A Master Thesis

entitled

Ensemble Classifier Design and Performance Evaluation for Intrusion Detection Using UNSW-NB15 Dataset

by

Zeinab Zoghi

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Master of Science Degree in

Engineering: Computer Science

Dr. Gursel Serpen, Committee Chair

Dr. Ahmad Y. Javaid, Committee Member

Dr. Mohammed Niamat, Committee Member

Dr. Richard G. Molyet, Committee Member

Dr. Amanda Bryant-Friedrich, Dean
College of Graduate Studies

The University of Toledo
August 2020

An Abstract of

Ensemble Classifier Design and Performance Evaluation for Intrusion Detection Using UNSW-NB15 Dataset

by

Zeinab Zoghi

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Master of Science Degree in Engineering: Computer Science

The University of Toledo
August 2020

In this study, an Intrusion Detection system (IDS) is designed based on Machine Learning classifiers and its performance is evaluated for the set of attacks entailed in the UNSW-NB15 dataset. This dataset is comprised of 2,540,226 realistic network data instances as well as 49 features. Most studies reported in the literature employ a representative subset of this dataset with predefined training and testing subsets, and containing a total of 257,673 records which this study also used. In light of relatively lower than expected performance of Machine Learning or Statistical classification algorithms tested on this dataset and as reported by others in the literature, this dataset was subjected to visual data analysis to explore potential reasons or issues which likely challenge Machine Learning classifiers. The consequent observations demonstrated the presence of class representation imbalance with respect to pattern counts and class overlap in feature space, which makes preprocessing strategies indispensable before this dataset can be meaningfully employed for data-driven model development for intrusion detection.

For preprocessing, we implemented min-max scaling in the normalization phase followed by the application of Elastic Net and Sequential Feature Selection (SFS) algorithms. We employed ensemble methods using three base classifiers, namely Balanced Bagging, XGBoost, and RF-HDDT, augmented to address the imbalance issue. Parameters of Balanced Bagging and XGBoost are tuned for the imbalanced data, and Random Forest is supplemented by the Hellinger distance metric to address the limitations of default distance metric. Two new algorithms are proposed to address the class overlap issue in the dataset and applied during training. These two algorithms are leveraged to help improve the performance on the testing dataset by affecting the final classification decision made by three base classifiers as part of the ensemble classifier which employsa majority vote combiner.

Performance evaluation of the proposed design for binary and multi-category classification was conducted with respect to standard set of metrics including those derived from the confusion matrix and compared to the performances of other studies reported on the same dataset in the literature. Results demonstrate that the proposed design outperforms those reported in the literature by a significant margin for both binary and multi-category classification case

Acknowledgements

I would like to offer my heartfelt thanks to almighty GOD, the compassionate the merciful, whose presence in my life enables me to battle on despite discouragement occurred in the absence of my parents.

My special appreciation goes to my parents for all their love and support. The words “I Love You” seem too few to demonstrate how I really love them.

Also, I would like to give my deep gratitude to my dear friend, Hossein Abedsoltanm, who has been inspiring and helping me a lot in every aspect of my life.

I would like to extend my sincere gratitude towards my research advisor Dr. Gursel Serpen for imparting his considerable knowledge and experience in this study.

I also want to express my warmest thanks to my committee members, Dr. Ahmad Y.Javaid, Dr. Mohammed Niamat, and Dr. Richard G.Molyet, for accepting my invitation to serve on my thesis committee and enhance the credibility of my work with their valuable comments.

Table of Contents

Abstract iii

Acknowledgements v

Table of Contents vi

List of Tables x

List of Figures xii

Chapter 1 **1**

1 Introduction 1

1.1 Cyber-attacks against computing infrastructure 1

1.2 Cost and consequences of attacks 6

1.3 Evolution of cyber-attacks 10

1.3.1 Efforts to Prevent Attacks 15

1.3.2 Efforts to Detect and Identify Attacks 16

1.4 Types of Intrusion Detection 17

1.4.1 Statistical anomaly intrusion detection system 19

1.4.2 Machine learning intrusion detection system 20

1.5 List of Datasets for IDS development 21

1.5.1 DARPA98 21

1.5.2 KDDCUP 99 22

1.5.3 NSL-KDD 22

1.5.4..... 23

ADFA..... 23

1.5.5 UNSW-NB15 23

Chapter 2 **24**

2 Background 24

2.1 Dataset 24

2.2. Feature Selection Algorithms 26

2.2.1. Wrapper Methods..... 27

2.2.2. Embedded Methods 27

2.3. Normalization Methods 28

2.3.1. Min-Max Scaler 28

2.4. Evaluation Metrics.....29

2.5. Classification Algorithms 32

2.5.1 Random Forest (RF) 32

2.5.2 Balanced Bagging (BB) 33

2.5.3. Extreme Gradient Boosting (XGBoost)..... 34

2.6. Splitting Criterion 35

2.6.1. Hellinger Distance 35

Chapter 3 **37**

3 Analysis of UNSW-NB15 Dataset 37

3.1 Visualization of dataset 37

Chapter 4 **45**

4 Data Preprocessing 45

4.1 Data Cleaning 45

4.2 Data Transformation 46

4.3 Feature Selection 54

Chapter 5 **58**

5 Proposed Methodology 58

5.1 Classifier development methodology 58

5.2 Training methodology 64

5.3 Testing methodology 73

Chapter 6 **77**

6 Simulation Results 77

6.1 Binary Classification 77

6.2 Multiclass Classification 80

Chapter 7 **86**

7 Conclusions & Future Work 86

7.1 Conclusion 86

7.2 Future work 90

References **91**

Appendix A : Python Code for Attribute Processing **113**

Appendix B : Miscellaneous Content and Tables **116**

List of Tables

Table 2.1 Number of records in training and testing subsets for each class 26

Table 4.1 The Mahalanobis distances between the Analysis centroid and the rest 51

Table 4.2 The Mahalanobis distances between the Backdoor centroid and the rest 51

Table 4.3 The Mahalanobis distances between the DoS centroid and the rest 52

Table 4.4 The Mahalanobis distances between the Exploits centroid and the rest 52

Table 4.5 The Mahalanobis distances between the Fuzzers centroid and the rest 52

Table 4.6 The Mahalanobis distances between the Generic centroid and the rest..... 52

Table 4.7 The Mahalanobis distances between the Normal centroid and the rest 52

Table 4.8 The Mahalanobis distances between the Reconnaissance centroid and the rest 53

Table 4.9 The Mahalanobis distances between the Shellcode centroid and the rest..... 53

Table 4.10 The Mahalanobis distance between the Worms centroid and the rest 53

Table 4.11 Accuracy comparison among the performances of six different scalers..... 54

Table 4.12 List of features selected by Elastic Net and SFS..... 56

Table 5.1 Missed alarm rate values for the set of 11 classifiers..... 59

Table 5.2 Entropy and Hellinger distance score measurements..... 61

Table 5.3 First seven rows of a probability score matrix generated by Balanced Bagging classifier..... 65

Table 5.4 (a) Confusion matrix and (b) First two rows of probability scores matrix. 70

Table 5.5 (a) The confusion matrix, (b) Mean and standard deviation arrays through Algorithm #1 and (c) Mean and standard deviation arrays through Algorithm #2..... 71

Table 6.1 Confusion Matrix on test data subset for binary classification..... 77

Table 6.2 Performance evaluation of the proposed model for binary classification..... 78

Table 6.3 Comparison of the proposed classifier design with four others cited (NR indicates Not Reported) 78

Table 6.4 Confusion matrix showing the performance of the ensemble method..... 82

Table 6.5 Confusion matrix showing the performance of the proposed design..... 82

Table 6.6 Performance of the proposed design for multi-class case..... 82

Table 6.7 The accuracy of proposed model vs. the accuracy of different models (NR: Not Reported)..... 84

Table 6.8 The sensitivity of proposed model vs. the sensitivity of different models (NR: Not Reported)..... 85

Table B.1 The values that the parameters of estimators are set by..... 116

Table B.2 Feature subset selected by the different literature..... 117

Table B.3 Different strategies that were utilized in seven literature in comparison with the proposed model 118

List of Figures

Figure 2-1 Confusion matrix and associated performance metric definitions 29

Figure 3-1 Distribution of records across classes 38

Figure 3-2 The visualization of the Shellcode instances using PCA 39

Figure 3-3 The visualization of the Worms instances using PCA 39

Figure 3-4 The visualization of the dataset using T-SNE 40

Figure 3-5 Visualizing the overlapping problem inherent in the dataset using the PCA. 41

Figure 3-6 The visualization of the Normal along with the (a) Exploits, (b) Fuzzers, and (c) Analysis data points 42

Figure 3-7 Mapping the dataset by 10 clusters using k-mean clustering algorithm 43

Figure 4-1 The Euclidean distance of the class centroids when the dataset is not normalized 47

Figure 4-2 The Euclidean distance of the centroids after normalizing the data (upper-left), scaling the dataset with min-max, maxabs (upper-right), robust (left-center), standard (right-center), quantile transformation (lower-left), and power transformation (lower-right) 48

Figure 4-3 The Mahalanobis distance of the centroids when the dataset in not normalized 49

Figure 4-4 The Mahalanobis distance of the centroids after normalizing the data (upper-left), scaling the dataset with min-max, maxabs (upper-right), robust (left-center), standard (right-center), quantile transformation (lower-left), and power transformation (lower-right)..... 50

Figure 4-5 (a) The F1-score of Balanced Bagging classifier across the number of input features. 56

Figure 5-1 An example to illustrate Hellinger distance metric utility 60

Figure 5-2 Classifier development schematic diagram..... 63

Figure 5-3 Illustration of partial output by Algorithm #1..... 68

Figure 5-4 An example to show how Algorithm 1 calculates the prediction error range 73

Figure 6-1 F-measure comparison 85

Chapter 1

1 Introduction

1.1 Cyber-attacks against computing infrastructure

Long before the term ‘technology’ came to prominence, humans invented different methods for long distance communication. In that era, messengers and birds carried the messages to the destination. Since the 17th century, when the semblance of modern technology first appeared, the world has seen the phenomenal growth in communication techniques, modalities and options. Network packets replaced messengers and birds, and cyber technologies contributed to our new way of communication. It is undeniable that we are becoming increasingly dependent on Internet every passing day due to human creativity and innovation. At the same time, the world of cybercrime has been populated with the criminals looking for a perfect crime such as stealing confidential information, data, funds or causing harm to target computing infrastructure. They explore possible avenues through the cyberspace and formulate different strategies called cyberattacks, to gain unauthorized

access to the computing systems. These strategies (aka attacks) may be highlighted as follows:

- Distributed Denial-of-Service (DDoS): A denial-of-service attack restricts the availability of resources by heavily overloading them with tasks often of no value or utility. It leads the victim computer to shut or slow down. In this attack, memory and disk space, connection limits, network resources and critical node capacity [1] are allocated by malicious software designed by the cyber criminals so that no user or client is able to request any services. TCP SYN flood attack and Smurf attack are two common DoS attack types [2-4].
    - Man-In-The-Middle (MITM): The man-in the-middle attack happens when a third party monitors the network traffic between two authorized parties in order to access the utilized services or modify the session contents [5-7]. When the connection is initialized between two parties, a digital certificate is also issued. The certificate comprises both the address of the packet sender and a connection key. Since the certificate is unlocked, the attacker can get access to and modify it. When the recipient receives the certificate with the attacker's address, it will identify the attacker as a secure party. This leads the attacker to behave as either the original sender or intended recipient [8].
    - Password-Based Attacks: Password-based attacks are executed when an attacker acquires the username and password of an authorized user. Based on password cracking techniques, this attack type is categorized into the following popular subtypes [9]:

- Brute Force: In this type of attack, every single possible password patterns are used randomly in order to gain access to the confidential information. These patterns are most probably associated with the user's name, birthday, part of or full address, job title or other similar items.
- Dictionary: In this attack type, password patterns are formed by the words that are commonly used in daily conversation. The name of historical monument can be a good example of such words.
- Shoulder Surfing: This attack happens when an authenticated user enters their password and the attacker traces their fingers movement or listen to the keyboard taps to find or guess the password.

- Phishing: This attack takes place when a victim enters their personal or financial information on a fake website which presents itself as authentic. This attack can also be an email containing a malicious link that redirects a user to the fake website or installs malware or ransomware on the user's computer system [3, 4, 10].
- Malware: The term Malware is short for Malicious Software [4]. Malware denotes any type of software threats against the integrity, confidentiality and availability of data on the victim's computer system. This type of attack is comprised of the following common types:
  - Virus: This type of malware is installed and transmitted from one computer to another through the contaminated emails, websites, CDs or USB drives. Viruses are able to replicate themselves by attaching to a host program and become a part of it. They can be activated and damage or steal the data when

the host program is executed on the victim's computer system [11]. Viruses are categorized into the main classes of encrypted, polymorphic and metamorphic [12].

- Trojan: This kind of malware masquerades as a genuine program while indeed it spies on the user's activities and produces damaging effects on the victim's device [13]. General Trojan and Remote-Access Trojan are the well-known subcategories of Trojans [12].
- Worm: This malware is a peace of code which unlike viruses is able to independently clone itself across the network without explicitly infecting host files. It may cause the DDoS attack by continuously reproducing itself and overloading the servers [13, 14].
- Rootkit: It is an accumulation (a "kit") of tools used by attackers in order to obtain administrative ("root") access to victim's device [14]. These tools cover up the malicious activities and make it difficult to identify the attack [11, 12].
- Ransomware: The objective of this subcategory is to encrypt a piece of data on victim's device or lock the victim out, make the data inaccessible through encryption, and demand a ransom payment in exchange for the decryption key [15, 16].
- Spyware: The aim of this malware is to monitor the victim's activities, collect the sensitive or personal information about the user and their device, transfer the findings to the attacker, and help the attacker to control the

target device remotely without the user's consent [17]. This malware is able to install and execute itself without the user's permission [12].

- Botnet: This attack results in a large number of infected devices that can be remotely managed by the attackers [18]. This malware mostly targets the personal computers, regenerate itself on them, and causes further attacks or collects personal information. It transfers the observations to the bot-master to help them conduct their malicious activities using the user's device without them knowing it [19].
- Key Logger: The key logger is a software installed and executed in the background of the user's computer system stealthily in order to record each keyboard's tap and steal the user's confidential information. This software could be either installed intentionally by the attacker or by the users with no knowledge of its harmful effect [20].
- Adware: This malware interrupts the current activities on user's device by displaying unwanted advertisements [21].

- SQL Injection: This type of attack exploits database-driven websites, injects malicious code to the SQL query in order to bypass the web application authentication and gain access to database and the database server [22].
- Cross-Site Scripting (XSS): This attack is a malicious Java script payload injecting to vulnerable websites. This code is transferred to the browser of the websites' visitors intending to get access to their confidential information such as credit card number [23].

- Eavesdropping: This attack is launched when the attacker intercepts the confidential communication between two legitimate users without their permission. The aim of this attack is to snoop on internet for conversations where the system resources have not been modified [24].

- Social Engineering: Social engineering attacks are carried out by the ones possessing remarkable social skills rather than technical skills. They take advantage of their good interpersonal skills to earn others' trust and get sensitive information to accomplish their malicious purposes [25]. This type of attack is subcategorized to social, technical, socio-technical and physical [26].

The attackers leverage the attacks to get access to system resources and either destroy them or collect valuable information. Based on the objective of carrying out the attacks, they can be categorized into active and passive attacks [27, 28]. The threats by which the attackers inject incorrect information or alter the source of data belong to active attacks. On the contrary, in passive attacks no system resources are modified: the sole aim of attackers in this type of attack is to make use of data without implementing any changes.

Cyberattacks in general, involve an unauthorized access to the confidential information on the victim's device or the device resources. It leads to damaging consequences on either or both of the target users and servers. The negative impacts of cyberattacks are discussed in the next session.

