IFSA

Sergey Y. Yurish
Editor

Advances in Networks, Security and Communications

Volume 3



Sergey Y. Yurish
Editor

Advances in Networks, Security and Communications

Volume 3

International Frequency Sensor Association Publishing

Sergey Y. Yurish
Editor

Advances in Networks, Security and Communications
Volume 3

Copyright © 2022 by International Frequency Sensor Association Publishing, S. L.

E-mail (for orders and customer service enquires): ifsa.books@sensorsportal.com

Visit our Home Page on http://www.sensorsportal.com

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (IFSA Publishing, S. L., Barcelona, Spain).

Neither the authors nor International Frequency Sensor Association Publishing accept any responsibility or liability for loss or damage occasioned to any person or property through using the material, instructions, methods or ideas contained herein, or acting or refraining from acting as a result of such use.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identifies as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

e-ISBN: 978-84-09-47239-0

BN-20221221-XX

BIC: TJK

Contents

Contributors **11**

Preface **13**

1. Novel Results on Blind Symbol Rate Estimation in Digital Satellite Receivers **15**
1.1. Introduction 15
1.2. Signal Model 16
1.3. Cramer-Rao Lower Bound 17
1.3.1. Computation of the Modified CRLB 17
1.3.2. Computation of the Asymptotic CRLB 17
1.4. Estimator Algorithms 20
1.4.1. CCB and ML Approach 20
1.4.2. DF Concept 21
1.5. Numerical Results 23
1.6. Conclusions 26
Acknowledgements 26
References 26
Appendix 28

2. Confidentiality Concerns in Cloud Environments **29**
2.1. Introduction 29
2.2. Overview of Existing Attacks in Cloud Computing 30
2.2.1. Authentication Attacks 30
2.2.2. Man-in-the-middle Attacks 31
2.2.3. Denial of Service Attacks 32
2.2.4. Cloud Malware Injection Attacks 33
2.2.5. Side Channel Attacks 33
2.2.6. Collusion Attacks 33
2.3. Data Confidentiality in Cloud Computing 34
2.3.1. Classification of Solutions Ensuring Data Confidentiality in Cloud Computing 34
2.3.2. Confidentiality Through Biometric Encryption 36
2.3.2.1. Biometric Encryption in Mobile Cloud Computing *36*
2.3.2.2. Privacy-preserving Biometric Identification in Cloud Computing *36*
2.3.2.3. CloudBI *37*
2.3.2.4. BE to Enhance Confidentiality in Cloud Computing *38*
2.3.2.5. Privacy Preserving Security Using Biometrics *38*
2.3.3. Confidentiality Through Homomorphic Encryption 39
2.3.3.1. Cloud El Gamal and Fast Cloud RSA Schemes *39*
2.3.3.2. Fast-variant Cloud Paillier Scheme *40*
2.3.3.3. A Solution Based on Fully Homomorphic Encryption *40*
2.3.3.4. Speedy Cloud-RSA *41*
2.3.3.5. Solution Based on Intel SGX *41*
2.3.4. Confidentiality not by Encryption only 42
2.3.4.1. HPI-Secure *42*
2.3.4.2. K-NN Classifier for Data Confidentiality *43*
2.3.4.3. Steganography-based Solution *44*
2.3.4.4. Confidentiality Preserving Through Encryption and Obfuscation *44*

2.3.4.5. Attribute-based Proxy Re-encryption ... *45*
2.3.4.6. TR-MABE ... *46*
2.3.4.7. Privacy-manager ... *47*
2.3.4.8. Hierarchical Classes of Computing for Privacy Preserving ... *48*
2.3.4.9. Encryption and Trust-based Solution ... *49*
2.3.4.10. Blockchain for Cloud-based Data Privacy ... *50*
2.3.4.11. L-EncDB ... *52*
2.3.4.12. CS-IBE ... *53*
2.3.4.13. GRBAC ... *53*
2.3.4.14. ISRSAC ... *54*
2.3.4.15. Solution Using Modular Encryption Standard ... *55*
2.3.4.16. CPDA ... *56*
2.3.5. Confidentiality According to the Type of Storage ... 57
2.3.5.1. Secret Sharing Scheme ... *57*
2.3.5.2. Scheme for Image Data Privacy ... *57*
2.3.5.3. DCSSP ... *58*
2.3.5.4. CAFE ... *59*
2.3.5.5. ARIANNA ... *60*
2.3.6. Confidentiality Without Encryption ... 60
2.3.6.1. NetDB2-MS ... *60*
2.3.6.2. Multiple Storage Solution ... *61*
2.3.6.3. MONcrypt ... *62*
2.3.6.4. Matrix-based Solution ... *63*
2.4. General Assessment of the Studied Solutions ... 63
2.5. Conclusion ... 67
References ... 68

3. BAC-DC: Blockchain-based Access Control for Decentralized Cloud Data ... 71
3.1. Introduction ... 71
3.2. Background and Related Works ... 72
3.2.1. Meta-key ... 72
3.2.2. Storj ... 73
3.2.3. Blockchain technology for Data Privacy in cloud computing ... 75
3.2.4. BACC (Blockchain-based Access Control for Cloud Data) ... 75
3.2.5. FileCoin ... 76
3.2.6. SIA 77
3.2.7. Discussion ... 77
3.3. Overview of BACC (Blockchain-based Access Control for Cloud Data) ... 79
3.3.1. The Main Components of the System ... 79
3.3.2. System Configuration ... 80
3.3.2.1. Data Upload ... *81*
3.3.2.2. Data Download ... *81*
3.3.3. The Limits of the BACC Scheme ... 81
3.4. Transition from Centralized to Decentralized Storage ... 83
3.4.1. Basic Cloud Computing ... 83
3.4.2. Why Decentralizing the Cloud? ... 83
3.4.3. Decentralized Cloud ... 84
3.5. Proposed Model BAC-DC: Blockchain-based Access Control for Decentralized Cloud Data ... 85
3.5.1. The Main Components ... 85
3.5.1.1. The Nodes ... *85*
3.5.1.2. The Peer-to-peer Communication and Discovery System ... *85*
3.5.1.3. The Reference Table or Metadata ... *86*

3.5.2. BAC-DC: Operating Mode....86
3.5.2.1. Data Upload....86
3.5.2.2. Data Download....87
3.5.3. Model Architecture....88
3.6. Link between Our BAC-DC Scheme and the One Studied BACC....89
3.7. Security Analysis....90
3.7.1. File Partitioning Before Storage....90
3.7.2. End-to-end Encryption....91
3.7.3. Parity Fragments for Redundancy....92
3.7.4. File Verification....92
3.7.5. Data Loss....92
3.7.6. Access Control....92
3.8. Security against Attacks....94
3.8.1. The Google Attack....94
3.8.2. The Sybil Redundancy Attack....95
3.8.3. The Incorrect Distribution....95
3.8.4. Cheating Client Node....95
3.8.5. The Hostage Byte Attack....95
3.8.6. Geppetto's Honest Attack....96
3.9. Advantages and Disadvantages of the BAC-DC Model....96
3.9.1. Advantages....96
3.9.2. Disadvantages....97
3.10. Conclusion....98
References....98

4. Attack Detection in IoT Networks Using Intelligent Classification Mechanisms....101
4.1. Introduction....101
4.2. Security Issues in IoT Networks....102
4.2.1. Physical Threats....102
4.2.2. Cyber Threats....103
4.2.2.1. Denial of Service: DoS / DDoS....103
4.2.2.2. Mirai....103
4.2.2.3. R2L (Remote to Local)....103
4.2.2.4. U2R (User to Root)....104
4.2.2.5. Man-in-the-middle....104
4.2.2.6. Injection Attack....104
4.2.2.7. Password Attack....104
4.3. Defense Solutions Associated with an IoT System....105
4.3.1. Network Based Intrusion Detection System (NIDS)....105
4.3.2. Host Based Intrusion Detection System (HIDS)....106
4.4. Anomaly Detection by Classification....106
4.4.1. Data Classification in AI....107
4.4.2. Classification Types....107
4.4.3. Classification Algorithms....108
4.4.3.1. Linear Models....108
4.4.3.2. Nonlinear Models....109
4.4.4. Anomaly Detection by Binary Classification....109
4.4.5. Detection and Identification of Anomalies by Multiclass Classification....110
4.5. Presentation of Used Tools, Datasets and Methods....111
4.5.1. Tools and Libraries....111
4.5.2. Dataset 1: UNSW-NB15....113
4.5.3. Dataset 2: IoT Network Intrusion Dataset....116
4.5.4. Method....117

4.5.5. Data Pre-processing 118
4.5.5.1. Shape Analysis 118
4.5.5.2. Background Analysis 121
4.5.5.3. Data Preparation 124
4.6. Results and Discussion 125
4.6.1. Evaluation Metrics 125
4.6.1.1. Confusion Matrix 125
4.6.1.2. Metrics 126
4.6.2. Analysis and Interpretation of Results 126
4.6.2.1. Anomaly Detection 126
4.6.2.2. Identification of Attacks 132
4.7. Conclusion 137
References 139

5. Emerging Trends Using IoMT for Healthcare System 141
5.1. Introduction 141
5.2. Literature Review 142
5.3. General Framework of IOMT 142
5.4. IoMT Conceptual Frameworks and Architecture 143
5.5. Technologies & Platforms behind IoMT 144
5.5.1. Device Integration Layer 146
5.5.2. Information Integration Layer 146
5.5.3. Information Extraction or Information Representation Layer 146
5.6. IoMT Communication Technologies 146
5.7. Typical IoMT-based E-health System 147
5.8. Real Time Health Monitoring Systems 149
5.9. Cloud, Fog & Edge Computing 150
5.10. Security Aspects in IoMT Communications 151
References 153

Index 157

Contributors

Cherif Diallo
Laboratoire d'Algèbre, Cryptographie, Codes et Applications (LACCA),
Dept. Informatique, UFR SAT, Université Gaston Berger, BP 234, Saint-Louis, Sénégal
E-mail: cherif.diallo@ugb.edu.sn

Wilfried Gappmair
Institute of Communication Networks and Satellite Communications,
Graz University of Technology, Austria

P. Madhavasarma
SASTRA Deemed University, Thanjavur 613401, Tamilnadu, India
E-mail: spms1975@yahoo.co.in

T. N. Prabakar
Sri Sairam College of Engineering, Bangalore, India

Barbara Suesser-Rechberger
Institute of Communication Networks and Satellite Communications,
Graz University of Technology, Austria

P. Veeraragavan
University College of Engineering, Tindivanam 604001, Tamilnadu, India

Sarah Wayzani
Laboratoire d'Algèbre, Cryptographie, Codes et Applications (LACCA),
Dept. Informatique, UFR SAT, Université Gaston Berger, BP 234, Saint-Louis, Sénégal

Preface

After successful publication of the first two volumes of '*Advances in Networks, Security and Communications'* open access Book Series in 2017 (Volume 1) and in 2019 (Volume 2), and positive feedback from our authors and readers, we have decided to publish the 3rd volume of this Book Series in 2022.

This volume contains 5 chapters written by 7 authors from 3 countries: Austria, India and Senegal. But it is not a simple set of reviews. Each of the chapters contains the extended state-of-the art followed by new, obtained by the authors results, unpublished before.

Chapter 1 presents in details the asymptotic Cramer-Rao lower bound (ACRLB) for carrier-blind non-data-aided (NDA) estimation of the symbol rate. This figure of merit represents the theoretical limit of the jitter variance produced by any estimator in the lower signal-to-noise ratio (SNR) range. It could be observed that the SNR value characterizing the intersection of ACRLB and modified Cramer-Rao lower bound (MCRLB) might be helpful in practice. Furthermore, a novel estimator algorithm has been discussed, which suffers only marginally from error floor effects, although this solution requires a second filter operated in parallel to the receiver matched filter.

The confidentiality concerns in cloud environments are considered in the Chapter 2. The recent solutions concerning this subject have been described by the authors. This will allow to identify a better perspective that will help to develop in future an effective cloud strategy based on existing solutions in order to counter the security challenges regarding data confidentiality.

The blockchain-based access control for decentralized cloud data proposed in Chapter 3 brings many advantages in terms of security. Authors have integrated the blockchain technology into the cloud while emphasizing access control managed by specific nodes called master nodes and not by a central entity. Indeed, it responds to security vulnerabilities identified in existing decentralized cloud computing solutions. Among the many advantages, it is very profitable, emphasizes data confidentiality, offers good redundancy and unlimited bandwidth.

Chapter 4 describes the models and techniques for intelligent classification of network traffic in order to increase accuracy in the anomaly detection and attack identification in IoT environments. The implementation of several artificial intelligence classification models namely a decision tree, random forest, AdaBoost and an artificial neural network for anomaly detection and attack identification in IoT networks have been analyzed and discussed.

Chapter 5 is focused on the emerging trends using Internet of Medical Things (IoMT) for healthcare system. The frameworks, architectures, technologies and platforms of IoMT on the different integration levels are described in the Chapter.

Every chapter in '*Advances in Networks, Security and Communications'* Vol. 3 is independent and self-contained. The book can be useful for post-graduate students, researchers, engineers and scientist working in the appropriate areas.

I shall gratefully receive any advices, comments, suggestions and notes from readers to make the next volumes of '*Advances in Networks, Security and Communications*'open access book Series very interesting and useful.

Dr. Sergey Y. Yurish

Editor
IFSA Publishing *Barcelona, Spain*

Chapter 4
Attack Detection in IoT Networks Using Intelligent Classification Mechanisms

Chérif Diallo

4.1. Introduction

For years, the evolution of the Internet has revolutionized the interaction of living beings with their environment, allowing communication with or between objects thanks to the presence of small devices called sensors. These sensors, which can be inserted into various objects, are capable of transforming an observed physical quantity into understandable and usable data via the Internet infrastructure. This evolution of a larger and more heterogeneous Internet, integrating diverse and varied objects, is known as the Internet of Things (IoT). In recent years, the IoT has experienced incredible growth thanks to the many applications derived from this technology. This therefore results in a considerable increase in data flows in this type of network of connected objects. These data contain useful and valuable information from the various network objects and are applicable in several areas of life such as agriculture, health, industry, connected cities and in particular home automation or smart homes. This large amount of data has sparked a proliferation of hackers who have carried out numerous attacks on IoT networks at the risk of becoming IoT Botnets (connected devices infected with malware). Thus, these attacks compromise the confidentiality, integrity and availability of data in the IoT. The vulnerability of these IoT networks is partly explained by the limitations of traditional security methods. This calls into question the security of IoT networks, hence the interest in using an intelligent system for classifying traffic data and detecting malicious flows. Indeed, the use of tools such as those of Artificial Intelligence (AI) could be an effective solution to identify and detect in an IoT network the presence of traffic that presents anomalies resulting from attacks or network malfunctions. In addition, AI allows to go even further in anomaly detection with the power of Deep Learning (DL), Machine Learning (ML) and other models based on artificial neural networks.

