring’, p. 3.

Shaikh, F. *et al.* (2018) ‘A Machine Learning Model for Classifying Unsolicited IoT Devices by Observing Network Telescopes’, in *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC). 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, pp. 938–943. doi: 10.1109/IWCMC.2018.8450404.

Sharma, K. and Nandal, R. (2019) ‘A Literature Study On Machine Learning Fusion With IOT’, in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI). 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1440–1445. doi: 10.1109/ICOEI.2019.8862656.

Wazid, M. *et al.* (2019) ‘IoMT Malware Detection Approaches: Analysis and Research Challenges’, *IEEE Access*, 7, pp. 182459–182476. doi: 10.1109/ACCESS.2019.2960412.

Westyarian, Rosmansyah, Y. and Dabarsyah, B. (2015) ‘Malware detection on Android smartphones using API class and machine learning’, in *2015 International Conference on Electrical Engineering and Informatics (ICEEI). 2015 International Conference on Electrical Engineering and Informatics (ICEEI)*, pp. 294–297. doi: 10.1109/ICEEI.2015.7352513.

Yeo, M. *et al.* (2018) ‘Flow-based malware detection using convolutional neural network’, in *2018 International Conference on Information Networking (ICOIN). 2018 International Conference on Information Networking (ICOIN)*, pp. 910–913. doi: 10.1109/ICOIN.2018.8343255.

Zhou, W. *et al.* (2019) ‘The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved’, *IEEE Internet of Things Journal*, 6(2), pp. 1606–1616. doi: 10.1109/JIOT.2018.2847733.