1.2 Cost and consequences of attacks

The attackers target both the large [29] and small [30] businesses, organizations as well as private citizens to carry out the cyberattacks for theft, capture of confidential

information, corporate espionage, system sabotage, money laundering, reputation enhancement, and curiosity [31, 32]. Although a large business or organization is equipped with the sophisticated cybersecurity tools, it still can be a perfect target due to the presence of ample and valuable information for the attackers to benefit from. On the contrary, a small firm or private citizen usually suffers from the lack of cyber security tools. Although, there is not enough information for intruders, the security breaches smooth their path for the further attacks on wider cyberspaces.

To reduce cyber threats or recover from damages caused by cyberattacks, the organizations assess the cyber risks. Any uncertainties related to the data resources and computer devices that threatens the confidentiality, availability and integrity of the information or information systems are identified as cyber risks [33]. Confidentiality represents the system resources protected from unauthorized access. While integrity preserves any piece of information from unauthorized changes, availability guarantees that the authorized users can always gain access to the required data resources [34]. Cybercriminals pose risks on technology assets through violating these three main aspects of information security and cause harmful impacts that are mainly categorized into four types [35-37]:

- Economic: Economic cyber harm represents negative financial consequences affected individuals or the organizations. Customer and revenue reduction, regulatory fine, forensic investigation and remediation costs, and litigation expenditures [38-41] have followed in the wake of cyberattacks and lead the individuals and organizations into the economic harms. This kind of cyber impact is common in terms of evaluation and investigation, however it is not

straightforward to measure the economic impacts particularly the indirect forms of financial loss such as remediation costs. It is estimated that cybercrime carries a global cost of $6 trillion annually by 2021 [42, 43].

- Physical: It reflects any physical damages that harm either or both the technology assets and individuals. This cyber harm is a costly kind. Not only have been the wirelessly connected infrastructures picked for implementing cyberattacks causing physical damages, but also the broken devices lead to deadly consequences for individuals. Wireless electronic medical devices and eldercare robots [44] are good examples of technology assets that can be subjected to cyberattacks. Any disruption in their regular performance will be fatal due to their proximity to humans and their main role in serving them. Adulterated food and water supplies, and air, road and rail accidents [45, 46] represent deadly cyber incidents as well.

- Psychological: This type of cyber harm focuses on individuals and their mental capacity. Feeling of anger, anxiety, shame, depression, confusion and annoyance of being cheated, loss of self-confidence [47, 48], and severe consequences on the attitudes of victimized population [49] are the psychological impacts that may be triggered by implementing cyberattacks.

- Reputational: It indicates the harm that causes loss of trust among the stakeholders of organizations or followers of prominent personalities. The consequences of this sort of cyber impact are generally accompanied with economic threats [50].

- Governmental: It harms the political system or politicians in terms of loss of public trust which is highly associated with reputational cyber harm.

- Cultural: This cyber harm reduces the social reliability and threatens cultural safety [51].

In general, each type of cyber harm is comprised of direct and indirect costs. Direct cost denotes losses or damages that are tangible for victims, while the victims are not able to perceive losses and damages in the context of indirect harm. Money withdrawn from users' bank account and loss of trust in online banking are examples for direct and indirect costs, respectively [52].

According to the study made by the Current Population Survey (CPS) and the American Community Survey (ACS) in 1998, 43 percent of households in the United States had personal computer, while 27 percent of them utilized the internet [53]. There were only 39 reported cyberattacks which were commonly associated with website deformation [54] and the total number of open cases in the United States regarding computer intrusions was 547 [55]. In the broader context, the population of global internet users stood at about 188 million and the total number of websites were 2,410,067 at the time [56]. Only 72 websites were reported for being distorted [55] and the overall cost carried by cybercrime commitments was about $250 million [57].

Unlike 1998 that the most common cyberattack was defacing the websites, today the most commonly reported cyberattack is Ransomware that affected business communities every 14 minutes and cost the world $11.5 billion in 2019 in the internet world with the users population of 4.39 billion [58] and almost 1.5 billion websites [59]. From $11.5 billion general losses, more than $3.5 billion [60] is recorded and issued by FBI's Internet Crime Complaint Center (IC3) from the complaints they received by individual and

business victims in the United States. It is predicted that a Ransomware attack may affect a business every 11 minutes and the concomitant damage reach $20 billion. The total loss caused by cybercrimes are also projected to rise by $6 trillion by 2021 [42, 61].

Since the global internet user population grows rapidly, the number of cyberattacks and the methods that the attackers use to target the victims constantly change for the worse. We will explore the evolution of cyberattacks from 1980s to 2020 in the next section.

1.3 Evolution of cyber-attacks

In 1977, Apple released its first fully assembled personal computer with the introduction of color display which led this young industry for a pending technology revolution [62]. Although the boom in personal computers provoked the hackers' curiosity, it was not the starting point of cybercrimes in its broadest sense. In fact, the hackers' interest had been awaken long ago, when the human society first embraced the technology. In the mid-1960s, John Thomas Draper tricked the call receiver using a whistle played at 2600 Hz and successfully entered the operator mode to place free long-distance call. He obtained the information about the phone technology from Bell telephone employee through the method that is now called social engineering attack [25, 63]. This attack type was also used by Kevin Mitnick in 1976 in order to convince a bus driver to give him the address of the bus company where he claimed that he could purchase a ticket punch for his school project. Soon after, he leveraged the unused slips found in the dumpsters next to the bus company building and got free rides for any bus in the Los Angeles area [64, 65]. Although, committing these cybercrimes facilitated accessing the services without paying a penny, the main aim behind them was basically curiosity about the state-of-the-art technologies.

In 1980s, the growing number of households had easily one personal computer. Falling prices of computer hardware such as floppy disks led to the computer viruses becoming pandemic and turned the phone and bus into computer hackers. In 1981, Rich Skrenta developed Elk Cloner, the first virus targeted the early Apple operating systems [66, 67]. This virus became active every 50^{th} time that a computer booted up with a floppy disk infected with Elk Cloner which caused a short poem to pop up on the monitor [68]. A year later, Fred Cohen coined the term 'Computer Virus' while he was demonstrating a short code stored in a floppy disk and was able to replicate itself by modifying other programs throughout a computer system and spread to other computers [69, 70]. Simultaneously, there was a group of youngsters ranging in age from 16 to 22 years, namely the 414s, who directly connected to the victims' device by means of telnet utility. This group carried out password-based attacks to establish a connection [71]. In 1986, the first IBM PC virus was developed by the Farooq Alvi brothers. The main cause of the virus infection was floppy disks [72, 73]. Two years later, Robert Tappan Morris crated the first known worm which used Distributed Denial of Service (DDoS) techniques [74], and released it from a computer placed in MIT in order to determine the scope or reach of Internet. He made use of known vulnerabilities in UNIX system resources at the time which was the passwords that can be easily guessed [75, 76].

The advent of World Wide Web in early 1990s and the development of the first website [77] in 1991 unleashed an Internet revolution and new historical period had begun in cyberattacks evolution. Money has become a very important matter in cybercrime lifecycle along with curiosity and quenching the thirst in exploring new technologies. Online channels smoothed the way the data traveled from one computer to another and led to an

explosion in the number of cyberattacks. In 1990, the first polymorphic virus was developed by Mark Washburn. This virus could mutate its code as well as its signature once it was duplicated [78, 79]. In 1992, Michelangelo virus was created and released to damage hard drive boot record which activated on the 6th of March every year. It was the first cyberattack that triggered digital mass panic [75]. In 1994, the first bank network attack occurred by Vladimir Levin. He took advantage of the drawbacks of bank networking programs to break into Citibank's network, utilized the username and password belonging to their customers, and transferred $10 million USD to a number of personal international bank accounts [71, 80]. A year later, the first macro virus, called Concept, was made. Macro viruses are the ones that are written in the same macro language that is employed in the applications such as Microsoft Word. This virus affected a computer device and its applications when the infected Word file was open [81]. Chernobyl virus first appeared in 1998. This virus overwritten the systems BIOS and made them unusable [82]. In 1999, Melissa, the fastest replicating virus for its time was released. This virus used the e-mail MAPI address book to transmit itself to the first 50 email addresses in the contact lists. It attached an MS-Word file that carried the virus to a short message [83], infected the computer device once the file was opened, and led to the rise in corporate e-mail traffic [84, 85].

In the first decade of the 3rd millennium, the idea of cloud computing [83, 86] was realized and the pre-cloud era attacks have loomed out of the clouds having gone through mutations. Incorporating internet services in mobile phones removed the time and place limitation upon the internet access and has made it ubiquitous. There is a large portion of data held on cyberspace, inspiring the criminals to capture or damage it. Although

Distributed Denial of Service (DDoS) attacks found their way since the genesis of the internet, they came to prominence on February 8, 2000, when Yahoo, Amazon, Buy, E-bay, and CNN announced that their websites were overloaded with the flood of internet traffic [87]. A couple of months later, a self-spreading worm called I LOVEYOU crawled through the Internet and infected the computing devices using the social engineering strategy. Unlike Melissa which was reliant upon MS Outlook, it was a stand-alone Visual Basic script worm which is capable of self-replication [85, 88]. In 2002, one of the more destructive DDoS attacks happened, targeting 13 Domain Name Service (DNS) root servers and terminating the activities of some of them [89-91]. The fastest ever worm was developed in 2003 infecting 90 percent of vulnerable Microsoft SQL Servers that did not employ Service Pack 3. The worm produced damaging DoS effects slowing down the internet traffic [92]. In 2004, the first mobile phone worm, namely Cabir, was discovered. This virus utilized Bluetooth technology in order to find the first Bluetooth device nearby to propagate itself; otherwise it drained the battery of infected mobile phone away since the scanning process to find a new Bluetooth device continued until the next such device in proximity or the depletion of battery power [93, 94]. A banking Trojan called Zeus was first identified in 2007. At the time, a large percentage of government networks got infected with Zeus [95]. It is a Botnet using form grabbing and key logging method for its propagation [96] and resulted in the theft of tens of millions of dollars from bank accounts during its lifetime [97]. In 2008, the largest identity theft case ever in the United State was reported by the U.S. Department of Justice. More than 40 million credit cards had been stolen from nine major U.S. retailers [98].

Throughout the present decade, the cyber threats have not only targeted the individuals but also increasingly breached to businesses and governments. From the earliest known incident of network intrusions taking place back in 1988 to the present, the cyber threats have been evolved from a single hacking event committing by amateur hackers and youngsters to more sophisticated cyber-attacks carried out by professional hackers and cyber pirates. In 2010, the first digital weapon known as Stuxnet affected numerous centrifuges at the Natanz uranium enrichment facility in Iran. Stuxnet utilized two stolen digital certificates to seem legitimate software to Microsoft Windows [99, 100]. A year later, DigiNotar's system was deceived into issuing hundreds of fraudulent digital certificates for highly trafficked websites such as Google. This led the users to trust the apparently secure web pages to enter their usernames and passwords in [101] and let the intruders leverage their account information. In 2012, thousands of windows-based workstations in Saudi Aramco were affected by a self-replicating virus named Shamoon. Once a computer in a network is infected, this virus replicates itself in the corresponding network, overwrites master boot record and makes it unusable, and eventually sends the report to the attacker confirming that the disk in completely wiped up [102]. In 2013, South Korea reported a cyber-attack known as Dark Seoul, affected 48,000 computers belonging to the number of South Korean banks and media companies. This virus erased several hard drives and caused denial of service [103]. One of the popular France TV channel, namely TV5 Monde, was hacked in 2015. This attack caused 18 hours interruption in signal transmission [104]. In 2017, Notpetya, the most destructive cyberweapon ever in history, occurred targeting Ukraine. This malware was made up of two cyber threats, namely EternalBlue and Mimikatz. Using the methodology adopted by EternalBlue and Mimikatz,

NotPetya was able to achieve a fully remote control over any unpatched computer systems and then infect other patched machines by extracting the passwords from their memory holding the password [105]. The recently developed cyber-attacks are mostly intending to be used for cyber warfare in comparison with the earliest ones. In fact, improving cybersecurity can be an effective remedy to prevent these attacks. In the next section, we will discuss a variety of cybersecurity strategies designed and employed since the history of cyber threats started.

1.3.1 Efforts to Prevent Attacks

Considering the threat landscape has rapidly evolved, cybersecurity has become loomed large to modern societies. The most basic strategies for securing computer systems merged since the earliest days of computing, when the internet had not become public yet. User authentication has been the first required strategy for providing secured access. It was first provided on time-sharing UNIX systems back in the early 1960s [106]. The first known antivirus, namely Reaper, was designed in the early 1970s to remove Creeper [107] which was made by Bob Thomas. In the 1990s, with the sharp growth of the World Wide Web and subsequently, a rise in cybercrimes, firewalls, and anti-viruses became a priority for every organization and rapidly found its way into many personal computers. The first commercial firewall found its place in the market in the early 1990s. The first generation of firewalls were the routers that filtered the internet packets [108] based on some information that they carried. They only controlled access from outside of the organizations or companies. After a while, firewall-to-firewall encryption, user authentication, and virus scanning were also incorporated into the old firewalls. The ubiquity of connectivity to the Internet aroused the new concerns over protecting the integrity and privacy of information,

paving the way for Internet Virtual Private Network (VPN) formation. In this decade, the significant majority of malwares placed their focus on Windows operating systems. This made Microsoft offer several patches to fix security vulnerabilities. Today, the significant growth in the number of cyberattacks fuel demands for cybersecurity tools while they have only protect the machines against known threats. Intrusion Detection Systems (IDSs) are the tools designed to address this issue arise from the regular anti-malwares.

1.3.2 Efforts to Detect and Identify Attacks

Intrusion Detection System (IDS) monitors a system or network traffics in real-time, analyzes them, detects security issues and policy violations, and keeps security administrators informed. IDS is passive which means that it alerts the security administrator when something unusual is detected while it is unable to prevent damages and further attacks. There are three main types of IDS based on the role it plays and the spatial dimension it has, including Network-based, Host-based, and Hybrid IDSs.

A network-based intrusion detection system monitors network traffic and analyzes the network and application protocol activity to identify suspicious activity. In the mid-1990s, two of the most popular IDS were developed. Wheelgroup designed Netranger [109], the initial version of a security product made by the U.S. Air Force. It traced the network traffic for vicious activities, recorded the details about the attacks that may cause any damages to a network, and triggered the real-time alarm to inform the network administrator of suspicious activities that can potentially be a damaging attack. In 1996, Internet Security Systems (ISS) released RealSecure to improve network security with real-time attack detection [110].

Network IDS lost its popularity since the networks had become larger and faster and host-based IDSs taken the place of network-based IDSs progressively. Host-based intrusion detection systems track network traffics on a single personal computer or a server, log and detect malicious activities, and provide real-time processing. Eindhoven University of Technology was subjected to a vicious attack by a hacker who gained root-level access to multiple computer systems at this university. In response to these attacks, TCP Wrappers was developed to keep control of host access, and log unusual activities. TCP wrappers makes use of the strategies that a client and a server communicate across a network. It places itself between the client and server to play the role of a server until the client has thoroughly authenticated to the host [111].

Hybrid intrusion detection systems is comprised of both network and host based intrusion detection, a module that traces the network packets and a module that traces the localhost. Snort is identified as a hybrid ID system [112], utilizing both host-based and network-based IDSs. However, when first released in 1998 for UNIX systems, its capacities were limited. While shortly in 1999, it was able to provide real-time logging and packet analysis, and report abnormal activities to SANS GIAC mobile device security analyst.

1.4 Types of Intrusion Detection

Intrusion indicates any unauthorized activities causing harm to computer systems, and threatening the information confidentiality, integrity or availability. Intrusion Detection Systems (IDS) are any physical or intellectual components that detect unusual activities on computer systems in order to keep them secure. The main goal of using an IDS is to identify malicious network traffic where the traditional security tools are not able to identify them.

IDSs can be classified into two main types: Misuse Intrusion Detection System and Anomaly Intrusion Detection System.