Based on these observations and given the importance of cybersecurity in the IoT, the object of this work is to propose models and techniques for intelligent classification of

Chérif Diallo
Laboratoire d'Algèbre, Cryptographie, Codes et Applications (LACCA), Dept. Informatique, UFR SAT, Université Gaston Berger, BP 234, Saint-Louis, Sénégal

network traffic in order to increase accuracy in the anomaly detection and attack identification in IoT environments.

Therefore, knowing how to extract abnormal traffic in order to identify potential attacks seems to be a task of primary importance. Here, we first seek to detect anomalies in smart home IoT network traffic through binary classification. Then we try to identify the attacks contained in the anomalous traffic by using learning models based on multiclass classification. In each of these two cases, we used two different datasets to study and compare four learning models based on Decision-Tree (Adaboost), Random Forest, KNN (K-Nearest Neighbors) and a Neural Network model. The results show that our models based on Decision-Tree and Random Forest are more efficient both in the detection of anomalies and in the identification of attacks. This is also the case both in predictions and in behavior with new data.

To achieve our objectives, we have organized this chapter as follows. We first address security issues in IoT networks. Then, we present the defense solutions associated with an IoT system before the section relating to the detection of anomalies by classification. Finally, we present our contribution which consists of the establishment of classification methods for anomaly detection and attack identification. In this part we present the tools, datasets and methods used as well as the models used in the detection of anomalies by classification based on the techniques of Machine Learning and neural networks. We end this chapter, which is an extended version of preliminary work reported in [1], by evaluating the performance of the different models in the results and discussion section.

4.2. Security Issues in IoT Networks

Security remains one of the biggest challenges that the IoT must overcome. Cybersecurity experts call IoT one of the most vulnerable technologies. Indeed, IoT systems have become a choice for attackers whether as a target or even as an attack tool, so it is essential to secure them.

The IoT uses the internet at the core of its network, so many vulnerabilities and challenges are due to it because its main problem is security. On the other hand, these vulnerabilities can be related to the localization of objects or sensors. This is how we distinguish two levels of types of attacks which are physical threats and cyber threats.

4.2.1. Physical Threats

Physical threats refer to all attacks carried out at the level of connected objects or even at the level of sensors. The limited computational and power constraints of the nodes are an obstacle to applying security countermeasures from conventional computer networks to wireless sensor networks (WSN). Moreover, in the latter, nodes become more vulnerable when placed in unprotected environments such as public spaces. Under these circumstances, attackers can easily capture nodes, access confidential information contained in their memories (e.g., cryptographic keys) and reprogram their behavior. Too

exposed, these objects can also suffer damage from natural disasters or malicious human actions.

4.2.2. Cyber Threats

Rather, it is the logical threat to the functional network of an IoT solution. It is also common for attackers to take advantage of the wireless nature of communications to intercept messages or jam the frequency band to prevent certain packets from being received correctly. At this level, we have mainly two types of threats:

– Passive threats: the latent risk is certainly driven by clandestine eavesdropping of correspondence flows or the system. By listening, the offense can gather data from the instrument, the proprietary instrument, or both;

– Active Threats: The attack is not only about cleverly eavesdropping on inter-object communications, but also altering IoT structures to change their design, regulate their interactions, deny them support, etc. Attacks can incorporate an array of malicious disruptions and modifications.

In an IoT system, there are several types of attacks, whether active or passive, each having a target at the layers of the basic architecture. Passive attacks are also often used to set up active attacks. We will explain some of these most common attacks.

4.2.2.1. Denial of Service: DoS / DDoS

A DoS attack is an attack in which the attacker attempts to make a machine or network resource unavailable to users by disrupting services. A DoS attack could therefore flood the targeted object with multiple useless requests thus trying to overload the structures and prevent some or all legitimate requests from being satisfied.

As for the DDoS attack, it uses various malicious sources to send an unlimited number of requests to the target using illegal source addresses, thus making it more difficult to identify the source of the attack.

4.2.2.2. Mirai

It is malware used to scan the internet and identify a large number of IoT devices, especially IP cameras and routers, which run Linux and use default credentials for Telnet connections. It is more often used for DDoS attacks. Attackers managed to turn these devices into BOTnet and use them to cause Denial of Service (DoS) attacks on DNS servers. This consequently results in breaking access to some popular websites.

4.2.2.3. R2L (Remote to Local)

R2L is a remote attack towards the target user of one or more systems connected by a network and installs files or malicious software there. The attacker here tries to find

vulnerabilities in the system to access personal data. The advantage of the attackers here is that they can steal valuable information from the user by accessing the system remotely. The attacker inserts a program or other device, called a probe, at an important point or junction in a network. The purpose of the probe is to monitor and collect information about network activity as well as weak spots in network devices or connections that could compromise the system.

4.2.2.4. U2R (User to Root)

U2R, or User to Root attack, uses certain privileges as a normal user and then later tries to exploit system vulnerabilities to gain root benefits. Simply, it is the act of obtaining root privileges for a non-root user with legal access to a local machine.

4.2.2.5. Man-in-the-middle

As the name suggests, it is interference where an attacker infiltrates communication between two or more parties who are unaware of the existence of this attacker. The attacker can either passively receive the information exchanged by the two participants or actively interfere and change the data or information that is communicated. To be explicit, he acts as an intermediary between the two interlocutors without the latter having any suspicions or any sign of attack due to normal communication or the absence of unusual events during their exchange. This type of attack could be dangerous due to the possession of the original message by the attacker who can trick the recipient into believing that the message he receives is legitimate.

4.2.2.6. Injection Attack

Injection attacks refer to a broad class of attack vectors that allow a hacker to gain entry into a program, interpreted as a command or request that alters the flow of software execution. Injection attacks are among the oldest and most dangerous web application attacks. These attacks can lead to data theft and/or loss, denial of service, and even complete system takeover. There are different types of injection attacks such as code injection, cross-site scripting, email injection, SQL injection, etc. What makes injection attacks particularly worrying is their huge attack surface (especially for SQL and XSS). There are countless free and reliable tools available that allow even inexperienced hackers to abuse these vulnerabilities.

4.2.2.7. Password Attack

Since passwords are the most commonly used mechanism to authenticate users of a computer system, obtaining passwords is therefore a common attack approach. A person's password can be obtained by searching the person's physical office, monitoring the network connection to acquire unencrypted passwords, accessing a password database, or simply guessing. This last "guessing" approach can be done randomly or systematically:

- Brute force attacks consist of taking a random approach i.e., trying different passwords and hoping that one of them will work. Some logic can be applied: try passwords related to the person's name, job title, hobbies, or similar;

- In a dictionary attack, a dictionary of common passwords is used to attempt to gain access to a user's computer and network. One approach is to copy an encrypted file containing the passwords, apply the same encryption to a dictionary of commonly used passwords, and compare the results.

4.3. Defense Solutions Associated with an IoT System

All IoT security is based on a stability of integrity which is the property that ensures that information cannot be falsified, modified, altered or deleted by an unauthorized user or unauthorized object; confidentiality which ensures that information can only be read by authorized entities; authenticity, which makes it possible to manage the secure recognition of an entity's identity, and availability, which is the mechanism for guaranteeing the proper execution of a service, and the proper functioning of the network data system. To ensure this security, we have traditional solutions and intrusion detection systems (IDS).

Even if classic solutions are proposed to avoid or limit attacks, they are not effective and applicable in all IoT networks, especially those on a large scale. With the increase in the large amount of data due to the many connected objects, IoT networks are increasingly exposed to attacks. These attacks are very dangerous because they slow down networks, cause large amounts of dropped packets, drain node batteries, and can compromise multiple services. This is how intrusion detection systems or IDS approach has been taken to limit or even eliminate attacks. IDS is a mechanism that listens to network traffic furtively in order to identify abnormal or suspicious activities and thus allows preventive action to be taken on the risks of intrusion by sending an alert to the administrator. We distinguish two types of intrusion detection systems: NIDS and HIDS.

4.3.1. Network Based Intrusion Detection System (NIDS)

NIDS or Network Based Intrusion Detection Systems are hardware or software systems that scan the network to identify network interference or any threats. They are placed at strategic points in the network so that they can keep an eye on the devices in the network. Once an intrusion is detected, an alert is sent to the administrator. When the NIDS has response capability, it is called a Network Intrusion Prevention System (NIPS). A NIPS differs from a NIDS in that it is able to modify network traffic flows to block an attack in progress for example.

NIDS use several solutions for the detection of attacks or anomalies among which we find mainly:

- **By signature**: a NIDS can use the signature method to verify the authenticity of network traffic and only detects specific attacks. Indeed, they use patterns such as sequences of TCP flags, bytes, and intrusions also called signatures stored in the NIDS

database. And when we have an input traffic sequence, the latter is compared to those stored in the database in order to identify the attacks and they can only detect the attacks for which a signature exists. Alarms are triggered once a sequence of events that matches one of the signatures is identified. The main advantage of this type of detection is the low rate of false positives (normal data considered as attacks), and it also has a low rate of false alarms. In the context of IoT applications in smart cities, signature-based detection is useful for identifying attacks targeting networks with regular behavior. Therefore, they fail to detect new attacks such as attacks exploiting zero-day vulnerabilities.

- **By anomaly detection**: the anomaly detection method is also used by NIDS and depends on the behavior of the system. When an attacker tries to change normal system behavior into abnormal behavior, the anomaly detection system detects it. NIDS based on anomaly detection learns the profile of network behaviors. When network traffic does not conform to the learned legitimate behavior profile, an alert is triggered. Alternatively, intrusion detection techniques looking for anomalies are able to identify changes in the system that are inconsistent with normal behavior. The main advantage of anomaly detection systems is that they can detect new attacks or so-called zero-day attacks.

4.3.2. Host Based Intrusion Detection System (HIDS)

HIDS or host-based intrusion detection systems play a vital role in the network. It is used to detect attacks or malware on host peripheries. It steps in where NIDS failed to detect intrusions.

4.4. Anomaly Detection by Classification

Anomaly detection becomes an adaptive solution to solve cybersecurity problems of an IoT network. Indeed, this anomaly detection amounts to distinguishing all suspicious and rare events or observations called outliers that reveal differences with the majority of other data. However, there are several types of data with different characteristics, which implies the existence of different anomaly detection methods, each finding its effectiveness in a particular field according to a defined objective. Most of these methods use decision thresholds based on techniques based on classification, clustering, regression, nearest neighbours and statistical tools. Each method is evaluated by measuring its performance of detection accuracy criteria, its speed and its scalability concerning the flow rate.

Anomaly detection techniques could be classified according to 4 categories:

- **Statistical techniques**: they can be parametric or non-parametric whose models are built based on a confidence interval from existing data and data that does not correspond to this model are considered abnormal;
- **Techniques based on proximity**: it includes the technique based on the nearest neighbours and clustering. They both use the distance approach and the density approach. Indeed, clustering classifies data according to their similarity;

- **Deep Learning-based techniques**: represent supervised and unsupervised deep learning algorithms for classification-based anomaly detection;
- **Other techniques**: other techniques include models based on machine learning and neural networks.

4.4.1. Data Classification in AI

Classification is a process of classifying a given set of data into class. It can be performed on structured or unstructured data. The process begins with the prediction of the class of given data points. Classes are often called targets, labels, or categories:

- **Classifier**: A classifier is an algorithm used to classify input data into a specific category;
- **Classification model**: a classification model predicts all other conclusions to the input data for training;
- **Feature**: a property of the observed phenomenon;
- **Classification procedure**:
  1) Initialize;
  2) Train;
  3) Predict;
  4) Evaluate.
- **Lazy Learners**: Lazy Learners are learning types that first store the training dataset and wait to receive the test dataset. In the case of lazy Learners, the classification is performed based on the most relevant data stored in the training dataset. It takes less time for training but more time for predictions. Example: KNN algorithm, Case-based reasoning;
- **Eager Learners**: Eager Learners develop a classification model based on a training dataset before receiving a test dataset. Unlike Lazy Learners, Eager Learners take longer to learn and less time to predict. Example: Decision trees, Naïve Bayes, ANN.

4.4.2. Classification Types

Binary classification: Binary classification is a type of classification with only two classes as results which can be for example true or false, one or zero, or yes or no. Binary classification is used in several applications in data science such as medical diagnosis (healthy or sick), email analysis (Spam or not Spam), financial data analysis (fraud or not fraud), marketing (will buy or will not purchase), anomaly detection (normal or abnormal), etc. In machine learning, there are many methods used for binary classification. The most common are: Support Vector Machines, Naive Bayes, Nearest Neighbour, Decision Tree, Logistic Regression, Neural Networks, etc.

Multi-class classification: a classification with more than two classes is called multi-class classification. It is a classification technique that allows us to classify the test data into multiple class labels present in the trained data as a model prediction. For example, using a pattern to identify the types of attacks in a network is an example of multiclass classification because there are many different attack classifications for each network traffic. Multiclass classification also requires that a sample has only one class (e.g. a blackhole attack is only a blackhole; it is not so man-in-the-middle). There are mainly two types of multi-class classification techniques:

- **One vs. All or One-vs-rest (One against all)**: In One vs. All, for a dataset of N class instances, we need to generate N binary classifier models. The number of class labels present in the dataset and the number of generated binary classifiers must be identical;
- **One vs. One**: In One-vs-One classification, for the dataset of N class instances, we need to generate the N*(N-1)/2 binary classification models. Using this classification approach, we split the primary dataset into a dataset for each class versus each other class.

Multi-Label Classification: This is a type of classification in which each sample is assigned to a set of labels or targets. In multi-label classification, the training set is composed of instances each associated with a set of labels (labels), and the task is to predict the label sets of unseen instances by analyzing the instances of training with known sets of labels. The difference between multi-class classification and multi-label classification is that in multi-class problems the classes are mutually exclusive whereas in multi-label problems each label represents a different classification task, but the tasks are somehow linked. For example, multi-class classification assumes that each sample is assigned to one and only one label: a fruit can be either an apple or a pear, but not both at the same time. In contrast, an example of multi-label classification might be that a text might be about religion, politics, finance, or education all at once or none of these topics.

4.4.3. Classification Algorithms

4.4.3.1. Linear Models

Logistic Regression: This is a machine learning classification algorithm that uses one or more independent variables to determine an outcome. The outcome is measured using a dichotomous variable, which means that there are only two possible outcomes.

Support Vector Machines: Support Vector Machine (SVM) is a supervised learning approach that is used to analyze data for classification to achieve optimal results. The concept of SVM is as follows: the input vectors are first transposed into a multidimensional space. Then, the optimal separation between the data is found to classify the data between different classes. SVMs are not recommended to use large datasets because training takes a long time.

4.4.3.2. Nonlinear Models

Naïve Bayes: This is a classification algorithm based on Bayes' theorem which gives an assumption of independence between the predictors. Simply put, a Naive Bayes classifier assumes that the presence of a feature in a class is unrelated to the presence of any other feature.

Stochastic Gradient Descent: This is a very effective and simple approach to fitting linear models. Stochastic gradient descent is particularly useful when the sample data is large. It supports different loss functions and penalties for classification.

K-Nearest Neighbour: It is a lazy learning algorithm that stores all instances corresponding to the training data in an n-dimensional space. It is a lazy learning algorithm because it does not focus on building a general internal model but on storing instances of the training data.

Decision Tree: The Decision Tree is a classification technique in machine learning based on the "divide and conquer" strategy. The decision tree consists of nodes and leaves, where the nodes represent a test on one of the features and the leaves represent the class label. It can vary from a very simple model to a complex combined model, which makes it suitable for anomaly detection systems.

Random Forest: Random decision trees (or Random Forest) are a set of learning methods for classification, regression, etc. It works by building a multitude of decision trees at the time of training and produces the class which is the mode of the classes or the classification or the average prediction (regression) of the individual trees.

Artificial Neural Network: A neural network consists of neurons arranged in layers, which take an input vector and convert it into an output. The process involves each neuron taking an input and applying a function to it, often nonlinear, then passing the output to the next layer.

In the following we will discuss the classification solutions proposed in the literature for the detection of anomalies and then for the identification of the type of anomalies. Binary classification is the one used for detection while for identification, multiclass classification is applied.

4.4.4. Anomaly Detection by Binary Classification

In [2], the authors presented a new intrusion detection system (IDS) for Fog Computing based on decision trees. The security of fog nodes is a challenge as they produce a massive amount of data and have limited computational capabilities. In their study, they used Classification and Regression Trees (CART) which construct binary trees by selecting features that produce the maximum information gain. The evaluation of the proposed IDS system was carried out using the KDDCUP99 dataset and several comparisons between the proposed IDS using decision trees and other approaches were made. The overall

results showed that the decision tree achieved the best detection accuracy, the best attack coverage and an acceptable detection time.

In [3], the authors proposed a traffic anomaly detection algorithm using the SVM algorithm. They also proposed an algorithm that calculates normalized entropy for data features. In the case of attacks, these values deviate from their normal values, so a threshold is set to discover any abnormal behavior in the network. The SVM was used to build a classifier to detect network anomaly traffic. To improve the quality of the SVM parameters, the optimization algorithm, Particle swarm optimization (PSO) was used to estimate the parameters more accurately. To evaluate the performance of the proposed method, DARPA and KDDCup99 datasets were used. The results showed that their method detected different types of attacks with high accuracy.

The authors of [4] have proposed a deep learning approach for an anomaly intrusion detection system using a recurrent neural network (RNN). The RNN is used for supervised classification training which introduces feedback from previous information and applies it to the current output. Model performance was evaluated using the NSL-KDD dataset in two experiments: binary classification and multiclass classification. The number of nodes in the hidden layers and the learning rate were tuned by training the network with different values.

The authors of [5] used an improved terminated backpropagation artificial neural network (RBNN) algorithm to train multi-layered perception. The resilient propagation algorithm performs a direct adaptation of the configuration of the weights based on the local gradient which depends on the state of the network. The proposed system consists of three steps: pre-processing, training and testing. First, in the data pre-processing step, the symbolic features were converted to numeric ones. Next, the dataset is divided into training, validation, and testing sets. Finally, the data are normalized by setting the mean of each line to zero and the standard deviation to one. For the training step, they used an iterative process to determine the best number of neurons in the hidden layer. The final architecture of the neural network consists of 42 neurons in the input layer, 26 neurons in the hidden layer and 5 neurons in the output layer. Then the network was trained by adjusting the error to be less than a fixed threshold. The NSL-KDD99 dataset was used for training and testing. The tests showed that the use of enhanced resilient backpropagation improved the performance of the system in terms of classification rate and epoch number. The system achieved an 89 % detection rate for the test dataset and a 15.7 % false positive rate.

The Euclidean distance map (EDM) is a new anomaly detection technique by sequential algorithms which was presented in [6]. The system analyzes the network traffic and uses distance maps to extract second-order statistics. These second-order statistics are exploited for the generation of new features that improve detection accuracy.

4.4.5. Detection and Identification of Anomalies by Multiclass Classification

In [7], an anomaly intrusion detection system was proposed using the J48 decision tree. The Kyoto 2006+ dataset was used. It is composed of three classes: normal, attack and

unknown attack. This dataset is obtained from different honeypots over three years of real traffic.

The authors of [8] proposed a feature selection method based on the Mutual Information (MI) algorithm. They then built an intrusion detection system by combining several classifiers to distinguish between different classes using the Least Square Support Vector Machine (LS-SVM) algorithm. The algorithm is a symmetric measure of the correlation between two random variables which is used to estimate the relevance between the feature and the class label. To estimate the MI, the entropies of the data were calculated using the average distance from each data point to the K nearest neighbours. To evaluate the performance of the proposed solution, three datasets were used: KDD Cup 99, NSL-KDD, and Kyoto2006+. The experiments showed that the IDS with the feature selection algorithm achieved good performance in network intrusion detection.

The online Average One Dependence Estimator (AODE) was proposed in [9] for a multi-classification problem in the UNSW-NB15 dataset using a supervised machine learning algorithm. The proposed classifier updates data over time to secure a dynamic network. The results show that AODE performs better than Naïve Bayes (NB).

There are also several other methods described in [10-19], but in most cases these are isolated cases of use. This is why, in this work, we propose to combine binary classification (for anomaly detection) with multiclass classification (for attack identification). In each case, we comparatively evaluate four different models (Decision-Tree, Random Forest, KNN and a Neural network model), each in two separate datasets (one dataset with more normal traffic than anomalies and another with more anomalies than normal traffic), in order to uncover the best possible approach.

4.5. Presentation of Used Tools, Datasets and Methods

4.5.1. Tools and Libraries

Python: The choice of the appropriate programming language for our work is focused on Python. It is an object-oriented programming language created in 1989 by Guido Van Rossum with characteristics that mark its performance such as portability, dynamism, extensibility and modularity. Its use is frequent in the development of machine and deep learning models thanks to the many libraries it holds, which makes their implementation easier.

Anaconda: Anaconda is a free and open-source distribution used to make programs in Python and R languages for data science and machine learning concerning predictive analysis, data processing and scientific calculations. It has more than 250 popular data science packages and a browser called anaconda browser which is a graphical interface for users to easily access applications or tools such as Jupyter Notebook, Visual Studio Code, Studio, Spyder, Jupyter Lab, QtConsol, etc. It also gives support to manage environments and Conda channel packages without using the command line.

Jupyter Notebook: For the initial learning part of the work, the interactive programming interface Jupyter Notebook is used, offering the possibility of bringing together text, images, mathematical formulas (in latex format) and executable code. It is initially developed for Julia, Python and R programming languages, hence the name Jupyter. Cells are the basic elements of Jupiter notebook; they can contain computer code or text in markdown format. Jupyter notebook allows for efficient testing since we can run the cells individually, and so we don't have to run the whole program every time a change is made. It is beneficial when performing machine learning since we don't have to run the pre-processing every time. Also, notebook provides visual outputs so that we can see graphs or tables directly. It is a full web interface that opens directly in an existing browser on your machine allowing you to execute Python code and facilitate its formatting and understanding. It comes with a kernel or IPython kernel, however there are currently several other kernels like Python 3 used in our case.

TensorFlow: TensorFlow is an open-source framework released by Google Brain (now Google AI) in 2015 to make it easier for developers to build, train, and work with deep learning models and data to make different kinds of predictions through a complete and flexible ecosystem of tools, libraries and community resources that it has. Initially it was a Framework developed for numerical calculations but it has steadily grown in popularity to be one of the most used in the development of machine learning, neural networks and Deep Learning. One of its main features is that common operations on neural networks are done through multidimensional data tables called Tensors (Tensor), hence the name TensorFlow. These tensors also include scalars, vectors, or n-dimensional matrices representing all data types.

Keras: Keras is a high-level deep learning API written in python capable of running on top of TensorFlow or Theano for neural networks. It supports multiple neural network calculations and makes it easy to implement. It was developed as part of the ONEIROS project research effort, and its main author and maintainer is François Chollet, a Google engineer. It runs smoothly on both CPU and GPU. Keras supports almost all neural network models and being modular in nature, it is incredibly expressive, flexible and apt for innovative research. Keras is an open-source API and wraps efficient numerical computation libraries Theano and TensorFlow and it allows to define and train a neural network in few lines of code. It minimizes user actions required for common use cases and provides clear feedback in case of user error, this makes Keras easy to learn and use.

Libraries: In Anaconda we have several libraries that facilitate data processing and the implementation of AI models. In our case we use the following libraries:

- **NumPy and Pandas** when pre-processing the dataset. NumPy allows manipulation of multidimensional arrays and can perform fast mathematical operations on them. Pandas provides us with the data frame structure, allowing us to easily convert to and from CSV files. Both NumPy and Pandas work efficiently on large data.
- For the learning phase of the project, we have the **Scikit-learn library**, which is an open-source library that works with NumPy and Pandas. It offers a wide range of tools and is simple to use, providing everything needed for learning in this project. It was

also used for the standardization of training and test data with the StandardScaler function.

– Also, the **matplotlib library** is used to plot graphs and visualize data with its sub-libraries Pyplot which creates an interface close to the commercial software MATLAB which contains functions very similar to this and seaborn for data visualization, specialized in statistical analysis.

4.5.2. Dataset 1: UNSW-NB15

UNSW-NB15 [20-25] is a dataset created by ACCS (Australian Center for Cyber Security) to evaluate NIDs, as older datasets like KDDCUP and NSLKDD did not reflect good realistic output performance due to the presence of redundancies and missing records but also an imbalance between normal and abnormal data. Thus, a test bed was developed (Fig. 4.1).

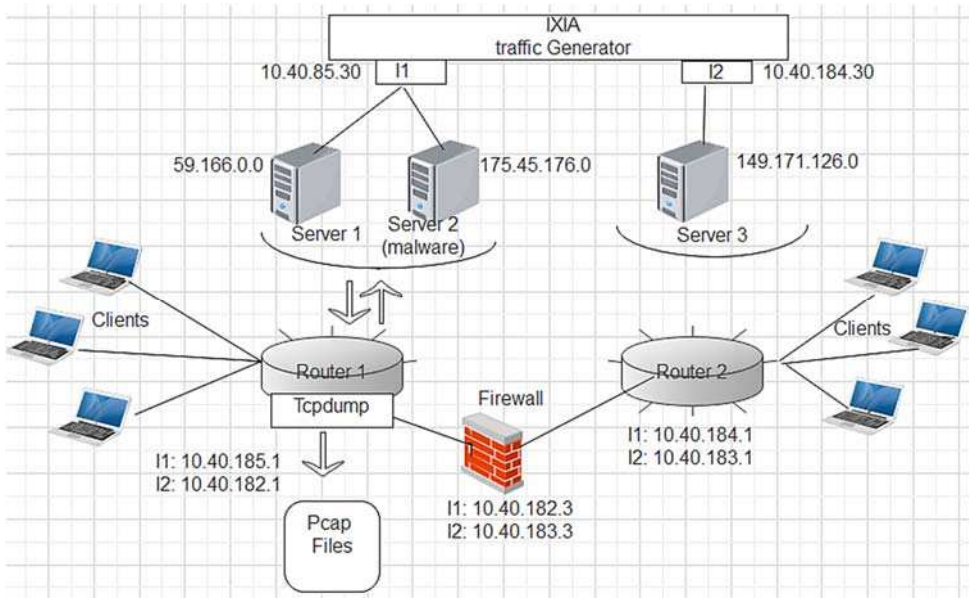


Fig. 4.1. Testbed of the UNSNB15 dataset.

The testbed has an IXIA traffic generator that is configured with the three virtual servers. Servers 1 and 3 are configured for normal traffic propagation while Server 2 shapes abnormal/malicious activity in network traffic. Establishing the intercommunication between servers, and acquiring traffic from public and private networks, there are two virtual interfaces having IP addresses 10.40.85.30 and 10.40.184.30. The servers are connected to the hosts through two routers. Router 1 has IP addresses 10.40.85.1 and 10.40.182.1, while Router 2 is configured with 10.40.184.1 and 10.40.183.1. These routers are connected to the firewall which is configured to pass all normal and abnormal traffic. The tcpdump tool is installed on Router 1 to capture the simulation uptime pcap

files. Moreover, the central intention of this whole testbed was to capture normal or abnormal traffic, which originated from the IXIA tool and was dispersed between network nodes (e.g., servers and clients). Once the pcap files are generated using the tcpdump tool, the characteristics of the UNSW-NB15 dataset are extracted using tools installed and configured on Linux Ubuntu 14.0.4 which are:

- Argus processes raw network packets (pcap files) and generates network flow packet attributes/characteristics;
- Bro-IDS is an open-source network traffic analyzer. It is primarily a security monitor that inspects all network traffic for malicious activity. The Bro-IDS tool is configured to generate three log files from the pcap files;
- Twelve algorithms are developed in C programming language to analyze the flow in depth.