Misuse intrusion detection systems can identify the intrusion with known signatures. In other words, if there are any similarities between an intrusion signature and the signature of a previous intrusions that already recorded in the signature database or logs, the intrusion is detectable. Signatures can be updated manually and automatically. Misuse ID systems are inherently unable to detect unseen attacks.

Anomaly intrusion detection systems can compensate for the weakness of misuse IDSs concerning the capacity to detect new or unknown attacks. In these systems, a model is designed regarding the normal behavior of old observations. Any noticeable difference between the behavior of new observations and the characterizing model can be interpreted as an intrusion and approach trigger an alert. The characterizing model designed by the normal behavior on the network, is created using machine learning and statistics. The designing is comprised of training and testing. The model is trained with normal traffic in the training phase and it is eventually evaluated using unknown instances. Over the past decade, the necessity for analysis of large and complex datasets has arisen. Several strategies have been adopted and now increasingly being used to fulfill this requirement and various models have been designed regarding the type of processing. Two common types of anomaly intrusion detection systems are Statistical and Machine Learning anomaly IDSs.

1.4.1 Statistical anomaly intrusion detection system

The Statistical anomaly intrusion detection system leverages statistical data analysis methods such as mean to form a normal profile. The statistical IDS is employed to find if the new observations deviate from the normal profile. A statistical ID system assigns a score to an instance whose profile mismatches the normal. If the score reaches the threshold value that has been already set on the basis of the number of events that take place over a period of time. There are two types of behavior in statistical anomaly based intrusion detection system, abnormal and normal. These two behaviors are distinguishable by applying statistical properties such as mean and variance of normal activities and statistical test. A scoring mechanism is applied to score an abnormal activity when the calculated score passes a determined threshold value, which will eventuate in an alarm generation. In statistical anomaly based intrusion detection system, the process to detect any type of anomaly starts by creating profiles for current activities and normal activities. Then, any deviation from the normal behavior is determined by comparing these profiles. The two main benefits of applying statistical anomaly based intrusion detection system as a detection system are no requirement for previous knowledge of security problems and the ability to detect new attacks. So, this system is useful to find malicious actions, which are happening in long time periods because it can specify the opportunities for denial of-service attacks. Generally, in statistical approaches, such as applying statistical anomaly based intrusion detection system, accurate statistical distributions are used to model the behaviors while for many of the application domains it is pretty hard to represent the problem formation by applying statistical approaches. They need assumptions on the basis of the parameters of a process, which cannot be proper for accurate anomaly detection systems.

Statistical anomaly based intrusion detection system is categorized into operational models and Markov process models.

1.4.2 Machine learning intrusion detection system

Machine learning is a data analysis approach that can teach the system and can gather knowledge from the tasks conducted by the system. This means that machine learning provides the capability to a system to develop its conducting mechanism. The systems, which are trained by machine learning approaches, are applicable for numerous applications although they are usually costly to implement. They can be implemented in different field of education working with big data such as Chemical engineering, Bio Engineering, and Mechanical Engineering [113-122]. The machine learning techniques encompasses the methods that are analogous to the statistical and data mining techniques. Machine learning techniques can be categorized into neural networks, fuzzy logic, and support vector machines. Neural Networks can apply a series of commands by the user to predict for the subsequent command. The neural network models can be applied to develop user-behavior model because it does not need the direct information of the model. A typical well-trained neural network model with back propagation and feed forward mechanism performs sufficiently as signature matching system. Fuzzy logic can create satisfying reasoning of data and facilitates to manage the uncertainty in the dataset- making it appropriate for many applications. Fuzzy logic have been implemented by applying different techniques since 1990's. One of its significant applications has been in the anomaly based intrusion detection systems. The fuzzy logic system can manage big input data, which may also have undetermined parameters. The systems that apply fuzzy logic are able to decrease the size of input data by applying data mining techniques and extracting

features from the input parameters. Numerous fuzzy intrusion recognition engines are accessible, which applies fuzzy sets and fuzzy rules. Also, many characteristics of fuzzy logic technique are available to make the technique compatible to be related to an intrusion detection system with a correlation.

1.5 List of Datasets for IDS development

The datasets are utilized to evaluate the effectiveness of the methods employing to develop different types of intrusion detection systems. Due to the policies of database sharing and privacy constraints, there are few available datasets in public. DARPA98, KDDCUP 99, NSL-KDD, ADFA, and UNSW-NB15 are the publicly-available and commonly used datasets in cybersecurity.

1.5.1 DARPA98

DARPA98 [123] was the first generation of cybersecurity datasets collected by the Lincoln Laboratory of Massachusetts Institute of Technology (MIT) in 1998. The dataset was comprised of the normal and abnormal network traffic captured from the simulated military network made by Defense Advanced Research Project Agency (DARPA). Nine weeks of raw TCP dump data were collected for simulating Lincoln Air Force Lab. They performed the simulated LAN to scatter it over more attack instances.

The collected network packets contains four gigabytes of compacted TCP dump file gathered from seven weeks of network transaction, containing 2 million connection records in the testing set and 5 million connection records in the training subset.

Although this dataset was broadly used to compare the performance of different approaches adopted to design intrusion detection systems, it was criticized for the low

detection rate and accuracy achieved by models when they were evaluated by the real-life network activities. It also required more memory size and the network transactions had to be converted from TCP dump file into intellectual structure.

1.5.2 KDDCUP 99

This dataset was derived from DARPA98 to compensate for its drawbacks in 1999. It contains 41 features and 4 attack types such as Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), and Probing attacks.

This dataset has been widely used in different studies yet condemned by many authors for lacking in the recent malware attacks, imbalanced number of normal and abnormal records, and the existence of redundant instances. However, the place of KDDCUP 99 [124] is still secure as one of the benchmark cybersecurity dataset involved in several IDS researches [125].

1.5.3 NSL-KDD

Tavallaee et al. [126] proposed NSL-KDD dataset in 2009 to address the issues arouse from its early version, KDDCUP 99. This dataset contains training and testing subsets consists of 125,973 and 22,544 instances, respectively. The number of instances in the NSL-KDD suffice to train machine learning models without the need for embracing randomness in data collection . This dataset is comprised of 4 attack classes as well as normal class, and the similar number of features as of that in KDDCUP 99. Although it was broadly used as a benchmark for intrusion detection research, it is claimed that it contains outdated and unrealistic network transactions.

1.5.4 ADFA

ADFA was proposed by G. Creech and J. Hu at [127] the Australian Defense Force Academy in 2013, providing the contemporary and realistic system calls. It contains the instances from both Linux (ADFA-LD) and Windows (ADFA-WD) operating systems designing for anomaly-based host intrusion detection systems.

To design ADFA-LD, the researchers, selected Ubuntu Linux version 11.04 which was the modern and commonly used version of this operating system at the time. It is comprised of Normal training data, Normal validation data, and Attack data, gathered from the host through its ordinary operation, with transactions ranging from web browsing to LATEX document preparation. On the other hand, ADFA-WD was developed to provide an effective Windows benchmark for IDSs [128-130].

1.5.5 UNSW-NB15

The UNSW-NB15 [131] was generated by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) and released in 2015. This dataset contains 2,540,044 records, providing realistic and modern normal and abnormal network activities, and 49 features including packet-based and flow-based features. The complexity of UNSW-NB15 arise from its structure makes it more comprehensive in the existing network intrusion detection systems evaluation.

Chapter 2

2 Background

2.1 Dataset

The UNSW-NB15 computer security dataset had been released in 2015 [131, 132]. This dataset is comprised of 2,540,044 records. The dataset in available in the UNSW webpage [133] and contains realistic and up to date normal and abnormal network activities. The structure of this dataset is more complex in comparison with the other benchmark datasets such as KDDCUP 99 [134, 135]. This makes the UNSW-NB15 more comprehensive for evaluating the existing network intrusion detection systems in a more reliable way [135].

The records for UNSW-NB15 were gathered by the IXIA traffic generator [136] with three virtual servers. Two servers were configured to distribute the normal traffic packets and the third one was configured to spread the abnormal traffic packets. A total of 49 features including packet-based and flow-based features were extracted from the records by Argus [137] and Bro-IDS tools [138]. Packet-based features are extracted from a packet

header and its payload (also called packet data). In contrast, flow-based features are produced from the sequence of packets, from a source to a destination, traveling in the network. The direction, inter-packet length and inter-arrival times are of paramount importance during packet examination. Record total duration (dur) and destination-to-source-time-to-live (dttl) are two examples of flow-based features. The features are categorized into Basic (numbered from 6 to 18), Content (numbered from 19 to 26) and Time (numbered from 27 to 35). Features numbered from 36 to 40 and 41 to 47 are labeled as general-purpose features and connection features, respectively. General purpose features category includes those features which are intended to explain the purpose of an individual record while connection features depict the characteristic of the connection between a hundred sequentially ordered records. The last two features include attack categories and labels.

Attacks are categorized as Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode and Worms. Normal class is represented using 2,218,761 records while Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms signatures include 24246, 2677, 2329, 16535, 44525, 215481, 13987, 1511, and 174 records, respectively. Consequently, there is considerable lack of balance for the dataset as 87% of the dataset comprises Normal records whereas only 0.007% of the dataset consists of Worms records. Developers of the dataset also subsampled and split the dataset into training and testing subsets as presented in Table 2.1, which has been employed by other researchers [135, 139, 140].

This dataset requires important and essential preprocessing before it can be meaningfully employed for a data-driven model development for intrusion detection. Therefore, it is necessary to provide a visual analysis of the dataset to offer a deep insight into the intricacies of the dataset to help the researchers devise and apply appropriate preprocessing techniques.

Table 2.1 Number of records in training and testing subsets for each class

Classes	**Training Subset**	**Testing Subset**
Normal	56,000	37,000
Analysis	2,000	677
Backdoor	1,746	583
DoS	12,264	4,089
Exploits	33,393	11,132
Fuzzers	18,184	6,062
Generic	40,000	18,871
Reconnaissance	10,491	3,496
Shellcode	1,133	378
Worms	130	44
Total Number of Records	175,341	82,332

2.2. Feature Selection Algorithms

Feature selection refers to identifying a subset of original features that will represent a problem domain accurately. The goals of feature selection algorithms are to choose the smallest subset of features which contains the most predictive power both to simplify the subsequent analysis and improve the performance of machine learning algorithms. These two can be done by reducing the dimensionality through reduction in feature count and eliminating the inherent bias.

In this section, two different feature selection algorithms that belong to wrapper methods and embedded methods for application on the UNSW-NB15 dataset are discussed.

2.2.1. Wrapper Methods

Wrapper feature selection algorithms carry out an assessment on the usefulness of a subset of features by involving a machine learning algorithm. This computational burden makes these methods more expensive in comparison with filter methods. However, they can usually produce the best subsets of features for a specific learning algorithm and consequently lead to improved performance when compared to filter methods [141].

Sequential Feature Selection (SFS) is a naive wrapper feature selection algorithm that starts with a null set and then adds one feature as the first step which depicts the highest value for the objective function [142, 143]. For the second step onwards, the remaining features are added one at a time to the current subset and thus the new subset is evaluated. This process is repeated until the required number of features are added as explained next.

SFS is an iterative method in which it starts with the entire d dimensional feature set (d is the number of features) as input $X = \{x_1, x_2, \dots, x_d\}$. In each iteration, it keeps adding the feature to Y_k until the criterion function including accuracy, F1-measure, FPR etc., is maximized and/or the number of features reaches the value of k, which represents the number of selected features, so it is less than d. The value of k needs to be determined prior to SFS algorithm runs.

$$Y_k = \{y_j \mid j = 1,2, \dots, k;\ y_j \in Y\},\ k = 0, 1, 2, \dots \ \&\ k < d$$

2.2.2. Embedded Methods

Embedded Methods inherit the qualities of both filter methods and wrapper methods such as low computational cost and better performance. Elastic net [144] feature selection

is proposed to select one feature among the highly correlated features in each iteration. It performs bridge regularization [145] with $0 < \gamma \leq 1$ and $\lambda \geq 1$.

$$p(w) = \sum_{i=1}^{n} |w_i|^{\gamma} + \left(\sum_{i=1}^{n} w_i^2 \right)^{\lambda} \tag{2.1}$$

where w_i is a linear classifier, p(w) is a regularization term and n is the number of predictors. Elastic Net is a combination of LASSO [143] and Ridge Regression [146].

$$\hat{\beta}_{en}(\lambda) = \underset{\beta}{\operatorname{argmin}} \left(\frac{\|Y - X\beta\|_2^2}{n} + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1 \right) \tag{2.2}$$

where $\|Y - X\beta\|_2^2 = \sum_{i=0}^{n} (Y_i - (X\beta)_i)^2$, $\|\beta\|_1 = \sum_{j=1}^{k} |\beta_j|$, $Y_i = \beta_0 + x_{i1}\beta_1 + \cdots + x_{ik}\beta_k$, $i = 1, \ldots, n$, n is the number of samples, $\beta_0, \ldots, \beta_k$ are the regression coefficients, and k is the number of predictors. $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are regularization parameters.

2.3. Normalization Methods

2.3.1. Min-Max Scaler

This algorithm is one of the most popular scaling algorithms. The idea is to subtract the minimum of all data (data in one column of the dataset) from each value and divide it by the difference between the minimum value and the maximum one. The idea is to subtract the minimum of all data (data in one column of the dataset) from each value and divide it by the difference between the minimum value and the maximum one. The following equation shows the computation:

$$X_{sc} = \frac{X_{all} - X_{min}}{X_{max} - X_{min}} \tag{2.3}$$

where X_{sc} represents the shrinkage of the original feature values to the interval 0 to 1 when the values are positive or to the interval -1 to 1 for negative data; X_{all} is the entire set of values in a column; X_{min} is the minimum value in a column, and X_{max} is the maximum value in a column.

2.4. Evaluation Metrics

The confusion matrix is used to help derive a set of performance metrics to evaluate the performance of the classifiers. All of the traditional metrics are calculated based on the confusion matrix, namely True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). TP, TN, FP and FN identify the records that are correctly detected as positives, correctly detected as negatives, incorrectly detected as positives, and incorrectly detected as negative data points, respectively. In the case of cyber security, True Negative (TN) is the number of records that are correctly predicted as Normal class and True Positive (TP) is the number of instances that are correctly predicted as attacks. These terms are depicted in Figure 2.1 where C_k represents different classes, k and n indicate the specific class number and total number of classes, respectively.

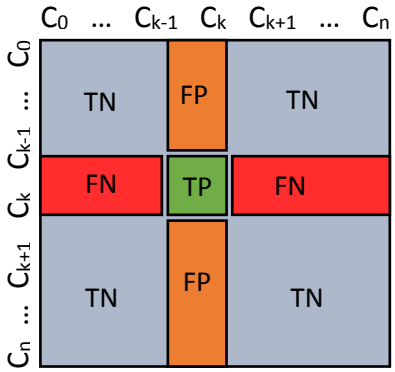


Figure 2-1 Confusion matrix and associated performance metric definitions

The most common metrics are sensitivity, specificity, false positive rate, false negative rate, precision and accuracy. However, for imbalanced datasets, the imbalance needs to be considered during the evaluation measurements. Thus, the accuracy is not an appropriate metric for the case of class imbalance [147]. Instead, F-measure is offered by [148] for imbalanced datasets. In this study, we measure the accuracy in order to perform a comparative evaluation with the same metric employed in studies reported in the literature. Nex, we explain the metrics which are used in this study in the following paragraphs in detail.

Accuracy is identified as the ratio of the correct classifications to the total number of samples and defined by the following formula:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.4}$$

Sensitivity or Detection Rate (DR) corresponds to the proportion of true positives with respect to all positives. It measures the probability of a sample being actually positive from all positive data points. Specificity indicates the proportion of false positives to all negatives. It measures the probability of a sample being actually positive from all data points that are predicted to be positive. They are used when only positives or negatives matter.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.5}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.6}$$

False Positive Rate (FPR) or False Alarm Rate (FAR) represents the ratio of incorrect positive predictions to the overall number of negatives. While the Precision measures the probability of samples classified as positives for actually being positive. These two metrics are defined as follows:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{2.7}$$

and

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2.8}$$

False Negative Rate (FNR) or Missed Alarm Rate indicates the ratio of incorrect negative predictions to the total number of positives:

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} \tag{2.9}$$

F-measure is the harmonic mean of Sensitivity and Precision and is given by

$$\mathrm{F-measure} = \frac{2 \times \mathrm{sensitivity} \times \mathrm{precision}}{\mathrm{sensitivity} + \mathrm{precision}} \tag{2.10}$$

This metric is the weighted harmonic mean of precision and sensitivity [149]. It takes into account uneven level of importance of FN and FP.

$$\begin{aligned} \mathrm{F}_\beta &= (1 + \beta^2)\frac{\mathrm{ps}}{\beta^2\mathrm{p} + \mathrm{s}} \\ &= \frac{(1 + \beta^2)\mathrm{TP}}{(1 + \beta^2)\mathrm{TP} + \beta^2\mathrm{FN} + \mathrm{FP}} \end{aligned} \tag{2.11}$$

where p represents the precision and s represents the sensitivity; and β is a weight attached to precision and sensitivity.

2.5. Classification Algorithms

2.5.1 Random Forest (RF)

Random Forest is an ensemble method comprising the collection of base estimators, typically decision trees. An individual decision tree has been formed by a different bootstrap sample of the original data and a set of randomly selected features (input variables) given the best split in the creation of sub nodes. Bootstrap sampling is a statistical data resampling method taking random number of samples with replacement from the training subset in order to train each estimator. Randomly chosen sub-samples and features has reduced the correlation between the estimators which leads to the accurate model. Also, the aggregation of several base estimators has made a robust model minimizing the generalization error in comparison with a single estimator.

Each decision tree has been trained with a sub-sample of original training data which is randomly chosen. Accordingly, some samples may be used to train the decision trees and some other may never be utilized. Random forest designate a randomized subset of features to construct each estimator and choose a best split from the designated features letting a decision tree grow. In this study, Hellinger distance has been utilized as a split criterion. Assuming the binary-class classification where X_A and X_B represent the distribution of class A and class B, respectively.

$$d_H(X_A, X_B) = \sqrt{\left(\sqrt{\frac{X_A^{left}}{X_A^{parent}}} - \sqrt{\frac{X_B^{left}}{X_B^{parent}}}\right)^2 + \left(\sqrt{\frac{X_A^{right}}{X_A^{parent}}} - \sqrt{\frac{X_B^{right}}{X_B^{parent}}}\right)^2} \qquad (2.12)$$

X_A^{left} and X_B^{left} represent the number of samples of class A and B in the left child node, X_A^{right} and X_B^{right} denote the number of class A and B samples in the right child node, and X_A^{parent} and X_B^{parent} indicate the total number of class A and B samples in the corresponding parent node, respectively.

Hellinger distance is in the range of $[0, \sqrt{2}]$. If class A and class B have been mutually exclusive, $d_H(X_A, X_B) = 0$ (maximal affinity) otherwise, $d_H(X_A, X_B) = \sqrt{2}$ (minimal affinity) [150-152]. Hellinger distance is used to trace the tendency of the selected features to generate the base estimators showing the minimal affinity between class A and class B. This criterion has addressed the skew sensitivity affecting the performance of decision trees.

After the decision trees being made, they have predicted the new data that they had not been trained by. The predictions have been eventually aggregated using bagging method to count the majority votes associated with each estimator. This method battles the overfitting problem and makes powerful learners.

2.5.2 Balanced Bagging (BB)

Balanced Bagging is the combination of standard bagging and resampling strategies. Bagging is an ensemble method implementing with the aim of improving the performance of a single weak estimator. Several base estimators have been aggregated in this strategy to form a strong learner. Each estimator has been trained with bootstrap data. The predictions have been eventually collected to make a final decision through a majority vote. Resampling is a technique which is used to modify the distribution of data in order to deal with the imbalanced class issue. It consists of oversampling and undersampling methods.

The simplest oversampling, namely random oversampling, represents the randomly selected samples from the minority class with replacement [153]. Random undersampling indicates the samples that have been selected randomly from the majority class either with or without replacement.

In this study, the hybrid of bagging and undersampling is utilized which is also called UnderBagging. This method is first proposed by Barandela et al. [154] in 2002. The authors concluded that the final learner presented the same results with and without data replacement.

2.5.3. Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting is comprised of sequence of decision trees utilizing gradient descent algorithm in order to minimize the errors of weak estimators in which the objective function consists of training loss and regularization term,

$$\mathcal{L}(\phi) = \sum_{i} l\left(y_{i_i}, \hat{y}_i\right) + \sum_{k} \Omega(f_k) \tag{2.13}$$

where $\sum_i l(y_{i_i}, \hat{y}_i)$ indicates the loss function and $\sum_k \Omega(f_k)$ represents the regularization term which is equal to $\gamma T + \frac{1}{2}\lambda\|w\|^2$. Extreme Gradient Boosting utilize the same regularization strategy as Regularized Greedy Forest [155] has used. The final weights gained by training the model have become smooth using this additional regularization.

In this algorithm, all the trees are trained once at a time improving the performance of the algorithm in terms of its run time. Every loss function at step t can be optimized by

taking the first and the second order gradient statistics, $g_i$and h_i, . Accordingly, the objective function for the new tree in the general setting can be shown as follows,

$$\mathcal{L}^{(t)}(q) = \frac{1}{2}\sum_{j=1i}^{T} \frac{\left(\sum_{i\epsilon I_j} g_i\right)^2}{\sum_{i\epsilon I_j} h_i + \lambda} + \gamma T \tag{2.14}$$

where $g_i = \partial_{\hat{y}_i^{(t-1)}} l\left(y_{i_i}, \hat{y}_i^{(t-1)}\right)$ and $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l\left(y_{i_i}, \hat{y}_i^{(t-1)}\right)$ and T is the number of leaves. To identify all the possible tree structures (q), basic exact greedy algorithm, approximate algorithm [156, 157], weighted quantile sketch using the idea of quantile sketch [158], and sparsity-aware split finding are utilized and proposed. The split candidates are eventually evaluated by the following gain function after the split,

$$\text{Gain} = \frac{1}{2}\left[\frac{\left(\sum_{i\epsilon I_L} g_i\right)^2}{\sum_{i\epsilon I_L} h_i + \lambda} + \frac{\left(\sum_{i\epsilon I_R} g_i\right)^2}{\sum_{i\epsilon I_R} h_i + \lambda} - \frac{\left(\sum_{i\epsilon I} g_i\right)^2}{\sum_{i\epsilon I} h_i + \lambda}\right] - \gamma \tag{2.14}$$

where I_L is the instance set of left nodes and I_R is the instance set of right nodes. In fact, it acts like the pruning strategies in tree-based models optimizing one level of the tree in order to avoid overfitting. Also, shrinkage technique [159] and feature subsampling [159] are the two additional strategies used in this algorithm to defend against overfitting.

2.6. Splitting Criterion

2.6.1. Hellinger Distance

Hellinger distance is a split metric which was first utilized along with decision tree [150]. It is derived from the Bhattacharyya coefficient measuring the similarity between two probability distribution [160]. If the conditional probabilities of class A and class B given X, which is drawn from discrete or continuous set of attribute values V, are

$P(Y_A|X)$ and $P(Y_B|X)$, the separability of class A and class B of data conditioned over the full set of attribute values are measured as follows,

$$d_H\left(P(Y_A), P(Y_B)\right) = \sqrt{\sum_{i \epsilon V} \left(\sqrt{P(Y_A|X_i)} - \sqrt{P(Y_B|X_i)}\right)^2} \tag{2.15}$$

In other words, if there is a continuous attribute, i is comprised of left and right where left represents the values of an attribute that are less than or equal to the threshold and right indicates the values of an attribute that are above the threshold. Accordingly, the equation can be restated as follows,

$$d_H\left(P(Y_A), P(Y_B)\right) = \sqrt{\left\{\sqrt{\frac{N_A^{left}}{N_A^{parent}}} - \sqrt{\frac{N_B^{left}}{N_B^{parent}}}\right\}^2 + \left\{\sqrt{\frac{N_A^{right}}{N_A^{parent}}} - \sqrt{\frac{N_B^{right}}{N_B^{parent}}}\right\}^2} \tag{2.16}$$

where N_A^{left} represents the number of the samples in class A in the left child of the tree, N_A^{right} represents the number of the samples in class A in the right child of the tree, and N_A^{parent} represents the number of the total samples in class A in the parent node of the tree. The result of $d_H\left(P(Y_A), P(Y_B)\right)$ is confined between 0 and $\sqrt{2}$ where 0 indicates the worst and $\sqrt{2}$ indicates the best split.

Chapter 3

3 Analysis of UNSW-NB15 Dataset

3.1 Visualization of dataset

In this section, we analyze the UNSW-NB15 dataset to identify and discuss significant issues of the dataset [131, 134, 135]. Consequently, two prominent issues are identified for analysis, namely class imbalance and class overlap. Class imbalance is composed of between-class and within-class imbalance. Between-class imbalance corresponds to the case where one class or multiple classes in a dataset are underrepresented in comparison with other classes. In other words, a dataset shows significantly unequal distribution among its classes. To illustrate this problem, we plot the distribution of entire attack classes for the UNSW-NB15 dataset in Figure 3.1. As seen in this figure, normal records constitute 87% of all records while the combined record count for all 9 attack classes is only 13%. Additionally, 13% of records in the overall dataset are not equally distributed among the 9 attack classes as 65% of all attack records belong to the Generic attack class while only 0.0008% of all attack records belong to the Worms attack class.

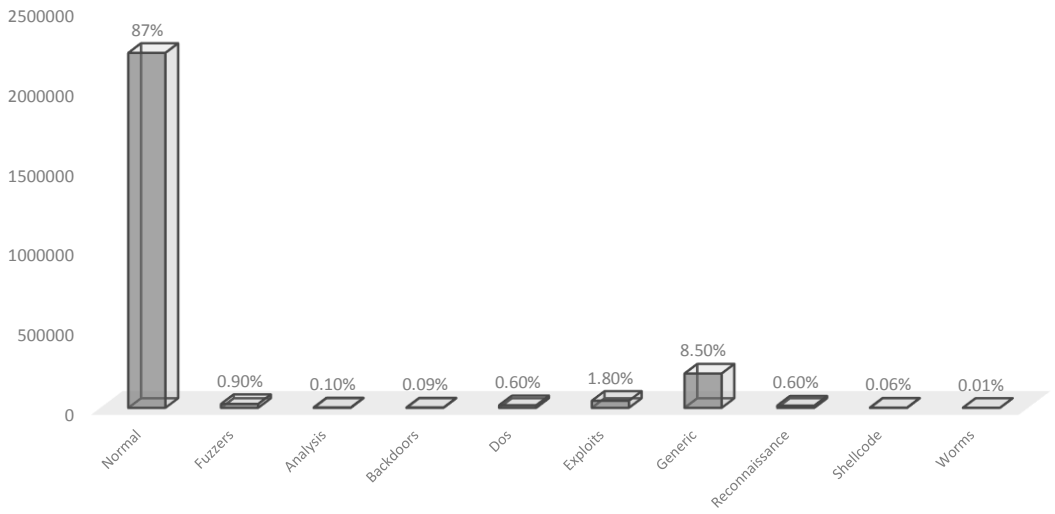


Figure 3-1 Distribution of records across classes

Within-class imbalance, on the other hand, represents the case where one class is comprised of several different subclasses with different distributions. To discover whether the classes are made up of imbalanced sub-clusters, we use two visualization techniques, namely PCA and T-SNE. Figures 3.2 and 3.3 show the within-class imbalance in the UNSW-NB15 dataset. As Figure 3.2 shows, the Exploits attack class is composed of several different size clusters where clusters are compact. The clustering is different for the Worms attack class for which there are two but different size cluster groupings as shown in Figure 3.3. The larger cluster grouping is composed of multiple subgroupings with gaps in between. Figure 3.4 illustrates all the concepts using a 2-dimensional scatter plot by considering all the data points in the dataset. All classes have multiple clusters of different sizes and spread across the two-dimensional analysis space. Many classes are composed of a few relatively large clusters and many small clusters. Additionally, the boundaries separating classes are not clear cut: there is noticeable overlap between or among multiple clusters belonging to different classes.