To stimulate the network, the IXIA PerfectStorm3 tool is used in the ACCS Cyber Range Lab to create a hybrid of network traffic with normal and abnormal flows. Anomalous traffic through the IXIA tool simulates nine attack families listed in Table 4.1. The IXIA tool contains all information about new attacks which is continuously updated from a CVE4 site which is a dictionary publicly known information security vulnerabilities and exposures.

Table 4.1. Attacks in the dataset UNSW-NB15.

Type	Fuzzing	Analyze	Backdoors	DoS	Exploits	Generic	Reconnaissance	Shellcode	Worms
Number	2 1218 761	2677	2329	16353	44525	215481	13987	151	174

This 100 GB data set is collected in 2 stages: over 15 hours on 02/17/1015 for 50 GB with 10 attacks per second, and over another period of 16 hours on 02/22/2015 for the remaining 50 GB with one attack per second. The main characteristics of the UNSW-NB15 dataset are a hybrid of real modern normal behaviors and synthetic attack activities. These data include 1400002 flows with 49 characteristics described in Table 4.2 including 9 qualitative and 40 quantitative with a percentage of 95 % normal data and 5 % abnormal. Finally, the output files of the two different tools, Argus and Bro-IDS are stored in the SQL Server 20088 database to match the features generated by Argus and Bro-IDS using stream features such as source and destination IP addresses which are unique, the source and destination port numbers, the transaction protocol. For the characteristics used, the IXIA tool generated an attack data report. This report is configured in the form of the ground truth table to match all transaction records. This table consists of eleven attributes, for example: start time, last time, attack category, attack

subcategory, protocol, source address, source port, destination address, destination port, attack name and attack reference. This dataset is labeled as shown in the table of characteristics, the attack categories (i.e., attack_cat) and the label for each record i.e., 0 if the record is normal and 1 if the record is an attack.

Table 4.2. Description of network traffic characteristics from dataset UNSW-NB15.

Features	Description
scrip	source IP address
sport	source port
dstip	destination IP address
dport	destination port
proto	transaction protocol
dur	total recording time
sbytes	number of bytes from source to destination
dbytes	number of bytes from destination to source
sttl	lifetime of stream from source to destination
dttl	lifetime of the stream from destination to source
sloss	retransmitted or dropped source packets
dloss	destination packets retransmitted or dropped
service	application service (ftp, http, dns, etc.)
sload	source bits per second
dload	destination bits per second
spkts	number of packets from source to destination
dpkts	number of packets from destination to source
swin	source tcp window warning
dwin	destination tcp window warning
stcpb	source tcp sequence number
dtcpb	destination tcp sequence number
smeansz	average of the size of the stream packets transmitted by the source
dmeansz	average size of flow packets transmitted by the destination
smeansz	average of the size of the stream packets transmitted by the source
trans_deph	http request/response transaction connection depth
res_bdy_len	the size of the content of the data transferred from the http service of the server.
sjit	source jitter (msec)
djit	destination jitter (msec)
stime	recording start time
ltime	t last time recording
sintpkt	source inter-packet arrival time (msec)
tcprtt	sum of synack and ackda of tcp packets
synack	time syn and syn_ack of tcp packages
ackdat	time syn_ack and ack of tcp packets
is_sm_ips_ports	if the source is equal to the destination the ip addresses and the port numbers are equal, this variable takes the value 1 otherwise 0
ct_state_ttl	for each state according to specific values for source/destination lifetime
ct_flw_http_mthd	number of streams that have methods like get and post in http service
is_ftp_login	if the ftp session is accessible by user and password, then 1 else 0.
ct_ftp_cmd	number of streams having a command in ftp session
ct_srv_scr	number of connections that contain the same service and source address out of 100 connections based on last seen.
ct_srv_dst	number of connections containing the same service and destination address out of 100 connections according to the last hour

Table 4.2. Continued.

Features	Description
ct_dst_ltm	number of connections from the same destination address out of 100 connections according to the last time
ct_src_ltm	number of connections from the same source address out of 100 connections according to the last time
ct_src_dport_ltm	number of connections from same source address and destination port out of 100 connections according to last time
ct_dst_sport_ltm	number of connections from the same destination address and source port out of 100 connections according to the last time
ct_dst_src_ltm	number of connections from same source and destination address in 100 connections according to last time.
state	The state and its dependent protocol
attack_cat	the name of each attack category. in this data set, nine categories
label	0 for normal and 1 for recorded attack
dintpkt	destination inter-packet arrival time (msec)

4.5.3. Dataset 2: IoT Network Intrusion Dataset

For the case study, we used a second public dataset called IoT Network Intrusion Dataset [26,27] developed by Huy Kang Kim, whose main authors are Hyunjae Kang, Dong Hyun Ahn, Gyung Min Lee, Jeong Do Yoo, Kyung Hose Garer and Huy Kang Kim. The last Dataset update took place on Friday, September 27, 2019 at 04:57. Revealed data of a smart home system over a period of 3 months 21 days from May 20, 2019 to September 10, 2019, consisting of two smart home devices such as SKT NUGU (NU 100) and EZVIZ Wi-Fi Camera (C2C Mini O Plus 1080P).

SKT NUGU (NU 100) is a smart speaker developed in Korea, which has a line-out terminal and can output sound when connected to an external device. It also runs on a built-in battery rather than a wired battery. It offers 30 features such as music appreciation, smart home, schedule management, weather alert, commerce, IPTV, traffic information and order delivery as well as some financial services and movie information.

The EZVIZ Wi-Fi Camera (C2C Mini O Plus 1080P) is a device that makes it easy to monitor all rooms in a house by providing discreet and effective security. This camera is equipped with a motion detector that will detect any abnormal movement. Once connected to a Wi-Fi network, we can ensure the security of the house by watching HD video at any time from a smartphone, tablet or computer and receive alert messages there via the dedicated application.

Both of these IoT devices are used victim devices and are wired to a smart home Wi-Fi router. Laptops, tablets and smartphones are also connected to the smart home router. These devices were used as attack devices in the testbed (Fig. 4.2). The dataset consists of 42 raw network packet files (pcap) at different times. Packet files are requested using the monitor mode of the wireless network adapter. Wireless headers are removed by Aircracking. All attacks except the Mirai Botnet category are packets called during attack simulation using tools such as Nmap. In the case of the Mirai Botnet category, the attack

packets were generated on a laptop computer and then manipulated to make it appear as if they came from the IoT device. We have four categories of attacks, which are further divided into eight sub-categories. Fig. 4.3 shows the taxonomy of attacks.

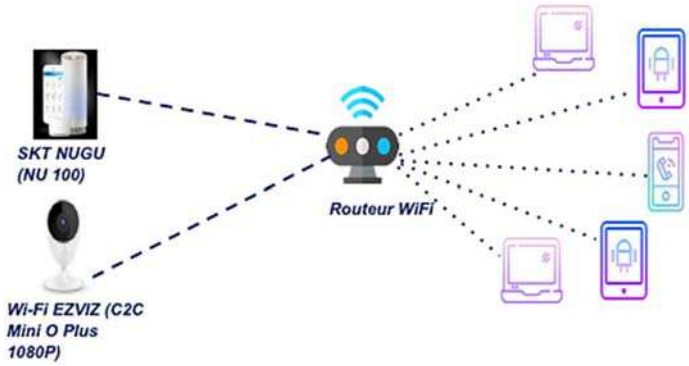


Fig. 4.2. Architecture of the IoT Network Intrusion dataset.

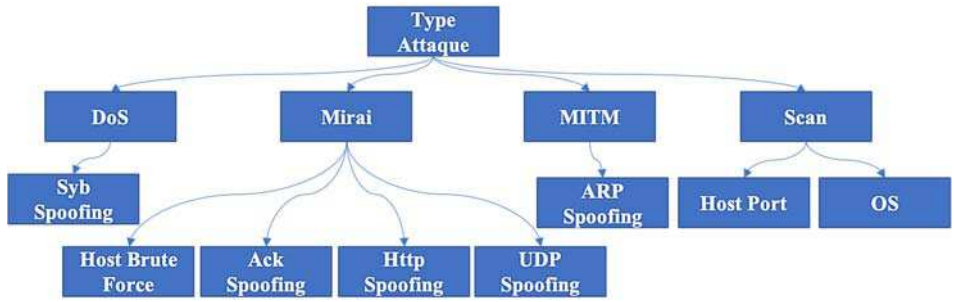


Fig. 4.3. Type of attacks in the IoT Network Intrusion dataset.

The main motivation for using two different datasets is to see if the performance of our models depends on the nature of the data. This is why we use a dataset with more anomalies (IoT Network intrusion) and another with more normal traffic (UNSW-NB15).

4.5.4. Method

For our anomaly detection and attack identification work in the IoT, we have adopted the following approach:

- First, we implement several selected anomaly detection models related to smart home type IoT applications;
- Then the best of the models is validated on another IoT application domain, here a generic IoT network;
- After that we implement several attack identifications models that will be used to identify the type of attacks once the anomaly detection phase is over.

The following Fig. 4.4 shows the architecture of our proposal.

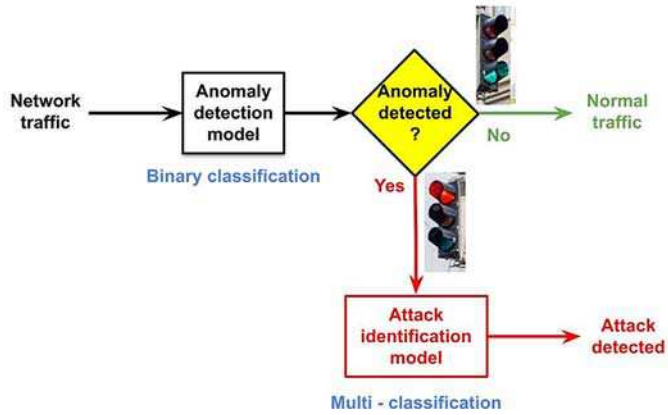


Fig. 4.4. Anomaly detection and attack identification.

4.5.5. Data Pre-processing

To understand the network traffic data we have, we are going to do an analysis both in form and in substance, that is to say, do a set of manipulation and visualization with statistical tools and thus make a description and a more in-depth statistical analysis.

4.5.5.1. Shape Analysis

For anomaly detection, we started by locating the "target variable" which represents our output, specified here as "label" (in both datasets), in other words it is a variable whose values must be modeled and predicted by the other variables.

The dataset is spread over 625783 rows and 86 columns for the IoT Network dataset, and over 1400002 rows and 49 columns for the UNSW-NB15 dataset. The columns represent network traffic characteristics and their descriptions are given in Tables 4.2 and 4.3 respectively for the UNSW-NB15 dataset and the IoT Network dataset.

The analysis of the nature of the data shows that the IoT Network dataset contains two categories of variables including 45 qualitative and 41 quantitative variables while the UNSW-NB15 dataset contains 9 qualitative and 40 quantitative variables. In addition, these variables are either real values, integers or of type objects. To finish with the shape analysis, we looked for missing data and the results show that there is no missing data for the IoT Network dataset (Fig. 4.5). The UNSW-NB15 dataset contains some missing data (Fig. 4.6) in three columns: ct_flw_http_mthd, is_ftp_login, attack_cat. The ct_flw_http_mthd and is_ftp_login features are not part of the features present in the two datasets, so we have removed them. The missing data for the attack_cat column represents normal flows, so we replaced it with another category that we created to represent the absence of an attack.

Table 4.3. Description of network traffic characteristics from IoT Network dataset.

Features	Description
Flow_ID	feed identifier
Src_IP	source IP address
Src_Port	source port number
Dst_IP	destination IP address
Dst_Port	destination port number
Protocol	protocol used
Timestamp	timestamp
Flow_Duration	flow duration
Tot_Fwd_Pkts	total number of forward packets
Tot_Bwd_Pkts	total number of packets transmitted in the reverse direction
TotLen_Fwd_Pkts	the total length of forward packets
TotLen_Bwd_Pkts	the total length of packets transmitted in the reverse direction
Fwd_Pkt_Len_Max	the maximum length of direct transmission packets
Fwd_Pkt_Len_Min	minimum forward packet length
Fwd_Pkt_Len_Mean	the average length of direct transmission packets
Fwd_Pkt_Len_Std	standard deviation of forward packet length
Bwd_Pkt_Len_Max	the maximum length of packets transmitted in the reverse direction
Bwd_Pkt_Len_Min	the minimum length of packets transmitted in the reverse direction
Bwd_Pkt_Len_Mean	the average length of packets transmitted in the reverse direction
Bwd_Pkt_Len_Std	standard deviation of the length of packets transmitted in the opposite direction
Flow_Byts/s	stream in bytes per second
Flow_Pkts/s	the flow in packets per second
Flow_IAT_Mean	average time between packet arrivals
Flow_IAT_Std	standard deviation of packet inter-arrival time
Flow_IAT_Max	maximum time between packet arrivals
Flow_IAT_Min	minimum time between packet arrivals
Fwd_IAT_Tot	total time between packet arrivals (forward direction)
Fwd_IAT_Mean	mean time between packet arrivals (forward direction)
Fwd_IAT_Std	Standard deviation of packet inter-arrival time (forward direction)
Fwd_IAT_Max	maximum time between packet arrivals (forward direction)
Fwd_IAT_Min	minimum time between packet arrivals l(forward direction)
Bwd_IAT_Tot	Total time between packet arrivals (reverse direction)
Bwd_IAT_Mean	mean time between packet arrivals (reverse direction)
Bwd_IAT_Std	packet inter-arrival time standard deviation (reverse direction)
Bwd_IAT_Max	maximum time between packet arrivals (reverse direction)
Bwd_IAT_Min	minimum time between packet arrivals (reverse direction)
Fwd_PSH_Flags	number of PSH (push) indicators (forward direction)
Bwd_PSH_Flags	number of PSH (push) flags (reverse direction)
Fwd_URG_Flags	number of URG (urgent) indicators (forward direction)
Bwd_URG_Flags	number of URG (urgent) flags (reverse direction)
Fwd_Header_Len	header length (forward direction)
Bwd_Header_Len	header length (reverse direction)
Fwd_Pkts/s	number of packets transmitted per second
Bwd_Pkts/s	number of packets transmitted per second (reverse direction)
Pkt_Len_Min	minimum bundle length
Pkt_Len_Max	maximum packet length

Table 4.3. Continued.