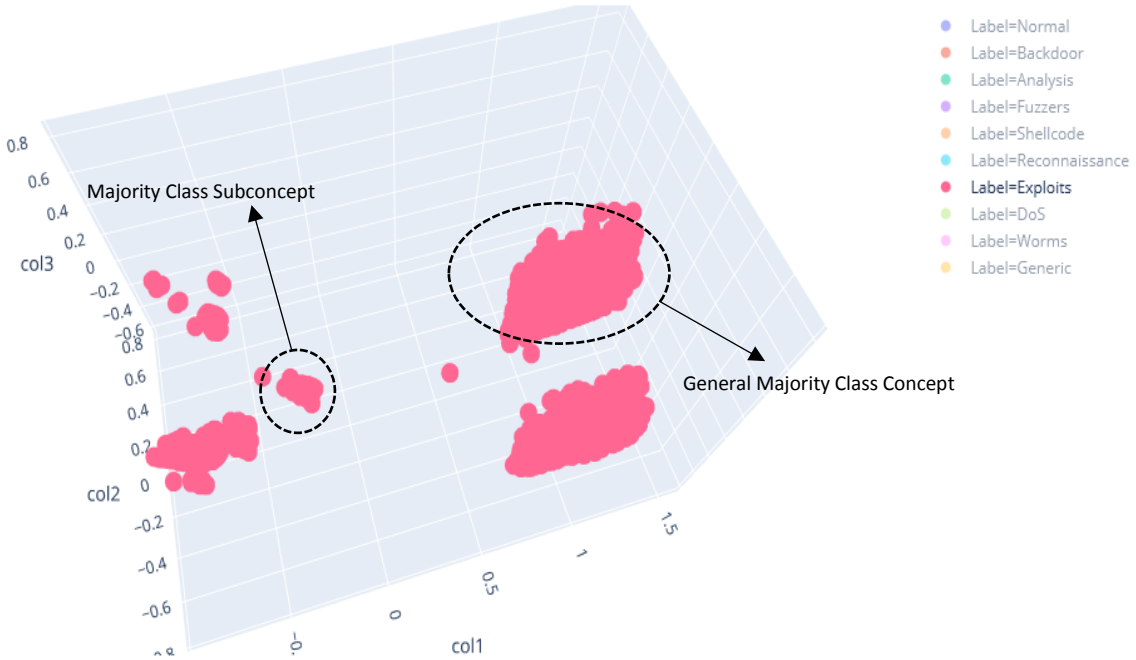


Figure 3-2 The visualization of the Shellcode instances using PCA

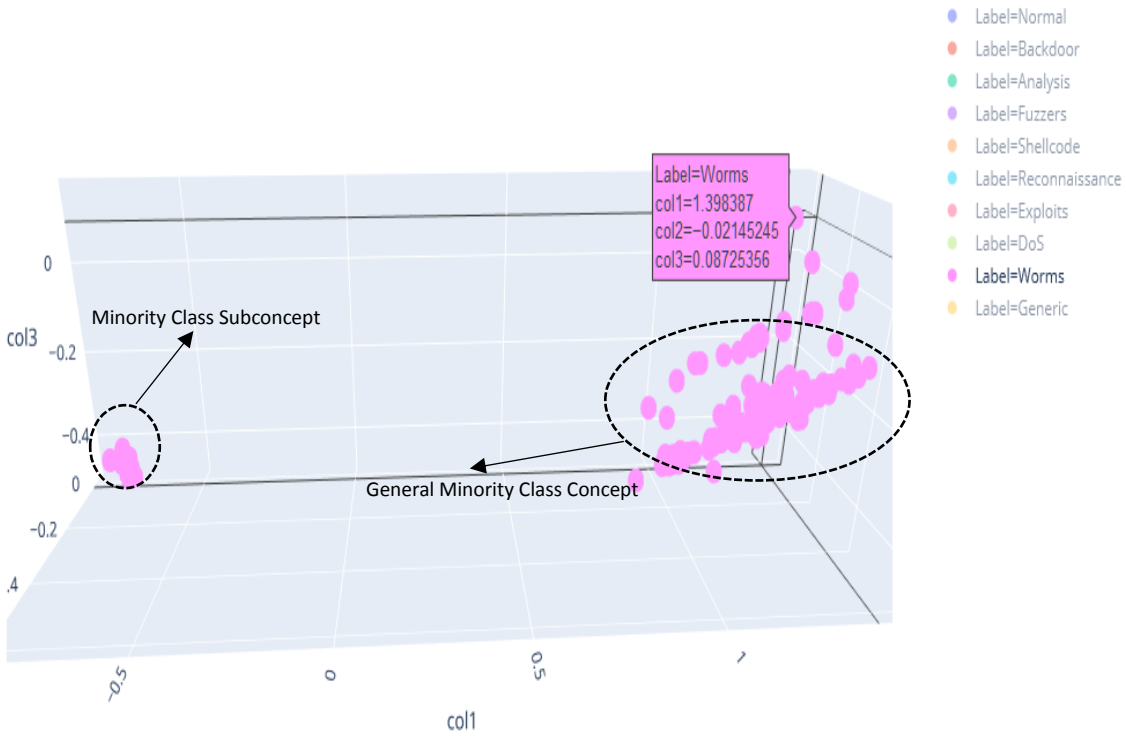


Figure 3-3 The visualization of the Worms instances using PCA

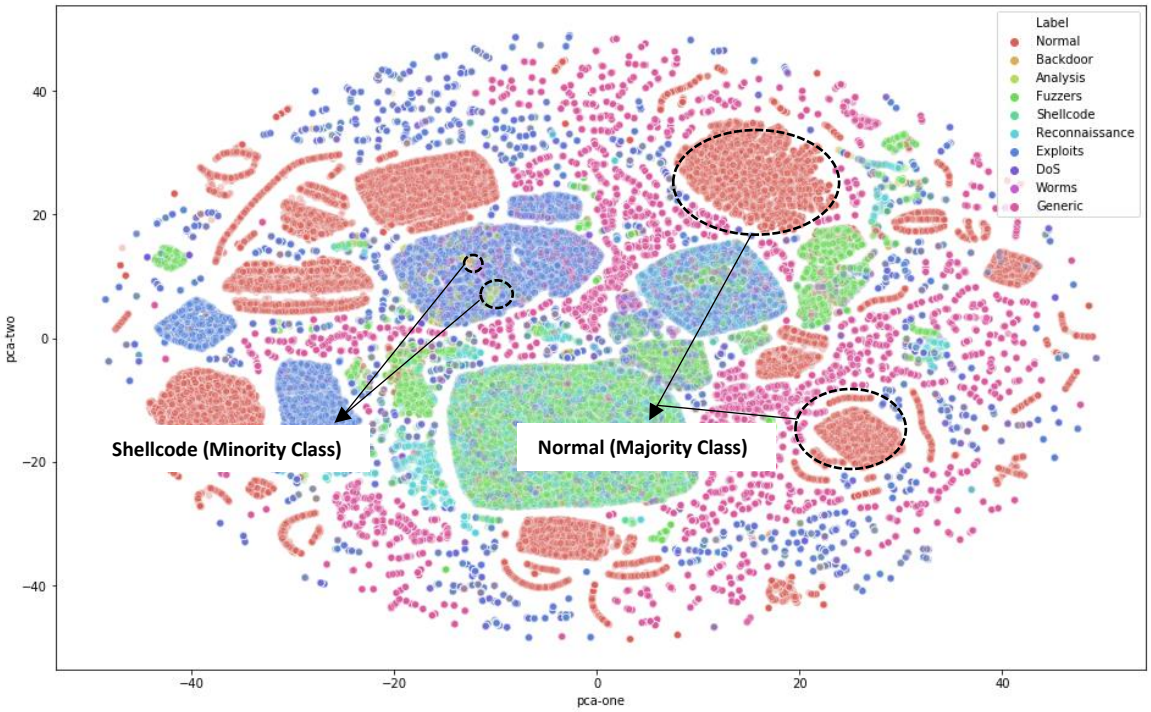


Figure 3-4 The visualization of the dataset using T-SNE

This dataset also suffers from the so-called "overlap problem." In particular, many attack class records mimic the behavior of the Normal records. While the emphasis of intrusion detection systems is detecting and/or identifying the malicious network traffic, if it is trained by this dataset without addressing the overlap problem, a satisfactory outcome will not be achieved. In order to expose the degree or scale of this problem, we first sketch the data points in a 3-dimensional scatter plot as shown in Figure 3.5. The same figure shows that many attack classes overlap as indicated with the content of the dotted circles.

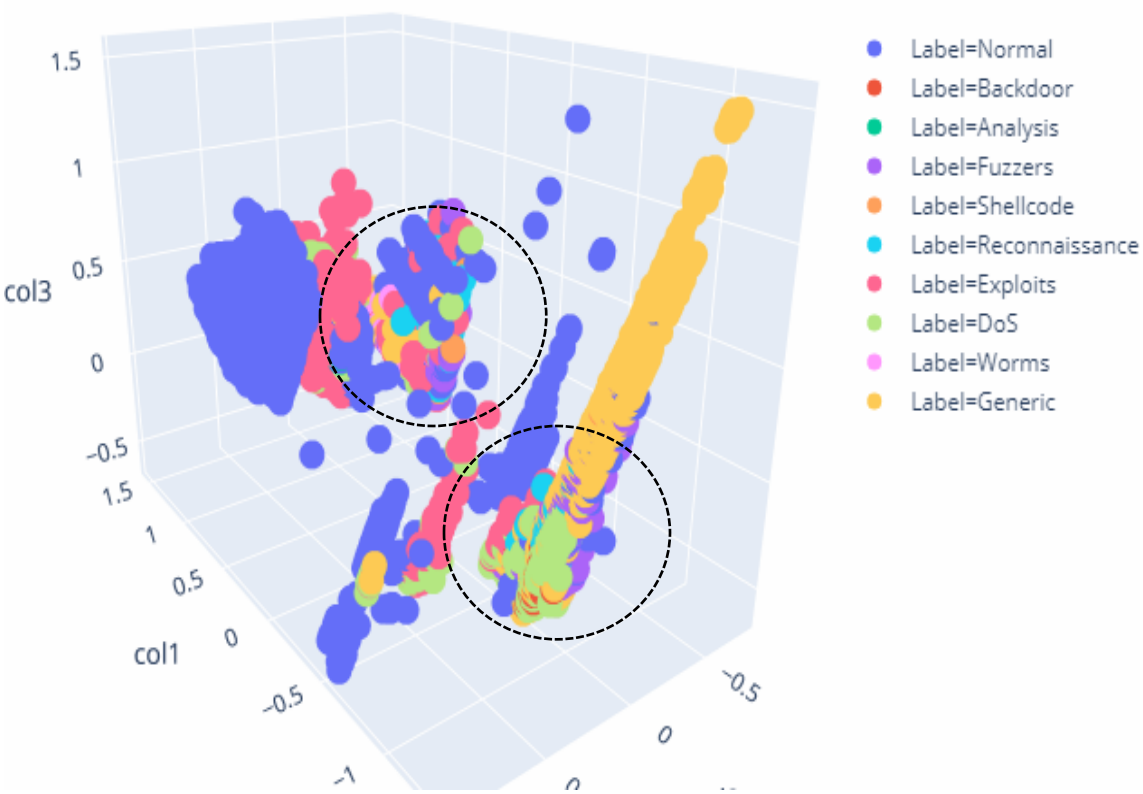
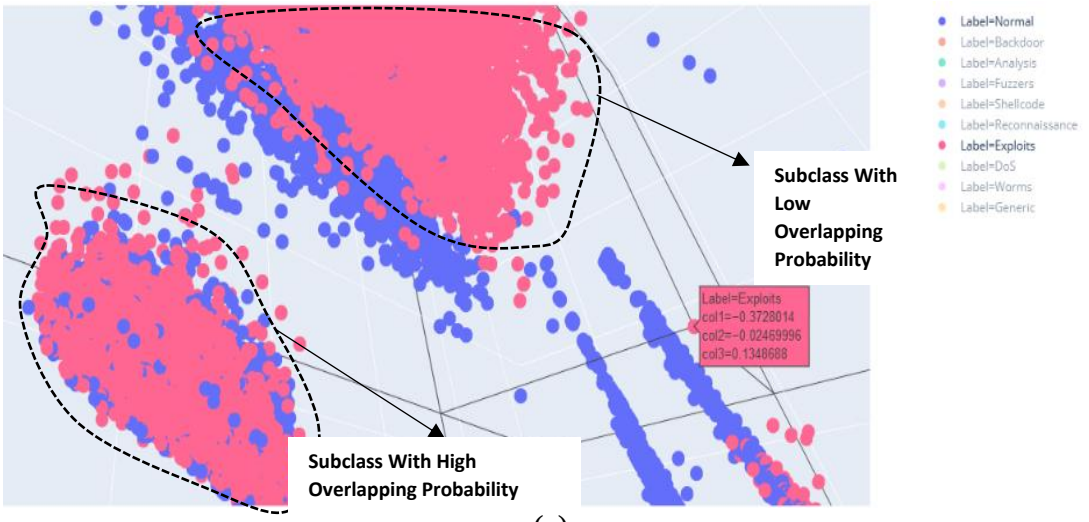
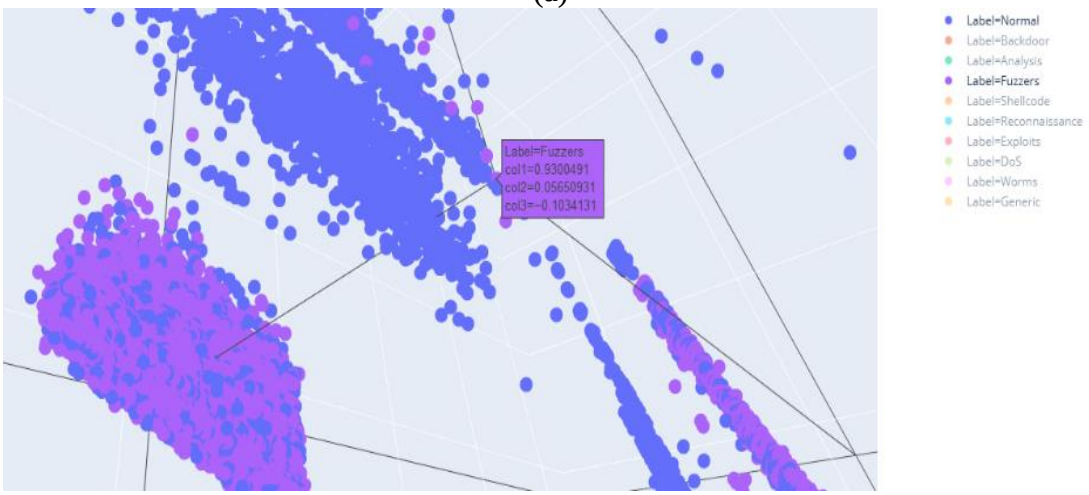


***Figure* 3-5** *Visualizing the overlapping problem inherent in the dataset using the PCA*

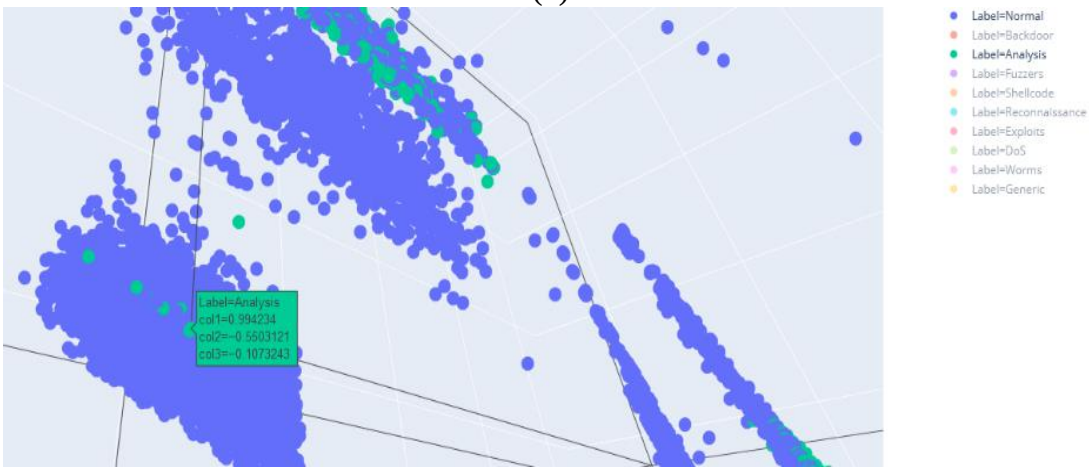
We further investigate for those attack class records which reside in the subspace where primarily Normal records are present. Figure 3.6 details the overlap cases among a subset of attack classes and the Normal class. As Figure 3.6 (a) illustrates, Exploits attack class have the overlapping problem with the Normal class. In Figure 3.6, there is noticeable overlap between the sets of data points belonging to Exploits (in red) and those belonging to Normal class (in navy blue). In Figure 3.6 (b), it is easy to observe that records belonging to Fuzzers and Normal classes are clustered together for the cluster in the lower left. Figure 3.6 (c), exposes the overlapping problem between Analysis and Normal classes.



(a)



(b)



(c)

Figure 3-6 The visualization of the Normal along with the (a) Exploits, (b) Fuzzers, and (c) Analysis data points

Figure 3.7 shows overlaps for the entire dataset. We used the K-means clustering method to compute the clusters. Inter-cluster distances are also utilized to sketch the map. As the figure shows, the degree or amount of overlap among classes 8, 4 and 1 as well as between 0 and 6, and between 9 and 2 are substantial.

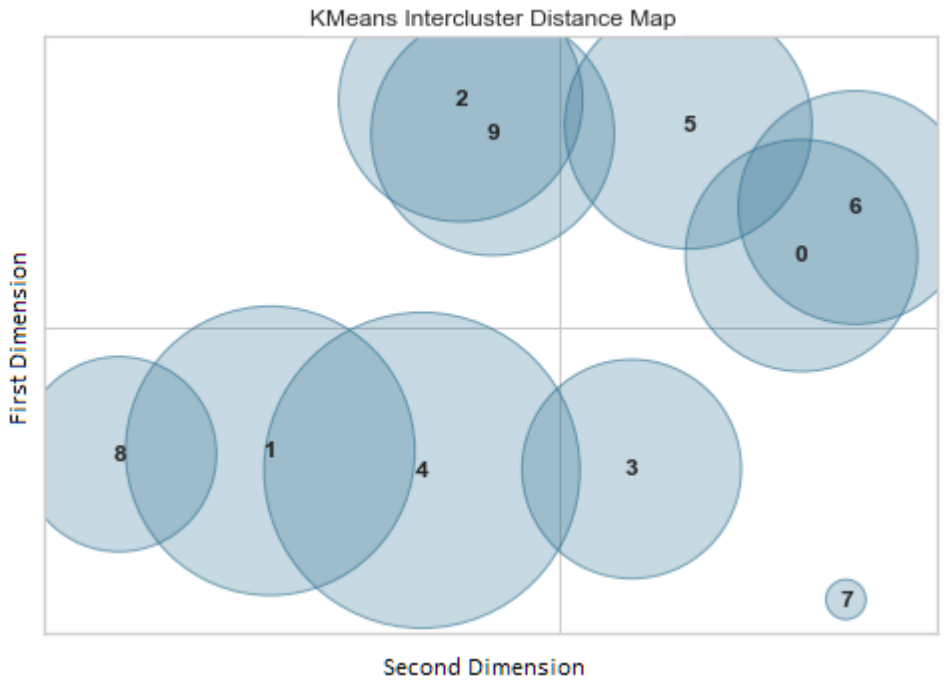


Figure 3-7 Mapping the dataset by 10 clusters using k-mean clustering algorithm

The UNSW—NB15 dataset is a good option for empirical studies on intrusion detection system development using data-driven approaches. However, the two major issues, namely class imbalance and class overlap, need to be addressed prior to being employed for model development. Class imbalance and overlap, if not addressed, are likely to hinder the attack detection and identification performance of an intrusion detection system developed using this dataset. We have used a variety of visualization techniques to expose and illustrate the degree of prominence of these two problems within this dataset. Given the degree of severity as presented through the visualization, it is imperative that

effective approaches need to be implemented to mitigate the adverse effects of these two problems on classification performance of any data-driven statistical or machine learning model.

Chapter 4

4 Data Preprocessing

Data preprocessing employs variety of methods such as data cleaning, data transformation and feature selection to convert the raw data into an appropriate framework or format prior to processing with a learning algorithm. These methods have significant impact on machine learning performance. In this section we discuss these methods in detail and show how they help optimize the performance of the proposed model.

4.1 Data Cleaning

Data cleaning eliminates the redundant observations from the dataset, which consists of duplicate or irrelevant records or features. There are no duplicate features or records in the UNSW-NB15. Yet, the entire 49 features are not required to be necessarily used. In this dataset, four features are specific to the computing infrastructure such as source IP address, source port number, destination IP address, and destination port number. Another two features, namely record_start_time and record_last_time, may not be very useful either

[134]. Keeping these features may lead the machine learning model to incur performance penalty as it may not generalize reliably due to potential for overfitting.

For the 43 remaining features, two of them are target features, attack_cat and label. The attack_cat feature is of type nominal and contains the names of attack categories. For the purpose of multi-class classification, this feature is needed. The feature label is binary valued for which a 0 value indicates normal and a 1 value shows attack records, which is relevant for binary classification cases. Accordingly, we use this feature for binary classification.

4.2 Data Transformation

Among the 41 non-target features, three of them are nominal. We convert the nominal features to numerical as most of the machine learning models and scalers can readily work with the numerical values. To convert the nominal to numerical features, we use label encoder implemented as in scikit-learn in Python. This transformation method is used to convert the nominal values to the numerical values using the following formula: numerical representation is computed as a value between 0 to the count of nominal values minus one.

We also transform all the features to the same range of values using scalers. This helps most learning algorithms weigh in the entire features equally. We measure nearest shrunken centroid [161] for each attack class. Let x_{ij} indicate the $i-\text{th}$ input variable (feature) where $i = 1,2,\dots,p$ and $j-\text{th}$ sample (record), where $j = 1,2,\dots,n$; p is the number of predictors (features), and n is the number of samples. If we have $k = 1,2,\dots,K$ and C_k is n samples in class k then the mean expression value in class k for input variable i is $\bar{x}_{ik} = \sum_{j\in C_k} x_{ij}/n_k$ and the mean expression value for overall class is $\bar{x}_i = \sum_{j=1}^{n} x_{ij}/n$.

To measure the distance between the attack class centroids, the Euclidean distance and Mahalanobis distance [162] are used. Figures 4.2 and 4.4 depict the Euclidean distance and Mahalanobis distance of class centroids against each other where six different scalers are implemented, respectively. Figures 4.1 and 4.3 represent the same distances when no scalers are applied on the dataset. Figure 4.1 shows that the distance between the centroids of Generic and Worms are much larger while Exploits and Fuzzers are close to most of the class centroids. In Figure 4.2, Normalization shows no impact on the distance values. Instead, the Min-Max scaler increases the distance between the centroids of the attack classes against the Normal class. Also, it seems that the distance of the Worms cluster is increased by robust scaling. No considerable impact can be ascertained for the distance values when standard scaler, quantile transformer, and power transformer are implemented.

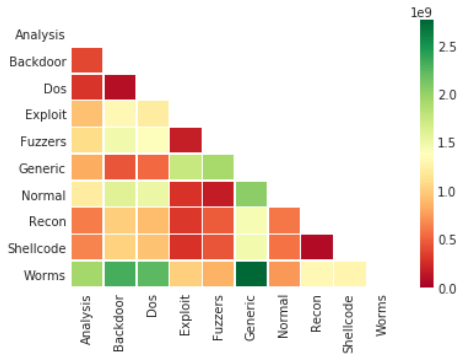


Figure 4-1 The Euclidean distance of the class centroids when the dataset is not normalized

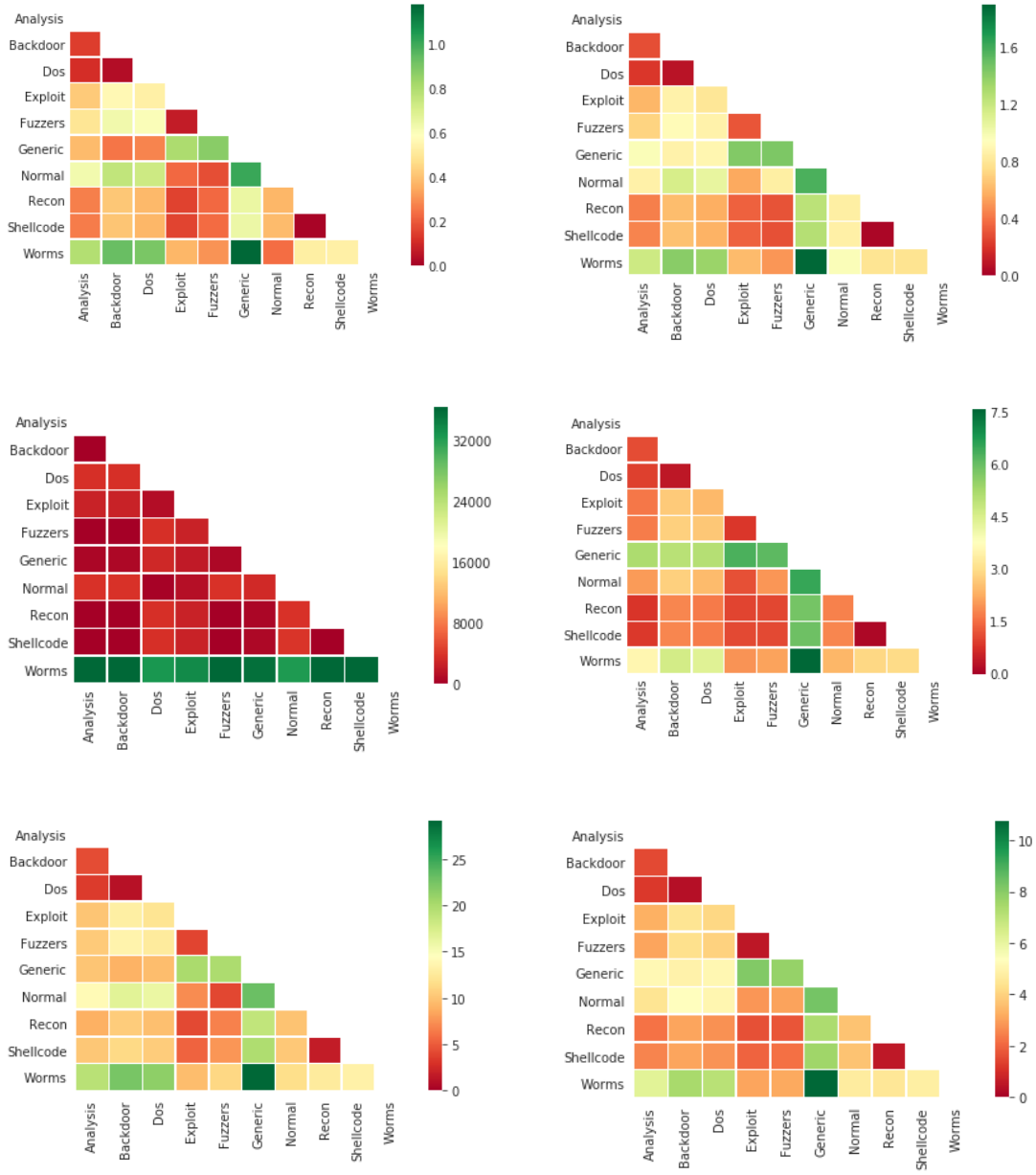


***Figure* 4-2** *The Euclidean distance of the centroids after normalizing the data (upper-left), scaling the dataset with min-max, maxabs (upper-right), robust (left-center), standard (right-center), quantile transformation (lower-left), and power transformation (lower-right)*

Although, the Euclidean distance is the most commonly used metric to measure the straight-line distance, it treats each feature equally while it is assumed that there is a correlation between the features. If the input variables are correlated to one another,

Euclidean distance provides misleading information regarding how similar two clusters are. Moreover, each scaler measures the distance values using different ranges when Euclidean distance is used. For instance, in Figure 4.2 for the robust scaler, the largest distance is 32,000 while for the standard scaler the largest distance is 7.5. Accordingly, it is hard to interpret the heat maps to compare the performances of the scalers qualitatively. Mahalanobis distance measures the correlation between the features as well as addressing the issue about comparing the heat maps by giving the distance values using the same scale.

Figure 4.4 demonstrates the link between the type of data normalization and the distance between the centroid of each class type where Mahalanobis distance is utilized as the distance measurement criterion. Comparing the heat maps, we can see that min-max scaler, and power transformation deliver almost the same results. They are both comprised of further number of green cells. Also, it seems that min-max scaler, robust scaler, and power transformation increased the distance between the Normal class centroid and the attack classes' centroids.

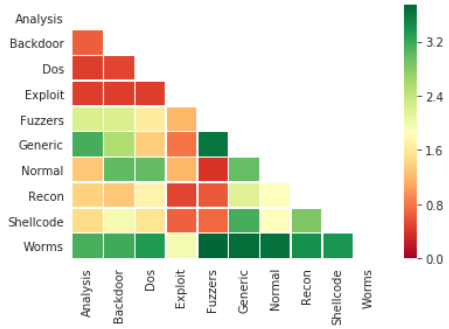


Figure 4-3 The Mahalanobis distance of the centroids when the dataset in not normalized

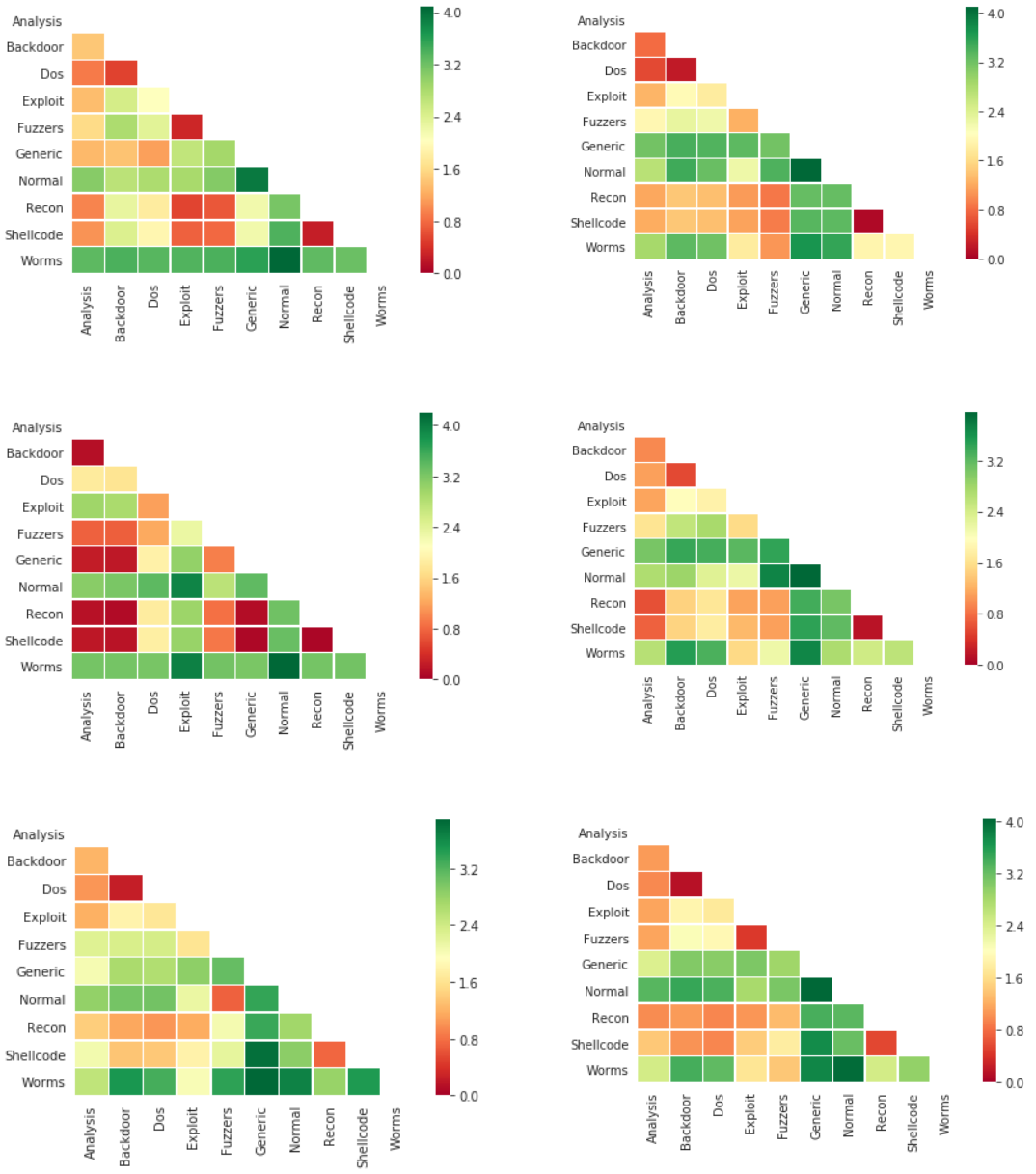


***Figure* 4-4** *The Mahalanobis distance of the centroids after normalizing the data (upper-left), scaling the dataset with min-max, maxabs (upper-right), robust (left-center), standard (right-center), quantile transformation (lower-left), and power transformation (lower-right)*

Next, we present the values in tables to facilitate comparison of the distance values quantitatively. This helps for the analysis of the heat maps quantitatively by tabulating the values representing the distances associated with each class centroid. They are presented

in Tables 4.1 through 4.10. These tables indicate that Min-Max scaler, Robust scaler and Quantile transformer increase the centroid distances for more cases. The main point of interest is in an algorithm that increases the distances among class centroids particularly Normal class against the attack classes in intrusion detection systems to decrease the false negative rate. Min-Max scaler is the one that increases the centroid distance between the Normal class and the rest. This is likely to help the intrusion detection systems identify the normal records against attack data points which will lead to a decrease for the missed alarm rate value. Analysis of data in Tables 4.2, 4.3, 4.6 and 4.8 show that distance values between any one of Backdoor, DoS, Generic and Reconnaissance classes and the Normal class for the Min-Max scaler are the largest compared to those for the other scalers. Distance values between any one of Fuzzers, Shellcode and Worms classes and the Normal class for the Min-Max scaler compared to distance values by the other scaler algorithms are also competitive.