Features	Description
Pkt_Len_Mean	average bundle length
Pkt_Len_Std	standard deviation of the length of a packet
Pkt_Len_Var	packet length variation
FIN_Flag_Cnt	number of END fields of the connection
SYN_Flag_Cnt	number of fields for synchronization
RST_Flag_Cnt	number of fields for reset
PSH_Flag_Cnt	number of push fields
ACK_Flag_Cnt	number of acquisition fields
URG_Flag_Cnt	number of emergency fields
CWE_Flag_Count	number of fields
ECE_Flag_Cnt	number of fields
Down/Up_Ratio	rate of packets delivered in both directions
Pkt_Size_Avg	the average packet size
Fwd_Seg_Size_Avg	the average packet segment size (forward)
Bwd_Seg_Size_Avg	the average size of a packet segment (reverse direction)
Fwd_Byts/b_Avg	the average of transmitted bytes (forward direction)
Fwd_Pkts/b_Avg	the average of the transmitted packets (forward direction)
Fwd_Blk_Rate_Avg	the average blocking rate in the forward direction
Bwd_Byts/b_Avg	the average of bytes transmitted (reverse direction)
Bwd_Pkts/b_Avg	the average of the transmitted packets (reverse direction)
Bwd_Blk_Rate_Avg	the average blocking rate in transmission in the reverse direction
Subflow_Fwd_Pkts	the number of packets transmitted in the forward direction
Subflow_Fwd_Byts	the number of bytes transmitted in the forward direction
Subflow_Bwd_Pkts	the number of packets transmitted in the opposite direction
Subflow_Bwd_Byts	the number of bytes transmitted in reverse
Init_Fwd_Win_Byts	Number of bytes sent in the initial (direct) window
Init_Bwd_Win_Byts	Number of bytes sent in the initial window (inerse direction)
Fwd_Act_Data_Pkts	active data packets that are transmitted (direct)
Fwd_Seg_Size_Min	the minimum size of a paquer segment (forward direction)
Active_Mean	Average length of time a stream was active before becoming inactive
Active_Std	standard deviation of the time a stream was active before becoming inactive
Active_Max	maximum duration of activity of a flow before becoming inactive
Active_Min	minimum duration of activity of a flow before becoming inactive
Idle_Mean	average length of time a flow was active before becoming active
Idle_Std	standard deviation of the time a flow was inactive before becoming active
Idle_Max	maximum duration of activity of a flow before becoming active
Idle_Min	minimum duration of activity of a flow before becoming active
Label	label
Cat	category
Sub-Cat	sub-category

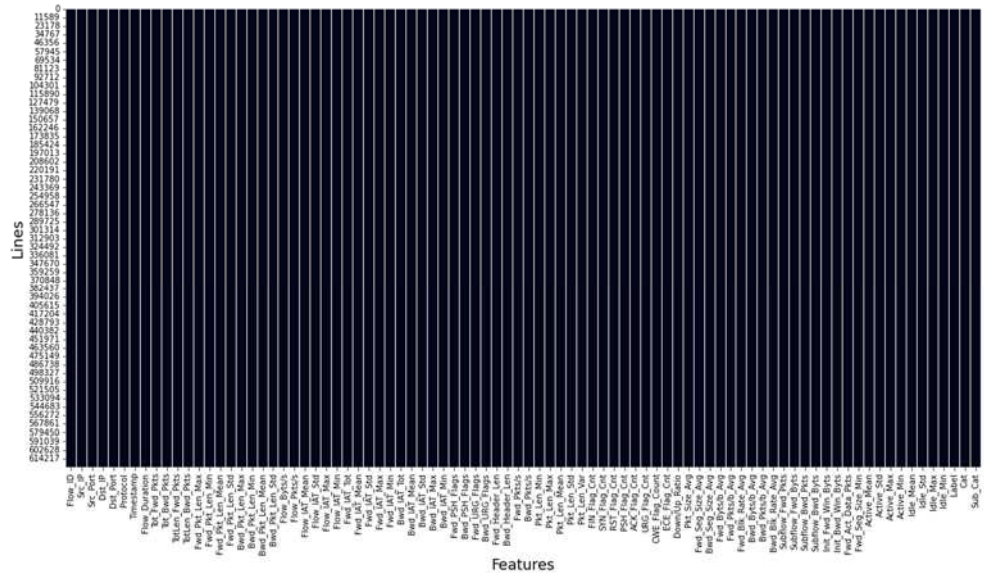


Fig. 4.5. IoT network Dataset missing data analysis.

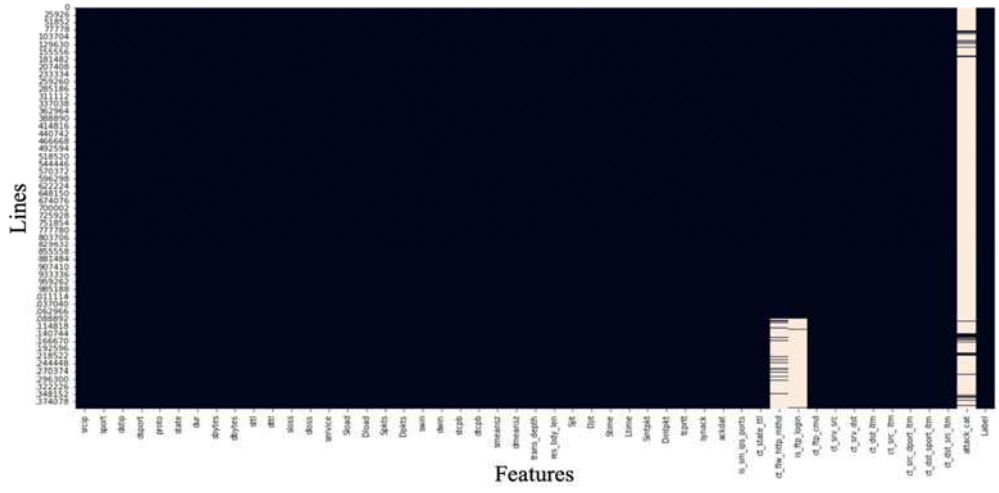


Fig. 4.6. Analysis of missing data from UNSW-NB15.

4.5.5.2. Background Analysis

For the background analysis we start by visualizing the “target variable” and the results show us that:

- The IoT Network dataset is made up of 583790 anomalies or 93 % of network traffic and 40073 normal data or 7 % of network traffic. This denotes more presence of anomaly than normal traffic which is normal since the data was collected from victim devices;

– UNSW-NB15 dataset is made up of 1325038 normal traffic or 95 % and 74964 of abnormal data or 5 % of network traffic.

Then, for the IoT Network dataset we found that some characteristics are represented by their total, average, maximum, minimum and standard deviation value for each flow. Thus, we considered that only the averages are sufficient to represent the value of these characteristics and consequently we removed the others. From this same observation, the stream identifiers, the source and destination addresses, and the timestamp are deleted because they have nothing to do with our output (Label). However, we keep the categories and subcategories of attacks for the detection and identification of the attack type part.

The IoT Network dataset analysis also shows that the distribution of flags compared to anomalies and normal data does not have a very big difference (Fig. 4.7), the trend is the same with the other flags, it is consistent on all flows so we can remove them as they have no influence on our output. As for the non-flags, the analysis of the distribution (density) shows that many of the remaining characteristics have an influence on the output (Figs. 4.8 and 4.9).

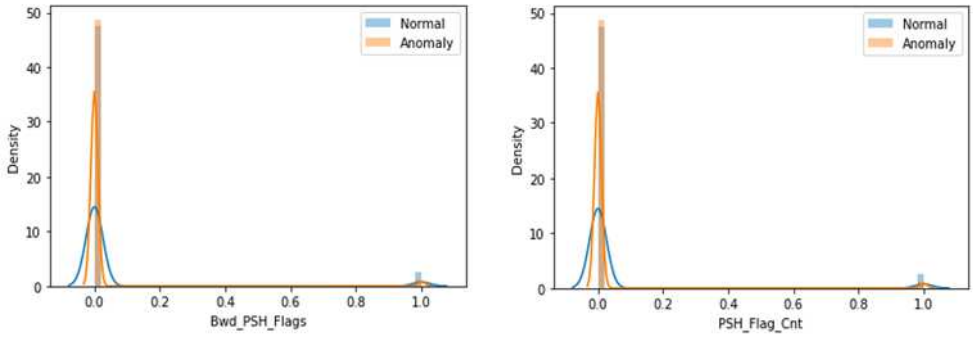


Fig. 4.7. IoT Network dataset: Distribution by transmission of PSH flags (left) and in relation to the number of PSH fields (right).

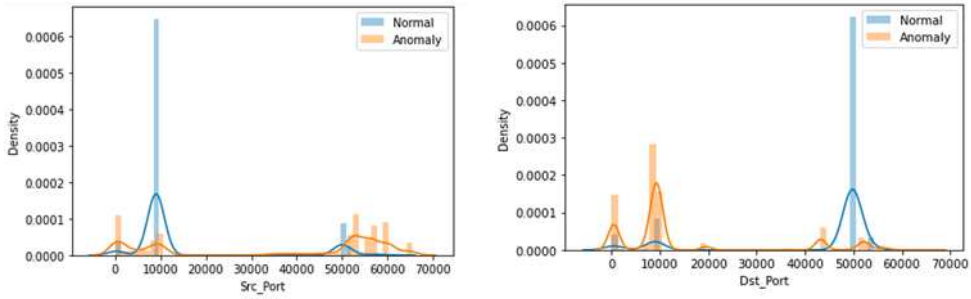


Fig. 4.8. IoT Network dataset: Distribution in relation to source (left) and destination (right) ports.

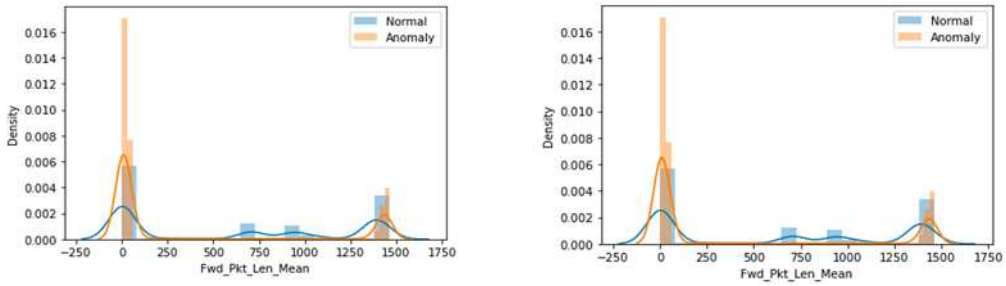


Fig. 4.9. IoT Network dataset: Distribution relative to the average length of transmitted (left) and received (right) packets.

The UNSW-NB15 dataset is already clean, so there is no need to work on removing some features.

Finally, we also examine the correlation that may exist between the features for which we have the characteristics on the abscissa and he ordinate (Figs. 4.10 and 4.11). This analysis shows that there are few characteristics that have a high correlation, among which most are only moderately correlated.

After this first analysis which offers a good understanding of the data, we perform a pre-processing to prepare the data for the application of the models.

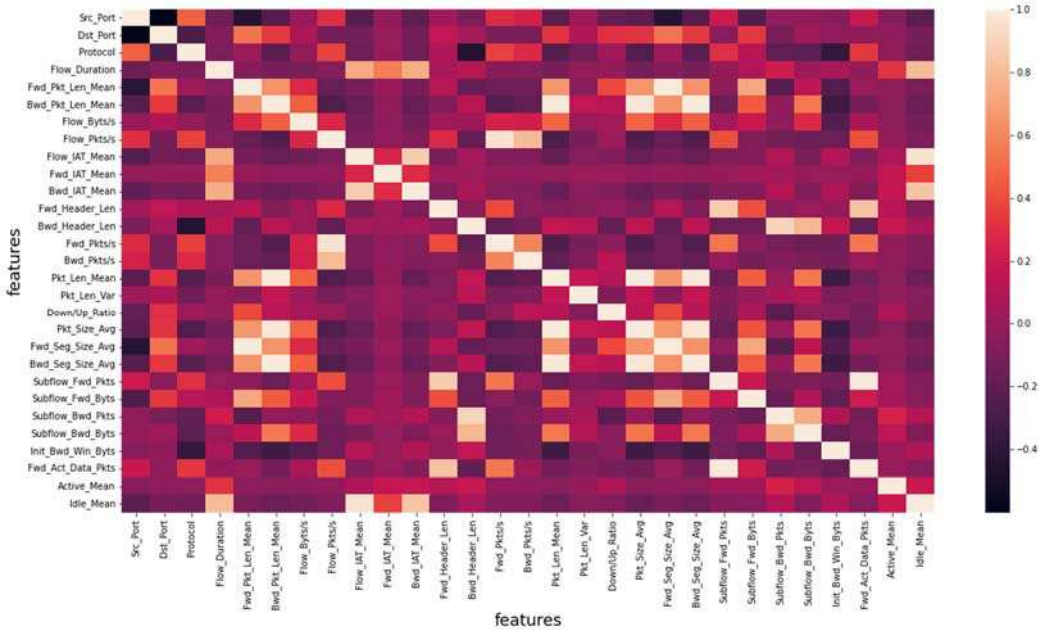


Fig. 4.10. Result of correlation between IoT Network dataset characteristics.

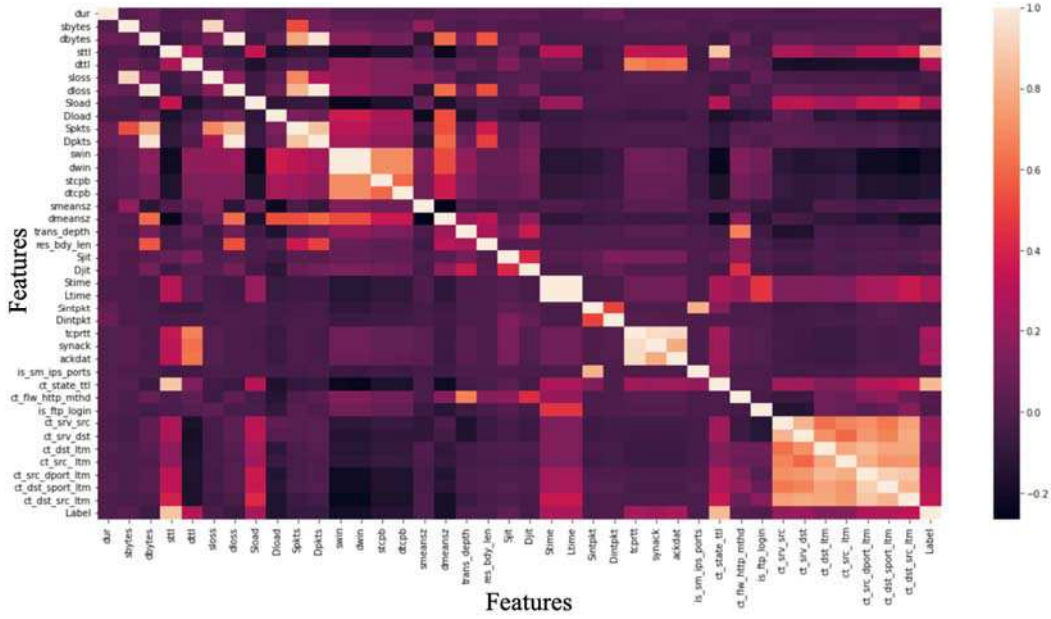


Fig. 4.11. Result of correlation between characteristics UNSW-NB15 dataset.

4.5.5.3. Data Preparation

As we are doing work comparing models on two datasets drawn from two different application domains, we proceed to work on a selection of network characteristics that appear in the two datasets. The following Table 4.4 shows the selected characteristics and their correspondences.

Table 4.4. Selected Features.

Features		Description
UNSW-NB15	**IoT Network**	
sport	Src_Port	source port
dsport	Dst_Port	destination port
proto	Protocol	stream transaction protocol
sbytes	Subflow_Fwd_Byts	number of bytes transmitted from source
dbytes	Subflow_Bwd_Byts	number of bytes transmitted from destination
spkts	Subflow_Fwd_Pkts	number of packets transmitted from the source
dpkts	Subflow_Bwd_Pkts	number of packets transmitted from the destination
smeansz	Fwd_Pkts/b_Avg	the average of the packets transmitted from the source
dmeansz	Dwd_Pkts/b_Avg	the average of the packets transmitted from the destination
dur	Flow_Duration	stream recording time
Sintpkt	Fwd_IAT_Std	Standard deviation of packet inter-arrival time from source
Dintpkt	Dwd_IAT_Std	Standard deviation of packet inter-arrival time from destination

We then do some pre-processing work on the two datasets with the selected features. The pre-processing consists of separating the data for the training phase and the test phase thanks to the train_test_split method of the Sklearn module. The training data represents 80 % of the whole and the rest reserved for the test. After that we separate the data in order to facilitate convergence. We thus obtain the following data: x_train, y_ train, for learning and x_test, y_test for tests. The size of these data is visualized in Table 4.5. Then an encoding is performed on the label with 0 which represents anomaly and 1 normal. Now the data is ready to test the ML models.

Table 4.5 below shows the data distribution of our output over the two datasets after separating them into training and test data.

Table 4.5. Distribution of output to training and test data.

	Label	Training data	Testing data
IoT Network	Anomaly	468618	117092
	Normal	32008	8065
UNSW-NB15	Anomaly	28191	7214
	Normal	59971	14993

4.6. Results and Discussion

In this part, we first present the evaluation metrics before giving the results of our various experiments. Our models based on Decision-Tree, Random Forest, Adaboost, KNN and neural networks are evaluated in both datasets and each time with both training and validation data.

4.6.1. Evaluation Metrics

In this section, we will first present the metrics and parameters of the confusion matrix that will allow us to evaluate our different models.

4.6.1.1. Confusion Matrix

The performance of a Machine Learning algorithm is mainly based on the ability to effectively predict the expected results. A confusion matrix or contingency table is a table that visualizes the performance of a tested classification algorithm against the actual classification. Indeed, it represents the comparison of the real values and the distribution by class of the correct and incorrect predictions. Each column of the table contains a class predicted by the algorithm and the rows of the actual classes. In addition, it allows us to know the errors made and especially their type.

The confusion matrix is composed of 4 terminological values which are:

– **TP (True Positives)**: result where the prediction and the actual value are positive. It corresponds here to the value where the anomalies are correctly predicted;

- **TN (True Negatives)**: result where the prediction and the actual value are negative. This is equivalent to the case where the normal data is correctly predicted;
- **FP (False Positives)**: result where the prediction is positive and the actual value is negative. This is the case where normal data is classified as anomalous;
- **FN (False Negatives)**: result where the prediction is negative and the actual value is positive. It defines here the case where the anomalies are classified as normal.

4.6.1.2. Metrics

The confusion matrix allows us to calculate indicators of performance measures called metrics:

Precision: percentage of correct predictions among positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

Recall/Sensitivity (Recall): percentage of true positives correctly identified.

$$Recall = \frac{TP}{TP + FN}$$

Accuracy: percentage of correct predictions on all the data.

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

F1Score: percentage representing the harmonic mean of the precision and recall values. It provides us with a balance between these two values to better evaluate the performance of our classification.

$$F1Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

4.6.2. Analysis and Interpretation of Results

Now, we proceed to the analysis and evaluation of the results, starting first with the problem of anomaly detection before focusing more particularly on the identification of attacks in the IoT network.

4.6.2.1. Anomaly Detection

We modeled the data using Machine Learning classification models. First, to adjust our pre-processing we chose a basic model, the Decision-Tree. Then, after adjusting our Dataset, we tested other classification models such as Random Forest and AdaBoost. All these models are tested with our two datasets from different application domains.

4.6.2.1.1. Evaluation of Decision-Tree

Table 4.6 below shows the result of our basic Decision-Tree model.

The confusion matrix in the IoT Network dataset (Table 4.6) shows that out of 117092 anomalies 117064 are indeed detected as anomalies and only 28 as normal. In the same way, the model produces normal traffic, i.e., out of 8065 normal traffic, 8019 are detected as normal and only 46 as anomalies. The same trend also emerges with the UNSW-NB15 dataset. Thus, its confusion matrix shows that out of 15083 anomalies, 14343 are indeed detected as anomalies and 740 as normal. In the same way, the model produces normal traffic much better because out of 264,916 normal traffic, 264,257 are detected as normal and therefore 659 as anomalies.

Table 4.6. Modeling results with Decision-Tree on IoT network and UNSW_NB datasets.

	Precision	Recall	F1_score	Support	Confusion Matrix
IoT Network					[[117064, 28], [46, 8019]]
0	1.00	1.00	1.00	117092	
1	1.00	0.99	1.00	8065	
Accuracy			1.00	125157	
Macro avg	1.00	1.00	1.00	125157	
Weighted avg	1.00	1.00	1.00	125157	
UNSW-NB15					[[117064, 740], [659, 264257]]
0	0.96	0.95	0.95	15083	
1	1.00	1.00	1.00	264916	
Accuracy			1.00	279999	
Macro avg	0.98	0.97	0.98	279999	
Weighted avg	0.99	1.00	0.99	279999	

The results seem to show perfect prediction with 100 % accuracy and 100 % accuracy with the IoT Network dataset. For the UNSW-NB15 dataset, the model offers a slightly lower accuracy of 98 % but the performance difference remains relatively small. It provides good results regardless of the field of application.

However, you have to make sure that the model has not overfit and see how it will behave with new data that it has not seen during the learning phase. For this we also conducted the training with validation data. The curve below (Fig. 4.12) shows the trend of the score with the training data and with the validation data on the two datasets.

There is a gap between the validation data and the training data, especially with the IoT.

Network dataset, which means that an improvement can always be made. Thus, to do this we can further adjust our pre-processing. For this we looked at what are the most important features in the prediction with this basic model. Figs. 4.13 and 4.14 show, for each dataset, the level of importance in the prediction of each feature.

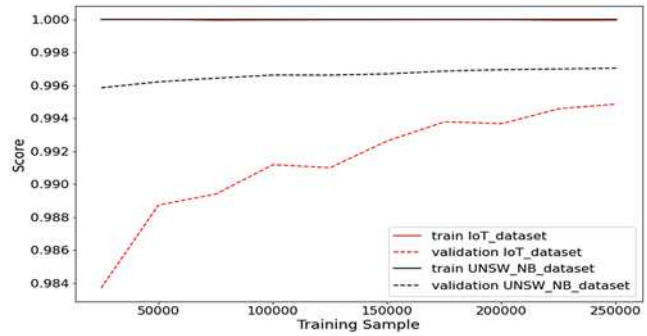


Fig. 4.12. Modeling results with Decision-Tree on the two datasets.

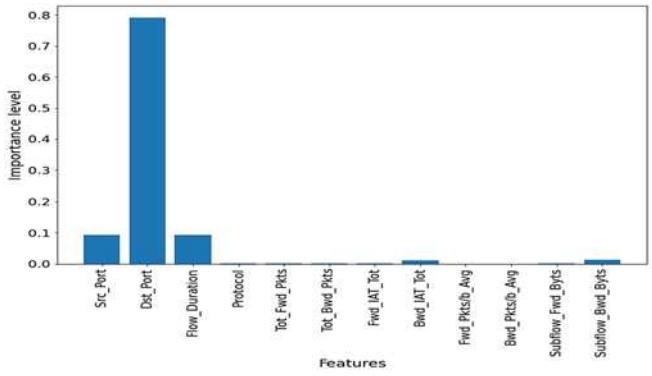


Fig. 4.13. Evaluation of important characteristics used by Decision-Tree (IoT Network).

The first histogram (Fig. 4.13) shows us that the most important characteristics of the IoT Network dataset are the source addresses, destination addresses, the flow duration and others with a slightly low rate. This means that we can still adjust and delete the less important features and test the models with the most important features. In the same way, with the UNSW-NB15 dataset (Fig. 4.14), the most important features are the average of the packets (from the source and the destination), the source and destination ports, the size in byte of the source and destination packets and the dwell time (inter-arrival of packets from source and destination).

4.6.2.1.2. Evaluation of Random-Forest and Adaboost Models

Several classification models were tested: Random Forest, SVM, KNN, and Adaboost. However, Random Forest and AdaBoost offer the best results. We, therefore visualize and analyze these two models.

To model these two models, we used the 12 characteristics common to the two datasets. The results are shown in Tables 4.7 (Random Forest) and 4.8 (Adaboost).

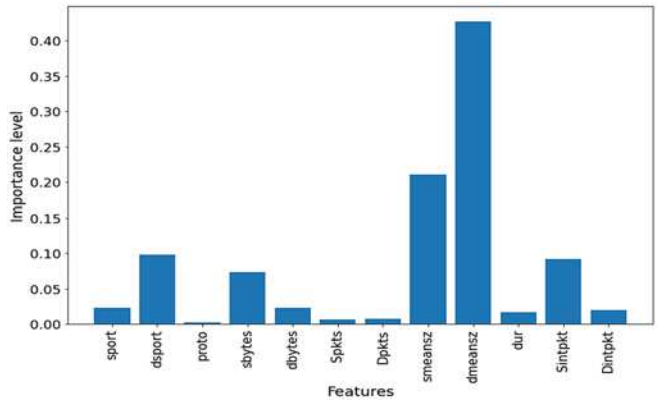


Fig. 4.14. Evaluation of important characteristics used by Decision-Tree (UNSW_NB).

Table 4.7. Modeling results with Random-Forest.

	Precision	Recall	F1_score	Support	Confusion matrix
IoT Network					[[117081, 11], [73, 7992]]
0	1.00	1.00	1.00	117092	
1	1.00	0.99	0.99	8065	
Accuracy			1.00	125157	
Macro avg	1.00	1.00	1.00	125157	
Weighted avg	1.00	1.00	1.00	125157	
UNSWNB-15					[[14513, 570], [508, 264408]]
0	0.97	0.96	0.96	15083	
1	1.00	1.00	1.00	264916	
accuracy			1.00	279999	
Macro avg	0.98	0.98	0.98	279999	
Weighted avg	1.00	1.00	1.00	279999	

Table 4.8. Modeling results with Adaboost.

	Precision	Recall	F1_score	Support	Confusion Matrix
IoT Network					[[117080, 12], [230, 7835]]
0	1.00	1.00	1.00	117092	
1	1.00	0.97	0.98	8065	
Accuracy			1.00	125157	
Macro avg	1.00	0.99	0.99	125157	
Weighted avg	1.00	1.00	1.00	125157	
UNSW-NB15					[[12737, 2346], [1361, 263555]]
0	0.90	0.84	0.87	15083	
1	0.99	0.99	0.99	264916	
Accuracy			0.99	279999	
Macro avg	0.95	0.92	0.93	279999	
Weighted avg	0.99	0.99	0.99	279999	

According to the results observed in the tables above, the Random Forest offers more performance on both datasets (f1_score of 100 % IoT Network and 98 % UNSW-NB15) compared to AdaBoost (f1_score of 99 % IoT Network and 93 % UNSW-NB15).

However, to be sure of the results we looked at their behavior with respect to the new data by visualizing the scores according to the training and validation samples. Thus, the results show that AdaBoost is preferable since it behaves better with new data (Fig. 4.15). For the Random Forest, the validation curve is far from that of learning, especially for the IoT Network (in red), which indicates a tendency of overfitting (Fig. 4.16).

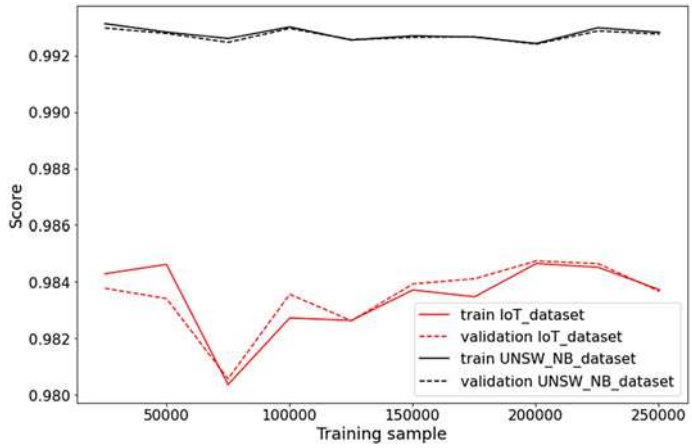


Fig. 4.15. Modeling results with Adaboost on the two datasets.