Table 4.1 The Mahalanobis distances between the Analysis centroid and the rest

	Backdoor	Dos	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
Original	0.67	0.45	1.64	2.23	1.41	2.57	1.31	1.47	3.15
Normal	**1.42**	0.89	1.34	1.61	1.32	3.06	0.97	1.06	**3.31**
Min-Max	0.80	0.58	1.28	1.94	**3.19**	2.71	1.19	1.22	2.85
Robust	0.13	**1.82**	**2.98**	0.76	0.26	3.13	0.16	0.20	3.25
Standard	0.97	1.11	1.14	1.64	3.05	2.70	0.62	0.72	2.64
Quantile	1.23	1.03	1.19	**2.27**	2.04	2.84	**1.43**	**2.07**	2.53
Power	1.10	0.98	1.17	1.15	2.38	**3.29**	1.01	1.42	2.47

Table 4.2 *The Mahalanobis distances between the Backdoor centroid and the rest*

	Analysis	Dos	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
Original	0.67	0.50	2.24	2.77	0.99	3.02	1.89	1.98	3.19
Normal	**1.42**	0.53	2.49	**2.79**	1.40	2.68	**2.29**	**2.42**	3.40
Min-Max	0.80	0.23	1.98	2.29	**3.44**	**3.78**	1.43	1.43	3.32
Robust	0.13	**1.74**	**2.89**	0.75	0.22	3.22	0.11	0.16	3.25
Standard	0.97	0.56	1.96	2.55	3.41	2.82	1.47	1.49	**3.50**
Quantile	1.23	0.25	1.79	2.33	2.68	3.01	1.12	1.34	**3.50**
Power	1.10	0.15	1.89	2.08	3.05	3.47	1.08	1.04	3.39

Table 4.3 *The Mahalanobis distances between the DoS centroid and the rest*

	Analysis	Backdoor	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
Original	0.45	0.50	**2.07**	2.34	1.37	3.01	1.73	1.55	**3.35**
Normal	0.89	0.53	2.05	2.35	1.13	2.79	1.79	**1.92**	3.34
Min-Max	0.58	0.23	1.79	2.21	**3.39**	**3.43**	1.36	1.37	3.21
Robust	**1.82**	**1.74**	1.17	1.23	1.90	3.39	**1.84**	1.88	3.24
Standard	1.11	0.56	1.81	**2.74**	3.34	2.29	1.67	1.75	3.31
Quantile	1.03	0.25	1.64	2.36	2.64	3.04	1.02	1.37	3.27
Power	0.98	0.15	1.74	1.94	3.01	3.36	0.97	0.96	3.26

Table 4.4 *The Mahalanobis distances between the Exploits centroid and the rest*

	Analysis	Backdoor	Dos	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
Original	1.64	2.24	**2.07**	2.28	2.53	1.19	0.39	1.98	3.39
Normal	1.34	2.49	2.05	0.33	2.64	2.83	0.54	0.74	3.37
Min-Max	1.28	1.98	1.79	1.26	**3.32**	2.39	1.11	1.15	1.79
Robust	**2.98**	**2.89**	1.17	**2.32**	3.06	**3.94**	**2.99**	**3.03**	**3.97**
Standard	1.14	1.96	1.81	1.55	3.22	2.17	1.11	1.27	1.54
Quantile	1.19	1.79	1.64	1.61	2.87	2.14	1.16	1.78	2.01
Power	1.17	1.89	1.74	0.46	3.08	2.79	1.06	1.45	1.70

Table 4.5 *The Mahalanobis distances between the Fuzzers centroid and the rest*

	Analysis	Backdoor	Dos	Exploit	Generic	Normal	Recon	Shellcode	Worms
Original	2.23	2.77	2.34	2.28	**3.63**	3.42	**2.30**	0.80	**3.74**
Normal	1.61	**2.79**	2.35	0.33	2.87	3.07	0.68	0.78	3.40
Min-Max	1.94	2.29	2.21	1.26	3.19	3.64	0.88	0.92	1.07
Robust	0.76	0.75	1.23	**2.32**	0.96	2.76	0.85	0.91	3.26
Standard	1.64	2.55	**2.74**	1.55	3.45	**3.72**	1.11	1.11	2.16
Quantile	**2.27**	2.33	2.36	1.61	3.10	0.71	2.05	**2.19**	3.39
Power	1.15	2.08	1.94	0.46	2.86	3.09	1.31	1.79	1.40

Table 4.6 The Mahalanobis distances between the Generic centroid and the rest

	Analysis	Backdoor	Dos	Exploit	Fuzzers	Normal	Recon	Shellcode	Worms
Original	1.41	0.99	1.37	2.53	**3.63**	2.99	2.16	2.86	3.69
Normal	1.32	1.40	1.13	2.64	2.87	3.93	2.19	2.19	3.58
Min-Max	**3.19**	**3.44**	**3.39**	**3.32**	3.19	**4.09**	3.27	3.35	3.70
Robust	0.26	0.22	1.90	3.06	0.96	3.38	0.11	0.07	3.22
Standard	3.05	3.41	3.34	3.22	3.45	3.96	3.34	3.45	3.71
Quantile	2.04	2.68	2.64	2.87	3.10	3.37	3.33	**3.82**	**3.89**
Power	2.38	3.05	3.01	3.08	2.86	4.03	**3.38**	3.72	3.77

Table 4.7 The Mahalanobis distances between the Normal centroid and the rest

	Analysis	Backdoor	Dos	Exploit	Fuzzers	Generic	Recon	Shellcode	Worms
Original	2.57	3.02	3.01	1.19	3.42	2.99	1.39	3.15	3.67
Normal	3.06	2.68	2.79	2.83	3.07	3.93	3.16	**3.40**	4.08
Min-Max	2.71	**3.78**	**3.43**	2.39	3.64	**4.09**	**3.31**	3.32	3.81
Robust	3.13	3.22	3.39	**3.94**	2.76	3.38	3.29	3.33	**4.20**
Standard	2.70	2.82	2.29	2.17	**3.72**	3.96	3.05	3.19	2.72
Quantile	2.84	3.01	3.04	2.14	0.71	3.37	2.74	2.88	3.68
Power	**3.29**	3.47	3.36	2.79	3.09	4.03	3.29	3.20	4.01

Table 4.8 The Mahalanobis distances between the Reconnaissance centroid and the rest

	Analysis	Backdoor	Dos	Exploit	Fuzzers	Generic	Normal	Shellcode	Worms
Original	1.31	1.89	1.73	0.39	**2.30**	2.16	1.39	**1.88**	**3.41**
Normal	0.97	**2.29**	1.79	0.54	0.68	2.19	3.16	0.26	3.29
Min-Max	1.19	1.43	1.36	1.11	0.88	3.27	**3.31**	0.08	1.91
Robust	0.16	0.11	**1.84**	**2.99**	0.85	0.11	3.29	0.05	3.26
Standard	0.62	1.47	1.67	1.11	1.11	3.34	3.05	0.16	2.46
Quantile	**1.43**	1.12	1.02	1.16	2.05	3.33	2.74	0.74	2.80
Power	1.01	1.08	0.97	1.06	1.31	**3.38**	3.29	0.55	2.47

Table 4.9 The Mahalanobis distances between the Shellcode centroid and the rest

	Analysis	Backdoor	Dos	Exploit	Fuzzers	Generic	Normal	Recon	Worms
Original	1.47	1.98	1.55	1.98	0.80	2.86	3.15	**1.88**	3.36
Normal	1.06	**2.42**	**1.92**	0.74	0.78	2.19	**3.40**	0.26	3.23
Min-Max	1.22	1.43	1.37	1.15	0.92	3.35	3.32	0.08	1.93
Robust	0.20	0.16	1.88	**3.03**	0.91	0.07	3.33	0.05	3.27
Standard	0.72	1.49	1.75	1.27	1.11	3.45	3.19	0.16	2.58
Quantile	**2.07**	1.34	1.37	1.78	**2.19**	**3.82**	2.88	0.74	**3.47**
Power	1.42	1.04	0.96	1.45	1.79	3.72	3.20	0.55	2.94

Table 4.10 The Mahalanobis distance between the Worms centroid and the rest

	Analysis	Backdoor	Dos	Exploit	Fuzzers	Generic	Normal	Recon	Shellcode
Original	3.15	3.19	**3.35**	3.39	**3.74**	3.69	3.67	**3.41**	3.36
Normal	**3.31**	3.40	3.34	3.37	3.40	3.58	4.08	3.29	3.23
Min-Max	2.85	3.32	3.21	1.79	1.07	3.70	3.81	1.91	1.93
Robust	3.25	3.25	3.24	**3.97**	3.26	3.22	**4.20**	3.26	3.27
Standard	2.64	**3.50**	3.31	1.54	2.16	3.71	2.72	2.46	2.58
Quantile	2.53	**3.50**	3.27	2.01	3.39	**3.89**	3.68	2.80	**3.47**
Power	2.47	3.39	3.26	1.70	1.40	3.77	4.01	2.47	2.94

To address the overlapping problem, we will identify and use that transformation algorithm which maximizes the intra-cluster distances. According to Table 4.7, the min-max scaler increases the distance of the centroids of four attack classes (Backdoor, DoS, Generic, and Reconnaissance) away from the Normal class centroid more than Normalization, Robust Scaler, Standardization, Quantile and Power transformation methods. Consequently, min-max scalar will be employed in our study.

To further support our findings that were derived based on observations using Tables 4.1 through 4.10, we also implement normalization, min-max, robust, standard, quantile

and power scalers in conjunction with Balanced Bagging, XGBoost and Random Forest classifiers. We assess the performance of the classifiers in terms of accuracy in Table 4.11.

Table 4.11 Accuracy comparison among the performances of six different scalers

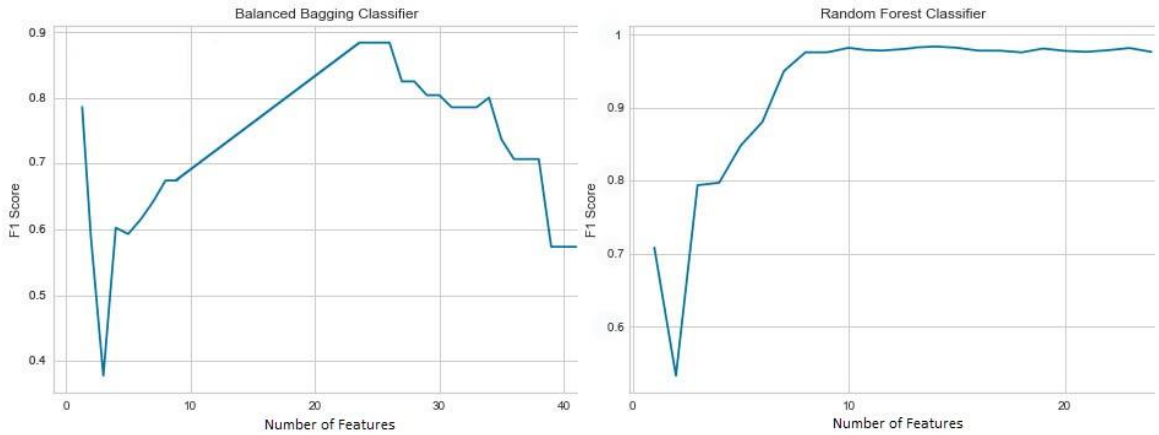
	Balanced Bagging (%)	XGBoost (%)	Random Forest (%)
Original Dataset	87.56	94.11	92.81
Normalizer	81.45	85.02	85.78
Min-Max Scaler	**88.91**	**96.40**	**93.33**
Robust Scaler	87.61	94.55	93.12
Standard Scaler	86.64	92.73	91.99
Quantile Transformer	87.32	93.95	93.12
Power Transformer	86.11	92.00	91.27

Increasing the distance of the attack centroids against the Normal class centroid is realized by comparing the performance of the classifiers with and without applying six different scalers. The first row of Table 4.11 demonstrates the accuracies of three estimators where no normalization strategy is used. Comparing the results in this row with the results of the Min-max scaler shows that the accuracies increased from 1 to 2 percent for each estimator when the min-max scaler applied on the dataset. In this table, the worst accuracies belong to the normalizer and the best ones are achieved by Min-Max scaler. Accordingly, the choice of the Min-Max scaler is appropriate, which will be employed prior to implementing the feature selection algorithm and classifier training phase.

4.3 Feature Selection

Fourteen different feature selection algorithms were implemented on the dataset to extract the informative as well as representative features. Chi-squared [163], Information gain (tree-based feature selection [164]), CFS (Appendix A), ReliefF [165], and mRMR [166], are employed among filter-based feature selectors; genetic algorithm (Appendix A), Recursive Feature Elimination (RFE) [167], and Sequential Feature Selection (forward

selection and backward elimination, Appendix A) are picked as wrapper methods; and Lasso and Elastic Net are chosen among embedded feature selectors. The effect of feature selection algorithms on the classifier performance was assessed and evaluated using ensemble algorithms including Random Forest [168], Bagging [169], Balanced Bagging (BB) [170], AdaBoost [171], XGBoost [172], Gradient Tree [172], Extremely Randomized Trees (ERT) [173], Easy Ensemble (EE) [174] and many other algorithms such as naïve Bayes [175], SVM and MLP Neural Network [176] using Back Propagation (BP) optimization algorithm. The code is deployed and implemented in Spyder v3.3.2 [177], written in Python v3.6.8 64-bit [178]. Pandas v1.0.1 [179], Numpy v1.17.0 [180], math v3.8 [181], matplotlib v3.2.1 [182], scipy v1.2.3 [183], seaborn v0.10.1 [184], Scikit-learn v0.22.2 [185], imblearn v0.6.2 [186] are the rest of the Python modules, packages and libraries employed for this study. Performance evaluation of classifiers on the dataset which was preprocessed with the set of feature selection algorithms indicated that the two best feature selection algorithms are Elastic Net and Sequential Forward Selection (SFS) running in conjunction with Random Forest, Bagging and XGBoost classifiers. The combination of Elastic Net feature selection algorithm with the Balanced Bagging machine learning classifier, and the SFS feature selector with the Random Forest classifier and XGboost performed the best among all possible combinations tested when considering the F1-score as the metric. Elastic Net feature selector in conjunction with the Balanced Bagging classifier performed the best for 24 features that were listed in Table 4.12 as shown in Figure 4.5 (a). The SFS feature selection algorithm in conjunction with the Random Forest classifer performed the best for 8 features which form a proper subset of 24 features as shown in Figure 4.5 (b).



4.5 (a) 4.5 (b)

Figure *4-5* (a) The F1-score of Balanced Bagging classifier across the number of input features.(b) The F1-score of Random Forest classifier across the number of input features

In the case of intrusion detection, decreasing the number of attack records that are incorrectly identified as normal (FN) matters a lot in comparison with reducing the number of normal records that are incorrectly identified as attack (FP). The concept of weighted harmonic mean of precision and sensitivity leveraged by the F1-score explains that an effective models considers a false negative β^2 times more costly than a false positive. In this context, F1-score is an effective alternative for accuracy. Also, it takes both false negative (FN) and false positive (FP) into consideration where the class distribution is imbalanced. As an example, less number of false positives and false negatives will generate a smaller F1 score, while a naïve classifier that is biased with the majority observation could yield a high accuracy score. Accordingly, F1-score is an appropriate metric to evaluate the models that are implemented on this dataset.

Table 4.12 List of features selected by Elastic Net and SFS

Feature No	Feature Name	Elastic Net	SFS	Description
1	dur	X		Record total duration
2	proto	X	X	Transaction protocol
3	service	X	X	Contains the network services
4	state	X		Contains the state and its dependent protocol
5	spkts	X		Source to destination packet count

Feature No	Feature Name	Elastic Net	SFS	Description
6	dpkts	X		Destination to source packet count
7	sbytes	X	X	Source to destination transaction bytes
8	dbytes	X		Destination to source transaction bytes
9	rate	X	X	Ethernet data rates transmitted and received
10	sttl	X		Source to destination time to live value
11	dttl	X		Destination to source time to live value
12	sload	X		Source bits per second
13	dload	X	X	Destination bits per second
14	sloss	X		Source packets retransmitted or dropped
15	dloss	X		Destination packets retransmitted or dropped
16	sinpkt	X		Source interpacket arrival time (mSec)
17	dinpkt	X		Destination interpacket arrival time (mSec)
18	sjit	X	X	Source jitter (mSec)
19	djit	X	X	Destination jitter (mSec)
20	swin	X		Source TCP window advertisement value
21	stcpb	X		Destination TCP window advertisement value
22	dtcpb	X		Destination TCP base sequence number
23	dwin	X		Destination TCP window advertisement value
24	tcprtt	X	X	TCP connection setup round-trip time

Chapter 5

5 Proposed Methodology

5.1 Classifier development methodology

In this research the training and testing subsamples existing on the UNSW website are used. There are 175,341 records in training and 82,332 records in testing data subsets each of which contains the records belonging to 9 different attack classes consisting of Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, Worms as well as the Normal class. Exploits, Generic and Normal instances make up a large portion of subsamples at 16%, 22% and 38% of the overall records in the training subset, respectively. As an extreme case, Worms attack instances form a mere 0.06% of both training and test sets. Since the number of records of majority classes significantly outnumber the records of minority classes, there exists a severe class imbalance problem, as discussed early and shown in Figure 3.1. Also, in this dataset we discovered the presence of the overlapping problem between the Normal class and some attack classes, and among attack classes,

shown in Figures 3.2 through 3.7. In this section, we develop a design to address these problems.