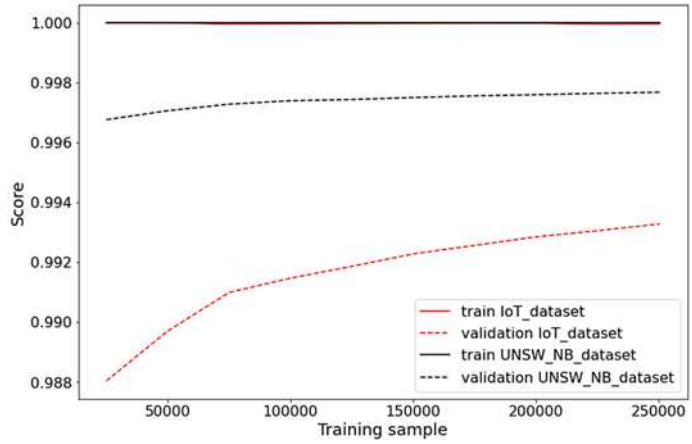


Fig. 4.16. Modeling results with Random-Forest on the two datasets.

In the following, we will develop a neural network model and compare its performance with the best model we have in the first step, i.e., the AdaBoost.

4.6.2.1.3. Evaluation of Results with Neural Networks

The neural network we designed is a classification neural network which is composed of 4 layers with 2 hidden layers, an input layer of 12 features and an output layer. On the output layer we used Sigmoid as an activation function and ReLu for the other layers. Table 4.9 gives the summary of the different layers. After each layer we added a layer of Dropout.

Table. 4.9. Summary of the composition of the neural network model on the two datasets.

Model: «sequential_1»		
Layer (type)	**Output Shape**	**Param #**
dense_3 (Dense)	(None, 20)	260
dropout_2 (Dropout)	(None, 20)	0
dense_4 (Dense)	(None, 10)	210
dropout_3 (Dropout)	(None, 10)	0
dense_5 (Dense)	(None, 1)	11
Total params: 481 Trainable params: 481 Non-trainable params: 0		

We run the model over 100 epochs and visualize the error and accuracy on both the training data and the validation data. For both datasets, the result shows that the model does not overfit and starts to have stable values from the 30th epoch. The best compromise of the model that we can thus have from this epoch.

This model offers very good accuracy (Fig. 4.17): IoT Network ~93 % and UNSW-NB15 ~94 %. Which is more than enough for satisfactory anomaly detection. However, according to the previous results, the AdaBoost offers the best performance with 99 % for IoT Network and 93 % for UNSW-NB15 without overfitting.

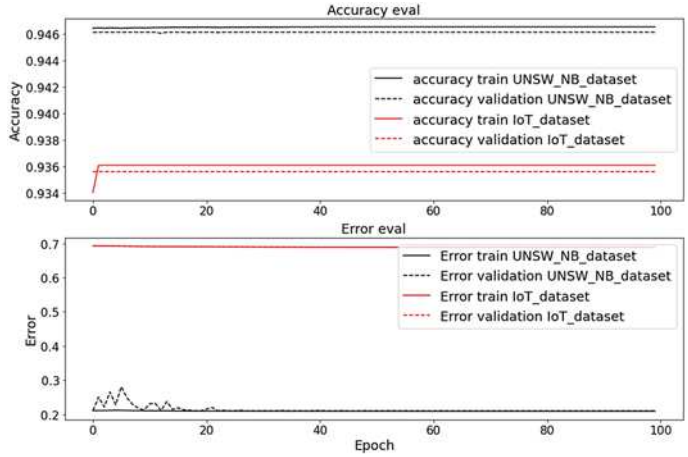


Fig. 4.17. Neural network performance against training and validation data.

Tables 4.10 and 4.11 present the summary of the performances of the models that we tested on respectively the IoT Network dataset and the UNSW-NB15 dataset for the detection of anomalies.

Table 4.10. Anomaly detection model performance summary (IoT Network).

Models	Precision	Accuracy	Risk of Overfitting
Decision-Tree	100	100	Medium
Random Forest	100	100	Medium
AdaBoost	100	100	Very low
Neural Network		99	Very low

Table 4.11. Anomaly Detection Model Performance Summary (UNSW_NB).

Models	Precision	Accuracy	Risk of Overfitting
Decision-Tree	98	100	Medium
Random Forest	98	100	Medium
AdaBoost	95	99	Very low
Neural Network		93	Very low

4.6.2.2. Identification of Attacks

In this part we will adopt the same approach as in the previous section but this time, instead of simply detecting the presence or absence of anomalies, we will identify the type of attack by carrying out a multiclass classification.

We will also work on the same fields of application as before. We will use the same datasets but we will change the output variable which is now the attack category whose encodings in numerical values are given in Tables 4.12 and 4.13. The latter has 4 classes for the IoT Network and 9 classes for the UNSW-NB15.

As in the previous section, we will use the Decision Tree model as a baseline to do the pre-processing and adjust our dataset.

Table 4.12. Encode of the IoT Dataset output variable.

Attack	Dos	MITM ARP Spoofing	Mira	Scan
Code	0	1	2	3

Table 4.13. Encode of the UNSW_NB Dataset output variable.

Attack	Analysis	Backdoors	DoS	Exploits	Fuzzing	Generic	Reconnaissance	Shellcode	worms
Code	0	1	2	3	4	5	6	7	8

4.6.2.2.1. Identification of Attacks by a Model Based on Decision-Tree

The results seem to show near perfect prediction for the IoT Network dataset with 99 % accuracy and 100 % accuracy while with the UNSW-NB15 dataset they show 66 % accuracy and 82 % accuracy. The UNSW-NB15 dataset offers less performance, which can be explained by the fact that it contains more attack type than the IoT Network dataset. Indeed, the more type of attack there is, the greater the probability of having attacks with similar signatures. It is a phenomenon that we observe on the confusion matrix that Analysis, Backdoors and DoS attacks are classified at more than 60 % as being Exploits. This means that the model never classifies normal traffic as an attack, which is a very good thing. This can be explained by the fact that these attacks have almost the same signatures, they all exploit a fault in the network.

Table 4.14 below shows the result of our basic model, Decision-Tree.

To confirm these results of our baseline, we will apply the random forest and the KNN as before. To be sure that the results of this baseline are solid, we will examine if he did not overfit. Fig. 4.18 shows that for the Decision Tree, the validation curve is a little distant compared to that of the learning especially for the UNSW-NB15 (in black), however the trend shows the curves evolve while approaching.

Table 4.14. Results of the multiclass classification with Decision-Tree on the two datasets.

	Precision	Recall	F1_score	Support	Confusion Matrix
IoT Network					[[11841, 3, 2, 2], [1, 6953, 89, 31], [1, 88, 83180, 38], [1, 53, 42, 14817]]
0	1.00	1.00	1.00	11848	
1	0.98	0.98	0.98	7074	
2	1.00	1.00	1.00	83307	
3	1.00	0.99	0.99	14913	
Accuracy			1.00	117142	
Macro avg	0.99	0.99	0.99	117142	
Weighted avg	1.00	1.00	1.00	117142	
UNSW-NB15					[[18, 1, 22, 172, 22, 1, 0, 0, 0], [1, 14, 26, 115, 25, 0, 0, 1, 0], [6, 0, 245, 867, 35, 10, 5, 2, 5], [21, 4, 355, 2604, 87, 54, 71, 9, 9], [7, 2, 23, 277, 1611, 3, 0, 0, 0], [0, 1, 41, 151, 2, 7017, 1, 1, 0], [0, 3, 23, 153, 4, 1, 735, 0, 0], [0, 1, 1, 12, 0, 4, 0, 99, 0], [0, 0, 0, 2, 0, 0, 1, 0, 10]]
0	0.34	0.08	0.12	236	
1	0.54	0.08	0.13	182	
2	0.33	0.21	0.26	1175	
3	0.60	0.81	0.69	3214	
4	0.90	0.84	0.87	1923	
5	0.99	0.97	0.98	7214	
6	0.90	0.80	0.85	919	
7	0.88	0.85	0.86	117	
8	0.42	0.77	0.54	13	
Accuracy			0.82	14993	
Macro avg	0.66	0.60	0.59	14993	
Weighted avg	0.82	0.82	0.81	14993	

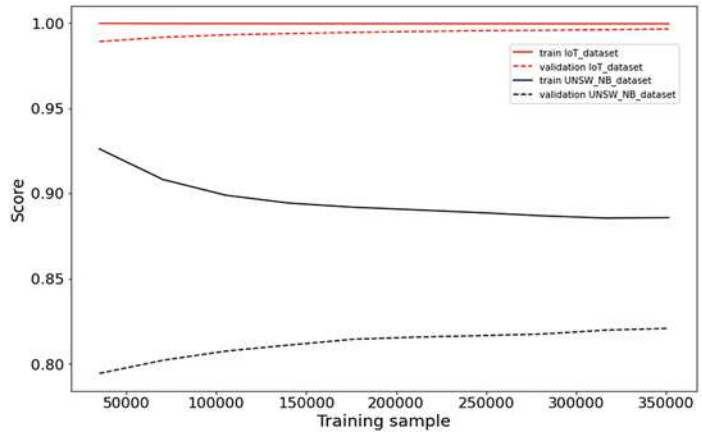


Fig. 4.18. Test and validation with the Decision-Tree on the two datasets.

4.6.2.2.2. Random Forest and KNN

Table 4.15 below shows the result of our basic model, Decision-Tree.

Table 4.15. Results of multiclass classification with Random Forest on IoT network and UNSW_NB datasets.

	Precision	Recall	F1_score	Support	Confusion Matrix
IoT Network					[[11727, 4, 7, 4, 0], [0, 6625, 396, 10, 47], [1, 237, 82929, 9, 3], [3, 20, 53, 7986, 2], [0, 81, 226, 2, 14949]]
0	1.00	1.00	1.00	117942	
1	0.95	0.94	0.94	7078	
2	0.99	1.00	0.99	83244	
3	1.00	0.99	0.99	8065	
4	0.99	0.98	0.99	15028	
Accuracy			1.00	125157	
Macro avg	0.99	0.98	0.98	125157	
Weighted avg	1.00	1.00	1.00	125157	
UNSW-NB15					[[9, 0, 16, 136, 40, 0, 0, 0, 0, 18], [4, 11, 10, 105, 31, 0, 0, 3, 0, 2], [3, 3, 146, 954, 40, 8, 6, 5, 0, 12], [6, 9, 148, 2931, 122, 26, 66, 13, 1, 56], [4, 6, 17, 25, 1155, 1, 0, 0, 1, 518], [0, 2, 13, 148, 2, 6938, 1, 1, 1, 13], [0, 1, 1, 5, 149, 4, 0, 779, 0, 0, 3], [0, 0, 2, 8, 0, 1, 0, 92, 0, 4], [0, 0, 0, 4, 0, 0, 0, 0, 6, 0], [0, 2, 3, 40, 413, 4, 1, 2, 2, 0, 264440]]
0	0.35	0.04	0.07	219	
1	0.32	0.07	0.11	166	
2	0.40	0.12	0.19	1177	
3	0.62	0.87	0.72	3378	
4	0.64	0.59	0.61	1958	
5	0.99	0.97	0.98	7119	
6	0.90	0.82	0.86	951	
7	0.79	0.88	0.83	105	
8	0.67	0.60	0.63	10	
9	1.00	1.00	1.00	164916	
Accuracy			0.99	279999	
Macro avg	0.67	0.60	0.60	279999	
Weighted avg	0.99	0.99	0.99	279999	

According to the results observed in the tables above, the Random Forest offers better performance on both datasets (99 % IoT Network and 67 % UNSW-NB15 accuracy, 100 % IoT Network and 99 % UNSW-NB15 accuracy) compared to KNN (accuracy of 63 % IoT Network and 58 % UNSW-NB15, accuracy of 80 % IoT Network and 82 % UNSW-NB15).

To confirm these results and ensure that the models do not overfit, we also look at their behavior concerning new data by submitting validation data to them. Figs. 4.19 and 4.20 show the result of this analysis for Random Forest and KNN (Table 4.16) respectively.

Thus, the results show that, on the two datasets, the Random Forest like the KNN does not seem to overfit. However, as it is the Random Forest that gives the best performance, we choose it as the model to be compared with that of the neural network.

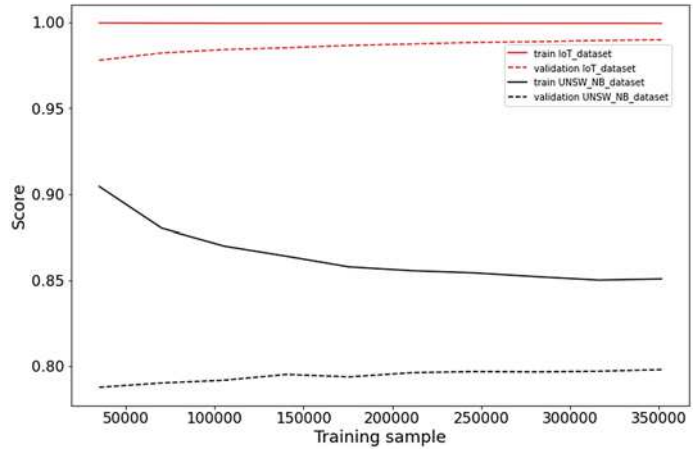


Fig. 4.19. Modeling results with Random-Forest on the two datasets.

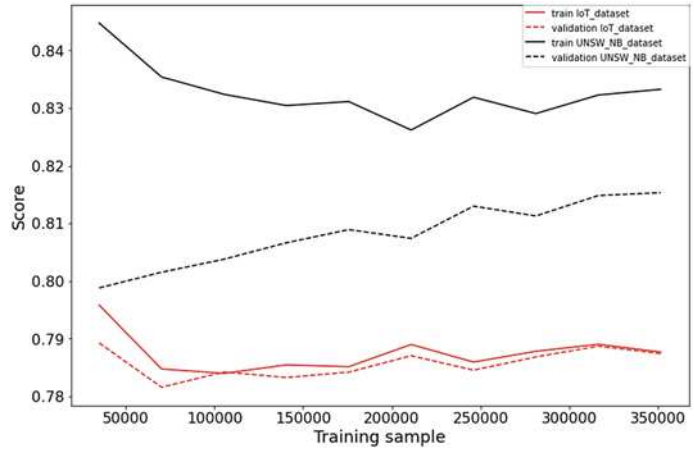


Fig. 4.20. Modeling results with KNN on the two datasets.

Table 4.16. Results of multiclass classification with KNN on IoT network and UNSW_NB datasets.

	Precision	Recall	F1_score	Support	Confusion Matrix
IoT Network					[[11825, 0, 23, 0], [0, 1812, 4308, 954], [0, 2168, 76595, 4544], [0, 1134, 10827, 2952]]
0	1.00	1.00	1.00	11848	
1	0.35	0.26	0.30	7074	
2	0.83	0.92	0.88	83307	
3	0.35	0.20	0.25	14913	
Accuracy			0.80	117142	
Macro avg	0.63	0.59	0.61	117142	
Weighted avg	0.76	0.80	0.77	117142	
UNSW-NB15					[[20, 1, 52, 161, 2, 0, 0, 0, 0], [4, 5, 33, 131, 2, 1, 2, 4, 0], [30, 15, 247, 817, 24, 27, 11, 3, 1], [64, 20, 338, 2578, 79, 51, 61, 20, 3], [9, 2, 68, 253, 1582, 7, 2, 0, 0], [6, 2, 37, 178, 14, 6973, 1, 3, 0], [2, 0, 32, 143, 5, 3, 734, 0, 0], [0, 3, 3, 31, 1, 1, 0, 78, 0], [0, 0, 0, 5, 1, 1, 1, 0, 5]]
0	0.15	0.08	0.11	236	
1	0.10	0.03	0.04	182	
2	0.30	0.21	0.25	1175	
3	0.60	0.80	0.69	3214	
4	0.93	0.82	0.87	1923	
5	0.99	0.97	0.98	7214	
6	0.90	0.80	0.85	919	
7	0.72	0.67	0.69	117	
8	0.56	0.38	0.45	13	
Accuracy			0.82	14993	
Macro avg	0.58	0.53	0.55	14993	
Weighted avg	0.81	0.82	0.81	14993	

4.6.2.2.3. Identification of Attacks by a Model Based on Neural Network

We have designed two multiclass classification neural network models that we will compare with the Random Forest respectively for the IoT Network dataset and the UNSW-NB15 dataset. The following tables (Tables 4.17, 4.18) summarize the composition in terms of the layers of the two models. In both models, we used Softmax as the activation function at the output layer and ReLu for the other layers.

Fig. 4.21 shows the result of the two models for the two datasets (IoT Network and UNSW-NB15) representing the accuracy and the error on the training and validation data over 100 epochs.

Table 4.17. Composition of the neural network model for UNSW-NB15 dataset.

Model: «sequential»		
Layer (type)	**Output Shape**	**Param #**
dense (Dense)	(None, 100)	1300
dense_1 (Dense)	(None, 24)	2424
dense_2 (Dense)	(None, 9)	225
Total params: 3,949 Trainable params: 3,949 Non-trainable params: 0		

Table 4.18. Composition of the neural network model for IoT Network dataset.

Model: «sequential_1»		
Layer (type)	**Output Shape**	**Param #**
dense_3 (Dense)	(None, 100)	1300
dense_4 (Dense)	(None, 24)	2424
dense_5 (Dense)	(None, 4)	100
Total params: 3,824 Trainable params: 3,824 Non-trainable params: 0		

The results show very good performance with an accuracy of 92 % for the IoT Network dataset and 73 % for the UNSW-NB15 dataset between the epochs where the models begin to be stable, i.e., between 70 and 85. Moreover, for both models the validation and training curves are almost superimposed which means that they behave well with new data. Compared to the Random Forest, which we had as the best model with 100 % IoT Network and 99 % UNSW-NB15, this one still has the best performance.

The two following tables (Tables 4.19, 4.20) give the summary of the performance of the models that we tested on the two datasets for the identification of attacks.

4.7. Conclusion

In this chapter, we have detailed, analyzed and discussed the implementation of several AI classification models for anomaly detection and attack identification in IoT networks. We have used two different datasets to assess the performance of our models: a dataset with more anomalies than normal traffic (IoT Network intrusion) and another one with more normal traffic than anomalies (UNSW-NB15).

For anomaly detection, four binary classification models were studied: a decision tree, random forest, AdaBoost and an artificial neural network. AdaBoost is chosen as a model because it offers better performance in both application domains even though the neural

network offers very good results and could also be used instead of AdaBoost. Then, we used a multiclass classification for the identification of attacks with Decision Tree, Random Forest, KNN and Artificial Neural Network models. In the latter case, Random Forest offers the best performance with an accuracy of 100 % for the IoT Network dataset and 99 % for the UNSW-NB15 dataset.

In perspective, these models could be improved so that they produce better performance by adjusting their hyperparameters. It's interesting but imperative work to find the optimal hyperparameters for each model so that it provides the best results. This work requires a lot of time and resources in terms of calculation, especially for voluminous data.

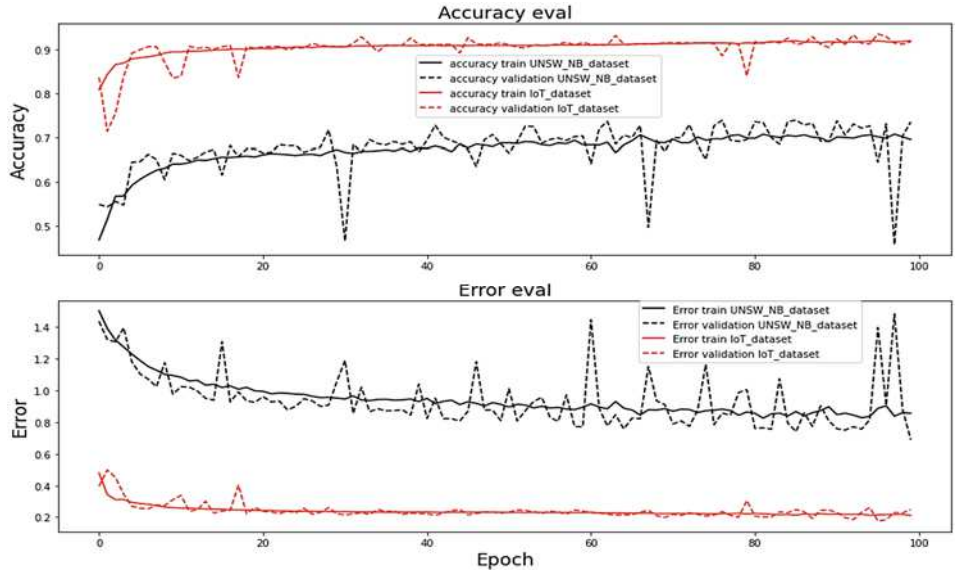


Fig. 4.21. Performance of neural networks against training and validation data.

Table 4.19. Summary of attack identification model performance (IoT Network).

Models	Precision	Accuracy	Risk of Overfitting
Decision-Tree	99	100	Medium
Random Forest	99	100	Medium
KNN	63	80	Medium
Neural Network		92	Very low

Table 4.20. Summary of attack identification model performance (UNSW_NB).

Models	Precision	Accuracy	Risk of Overfitting
Decision-Tree	66	100	Medium
Random Forest	67	99	Medium
KNN	58	82	Medium
Neural network		73	Very low

It is important to stabilize the result that we already have on the most important characteristics that denote the most impacted network parameters in their normal (usual) values by an attack. The identification of these characteristics makes it possible to know which parameters must be monitored in particular for an abnormal event when it occurs.

References

[1]. C. Diallo, IoT Anomaly Detection and Attack Identification Using Smart Traffic Classification Techniques, *2022 7th IEEE International Conference on Frontiers of Signal Processing (ICFSP)*, 2022, pp. 51-58, doi: 10.1109/ICFSP55781.2022.9924901.

[2]. C. Huang, L. Zheng, S. Wang, V. C. M. Leung, T. Lin, K. Peng, Intrusion detection system based on decision tree over big data in fog environment, *Wireless Communications and Mobile Computing*, Vol. 2018, 2018, 4680867.

[3]. Y. Lei, Network anomaly traffic detection algorithm based on SVM, in *Proceedings of the International Conference on Robots and Intelligent System (ICRIS'17)*, 2017, pp. 217-220.

[4]. C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks, *IEEE Access*, Vol. 5, 2017, pp. 21954-21961.

[5]. R. S. Naoum, N. A. Abid, Z. N. Al-Sultani, An enhanced resilient backpropagation artificial neural network for intrusion detection system, *IJCSNS International Journal of Computer Science and Network Security*, Vol. 13, Issue 3, 2013, pp. 98-104.

[6]. Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu, Denial-of-service attack detection based on multivariate correlation analysis, in Neural Information Processing (B.-L. Lu, L. Zhang, J. Kwok, Eds.), *Springer*, Berlin, Heidelberg, 2011, pp. 756-765.

[7]. S. Sahu, B. M. Mehtre, Network intrusion detection system using J48 Decision Tree, in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI'15)*, 2015, pp. 2023-2026.

[8]. M. A. Ambusaidi, X. He, P. Nanda, Z. Tan, Building an intrusion detection system using a filter-based feature selection algorithm, *IEEE Transactions on Computers*, Vol. 65, Issue 10, 2016, pp. 2986-2998.

[9]. M. Nawir, A. Amir, N. Yaakob, O. B. Lynn, Multiclassification of UNSW-NB15 dataset for network anomaly detection system, *Journal of Theoretical and Applied Information Technology*, Vol. 96, Issue 15, 2005, pp. 5094-5104.

[10]. S. Sahu, B. M. Mehtre, Network intrusion detection system using J48 Decision Tree, in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI'15)*, 2015, pp. 2023-2026.

[11]. S. Omar, A. Ngadi, H. H. Jebur, Machine learning techniques for anomaly detection: An overview, *International Journal of Computer Applications*, Vol. 79, Issue 2, 2013, pp. 33-41.

[12]. M. Ahmed, A. N. Mahmood, J. Hu, A survey of network anomaly detection techniques, *Journal of Network and Computer Applications*, Vol. 60, 2016, pp. 19-31.

[13]. S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. Gomez Chavez, B. Rubinstein, Machine learning in network anomaly detection: A survey, *IEEE Access*, Vol. 9, 2021, pp. 152379-152396.

[14]. M. Sarhan, S. Layeghy, N. Moustafa, M. Portmann, NetFlow datasets for machine learning-based network intrusion detection systems, in *Proceedings of the 10th EAI International Conference on Big Data Technologies and Applications (BDTA'20), and 13th EAI International Conference on Wireless Internet (WiCON'20)*, Virtual Event, December 11, 2020, pp. 117-135.

[15]. V. Garcia-Font, C. Garrigues, H. Rifà-Pous, A comparative study of anomaly detection techniques for smart city wireless sensor networks, *Sensors*, 2016, Vol. 16, Issue 6, 868.

[16]. A. Diallo and C. Diallo, Human Activity Recognition in Smart Home using Deep Learning Models, *2021 IEEE International Conference on Computational Science and Computational Intelligence (CSCI)*, 2021, December 15-17, Las Vegas, USA, pp. 1511-1515.

[17]. C. Diallo, An Intelligent System for Detecting People from Images and Videos Provided by IoT Devices, *2021 IEEE International Conference on Computational Science and Computational Intelligence (CSCI)*, 2021, December 15-17, Las Vegas, USA, pp. 1721-1727, doi: 10.1109/CSCI54926.2021.00326.

[18]. J. C. J. Badji, C. Diallo, A CNN-based Attack Classification versus an AE-based Unsupervised Anomaly Detection for Intrusion Detection Systems, *2022 IEEE International Conference on Electrical, Computer and Energy Technologies (ICECET)*, 2022, July 20-22, Prague, Czech Republic.

[19]. A. Diallo, L. Affognon, C. Diallo and E. C. Ezin, Deep Learning Based Binary and Multi-class Classification Comparison for Anomaly Detection, *Proc. of the 8th IEEE International Conference on Engineering and Emerging Technologies (ICEET 2022)* 27-28 October 2022, Kuala Lumpur, Malaysia.

[20]. M. Nour, and J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), *Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015*.

[21]. M. Nour, and J. Slay, The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset, *Information Security Journal: A Global Perspective (2016): 1-14*.

[22]. M. Nour, et *al.* Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks, *IEEE Transactions on Big Data (2017)*.

[23]. M. Nour, et *al.* Big data analytics for intrusion detection system: statistical decision-making using finite Dirichlet mixture models, *Data Analytics and Decision Support for Cybersecurity. Springer, Cham, 2017. 127-156.*

[24]. M. Sarhan, L. Siamak, M. Nour, and M. Portmann, NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems, *In Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event*, December 11, 2020, Proceedings (p. 117). Springer Nature.

[25]. The UNSW-NB15 Dataset, Intelligent Security Group, UNSW Canberra, Australia. https://research.unsw.edu.au/projects/unsw-nb15-dataset, last visit on August 15, 2022.

[26]. H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, H. K. Kim, September 27, 2019, IoT network intrusion dataset, *IEEE Dataport*, doi: https://dx.doi.org/10.21227/q70p-q449.

[27]. IoT Network Intrusion Dataset, IEEEDataPort. https://ieee-dataport.org/open-access/iot-network-intrusion-dataset, last visit on August 15, 2022.

Advances in Networks, Security and Communications, Volume 3

Sergey Y. Yurish, Editor

Thy 3rd volume of *'Advances in Networks, Security and Communications'* open access Book Series contains 5 chapters written by 7 authors from 3 countries: Austria, India and Senegal. Each of the chapters contains the extended state-of-the art followed by new, obtained by the authors results, unpublished before.

Like the first two volumes of this Book Series, the 3rd volume also has been organized by topics of high interest. Every chapter in the volume is independent and self-contained and has the same structure: first an introduction to specific topic under study; second particular field description including applications, ending by well selected list of references with books, journals, conference proceedings and web sites.

With this unique combination of information in each volume, the in *'Advances in Networks, Security and Communications'* Book Series will be useful for post-graduate students, researchers, engineers and scientist working in the appropriate areas.

ISBN 978-84-09-47239-0