We implemented numerous machine learning classifiers such as SVM, naïve Bayes (NB), multi-layer perceptron (MLP) neural network (NN), Bagging, Random Forest (RF), Extremely Randomized Trees (ERT), AdaBoost, Gradient Tree (GT), Balanced Bagging (BB), XGBoost, and Easy Ensemble (EE) on the dataset for the case where the classifier algorithms employed all of the original features without implementing data normalization methods. The results are shown in Table 14 in terms of missed alarm rate (MAR) which is one of the, if not, most critical performance metrics for intrusion detection context. Table 5.1 shows that Balanced Bagging, XGBoost and Random Forest lead in their performances for this dataset with respect to the MAR metric. Consequently, these three classifiers will be employed in the design of an ensemble classifier.

Table 5.1 Missed alarm rate values for the set of 11 classifiers

Classes	**SVM**	**NB**	**Bagging**	**NN**	**RF**	**ERT**	**AdaBoost**	**GT**	**BB**	**XGBoost**	**EE**
Analysis	1.00	1.00	0.99	1.00	0.86	1.00	0.87	1.00	**0.77**	0.87	0.86
Backdoor	1.00	1.00	0.93	0.98	0.76	0.95	0.93	0.95	**0.59**	0.64	0.60
DoS	0.90	0.99	0.88	0.98	0.87	0.87	0.99	0.93	**0.81**	0.83	0.99
Exploits	0.90	0.97	0.21	0.82	0.21	0.28	0.70	0.09	0.42	**0.04**	0.99
Fuzzers	0.94	0.63	0.42	0.89	0.39	0.42	0.93	0.55	0.32	**0.29**	0.75
Generic	0.51	0.03	0.03	0.03	0.03	0.03	0.42	0.03	0.04	**0.02**	0.42
Normal	0.04	0.65	0.24	0.24	**0.21**	0.23	0.86	0.34	0.34	0.39	0.70
Recon	0.65	0.71	0.19	1.00	**0.17**	0.22	**0.17**	0.21	**0.17**	0.18	0.99
Shellcode	0.96	0.98	0.31	1.00	0.30	0.52	0.92	0.60	**0.06**	0.12	0.81
Worms	1.00	0.95	0.86	1.00	**0.04**	0.86	1.00	0.56	0.09	0.43	0.88

We also employ the Hellinger distance criterion [151, 187] along with the Random Forest classifier to improve the quality of split. Decision Trees are easy to code, interpretable, fast, and nonlinear. However, they suffer from overfitting, axis-parallel splitting and skewness sensitivity. The overfitting problem is mitigated by tree pruning, while axis-parallel splitting can be addressed by building a forest of orthogonal [188] and

oblique decision trees [154]. Skewness sensitivity of decision trees arises due to utilizing some popular splitting criteria including information gain and Gini measure. Hellinger distance measure can address this problem due to its skew insensitivity property. For instance, suppose that we have two classes and Random Forest is applied on the training subset with 175,341 records. Also, in this scenario, we have 1% of the entire data in class 'A' and the remaining records are in class 'B'. In the case that Random Forest splits the data on the feature 'rate' using a test or threshold value of 200,000, one splitting scenario for such an imbalanced dataset could be as presented in Figure 5.1.

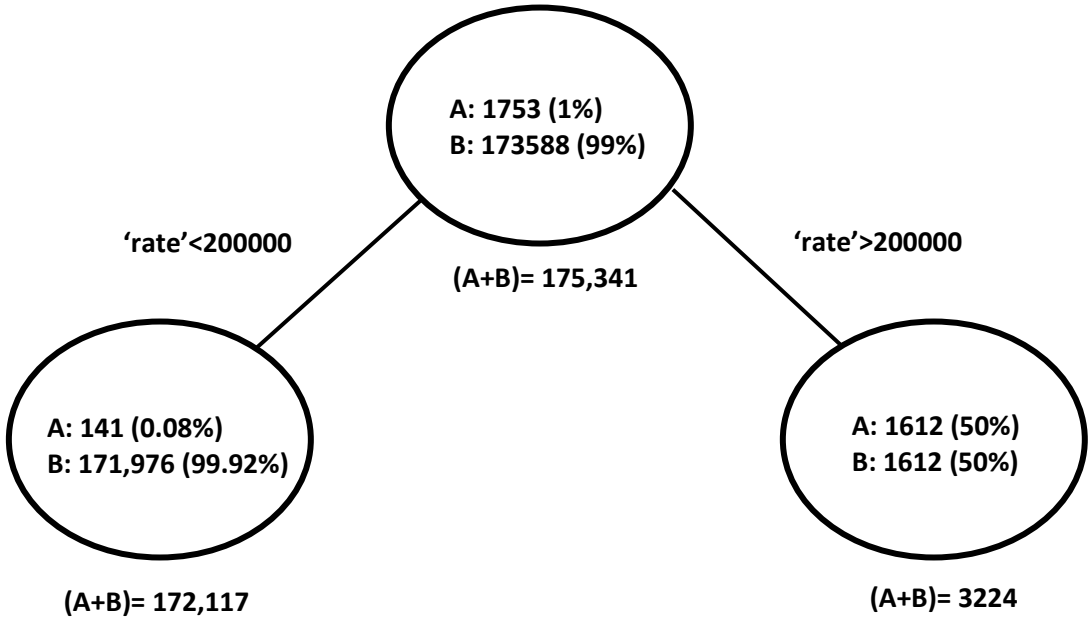


Figure *5-1* An example to illustrate Hellinger distance metric utility

In fact, regardless of balanced or imbalanced property of a dataset, the best split is made for binary classification when the entire data points in class 'A' are placed in the left node and all data points in class 'B' are placed in the right node. If that is the case, the perfect score for Entropy and Hellinger distance would be 1.0 and $\sqrt{2}$, respectively. In this example, which is a reasonably good but not a perfect split, the scores measured by Entropy and Hellinger are tabulated as follows,

Table 5.2 Entropy and Hellinger distance score measurements

	Entropy	**Hellinger**
Perfect Score	1.0	$\sqrt{2} \simeq 1.41$
Evaluated Score	0.015	1.29

The score obtained by Hellinger distance is calculated using Equation 2.16. According to this score, Hellinger distance takes this split into account to form the final prediction. In contrast, given the Entropy formula as

$$\mathrm{E} = -\sum_{\mathrm{i}=1}^{\mathrm{c}} \mathrm{p_i} \log_2 \mathrm{p_i} \tag{5.1}$$

and the Information Gain formula as

$$\mathrm{Gain} = \mathrm{E(parent)} - \frac{\mathrm{w_L}}{\mathrm{w_{parent}}}\mathrm{E_L} - \frac{\mathrm{w_R}}{\mathrm{w_{parent}}}\mathrm{E_R} \tag{5.2}$$

The split in Figure 5.1 does not appear to be "desirable". In these equations, $\mathrm{p_i}$ is the probability distribution associated with class i, $i = \{1,2,\dots,c\}$, where c is the number of classes. Information gain is the difference of Entropy in parent node ($\mathrm{E(parent)}$) from the Entropy of its left ($\mathrm{E_L}$) and right ($\mathrm{E_R}$) child where $\mathrm{w_L}$ is the number of data points in the left node, $\mathrm{w_R}$ is the number of data points in the right node, and $\mathrm{w_{parent}}$ is the number of data points in the parent node.

For this split, it looks very probable that a record coming into the right node will be misclassified. This misclassification probability is $\frac{3224}{175341}$ (2% of the entire data), while 98% of the data will be most probably classified correctly if they find their way into the left node. Accordingly, it can be considered as a good split. However, the information

gained by Entropy measurement indicates that this split is a very bad one since the score is 1.5% of the perfect score shown in Table 5.2. All the while, the Hellinger distance metric values this split by assigning it 91.49% of the perfect or maximum achievable score. Hellinger distance metric thus addressed, for the most part, the problem of skewness sensitivity if utilized by a decision tree classifier and will likely improve the performance of such an algorithm.

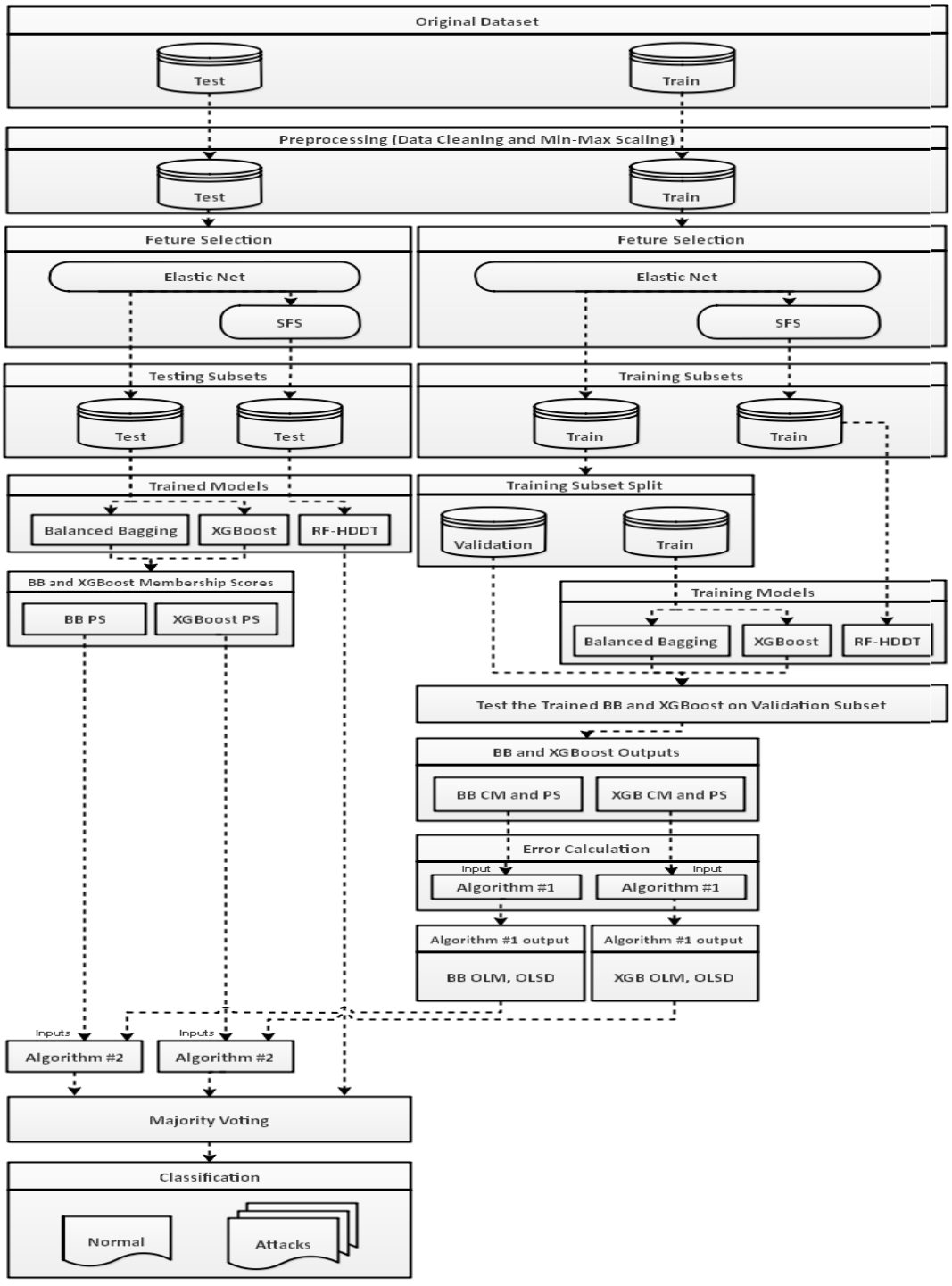


Figure *5-2* Classifier development schematic diagram

5.2 Training methodology

The training set is split into two subsets: one subset, aka training subset, is used to train the classifier and the second subset, aka validation subset, is used to calculate its error rate to determine the convergence or stopping point. The training set is split into two subsets using stratified sampling: 90% of the training set is extracted in order to train the proposed model and the remaining 10% is kept to calculate the model's error. Each stratum is formed by ten different classes following a frequency distribution. In other words, the samples are picked randomly from each attack class as well as Normal records. Next, the training subset is processed with the Elastic Net feature selection algorithm. This algorithm selects 24 features out of 40 before the training subset is used to train the Balanced Bagging and XGBoost classifiers. Concurrently, the SFS algorithm selects 8 informative features out of 24 that were already selected among the original 40 by the Elastic Net. The output of SFS, which are 8 features, is used to train the Random Forest classifier.

Each classifier generates a matrix consisting of probability scores. Entries in this matrix are computed by dividing the number of votes for each class by the number of decision trees in each model. For instance, if we had 30 decision trees in Random Forest and 20 of them vote for normal class on a new sample, the probability of Normal class is 0.67 (20/30). The first seven rows of the matrix produced by the Balanced Bagging is shown in Table 5.3. It consists of 10 columns, representing 10 (9 attack plus Normal) classes, and N rows, where N designates the number of data samples in the validation subset. Since the validation subset has 35,068 records representing 10% of the training set, each model generates a probability matrix consisting 35,068 rows. Both the probability matrix and the confusion matrix produced by XGBoost and Balanced Bagging classifiers

are used as the inputs of Algorithm #1. This algorithm is employed to process the XGBoost and Balanced Bagging outputs to compute the errors caused by class overlap issue associated with the dataset.

Table 5.3 First seven rows of a probability score matrix generated by Balanced Bagging classifier

Row Index	Analysis	Backdoor	DoS	Exploit	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
0	0.0083	0.0103	0.0988	0.1506	0.1718	0.0167	0.0750	0.0212	0.4463	0.0003
1	0.0034	0.0044	0.0632	0.1580	0.4279	0.0240	0.2884	0.0118	0.0144	0.0045
2	0.0057	0.0087	0.0641	0.1094	0.5057	0.0141	0.2659	0.0143	0.0111	0.0011
3	0.0062	0.0101	0.0627	0.1301	0.5491	0.0118	0.1968	0.0154	0.0159	0.0017
4	0.0018	0.0036	0.0339	0.0843	0.6023	0.0113	0.2427	0.0079	0.0109	0.0013
5	0.0044	0.0080	0.0827	0.1667	0.5055	0.0165	0.1213	0.0149	0.0774	0.0027
6	0.0034	0.0051	0.0605	0.1625	0.4867	0.0242	0.2271	0.0105	0.0140	0.0061

The output of Algorithm #1 is a nested list in Figure 5.3 that consists of 10 items representing all the existing attack classes along with the Normal class which constitute the Level-1. Each item in Level-1 points to 9 sub-items in Level-2 as well. The corresponding sub-items in Level-2 for each item will not hold the item itself in Level-1. For each sub-item in the Level-2 of the nested list, there is a corresponding two-element list in Level-3. To make it more clear, we can consider this sole nested list as two two-dimensional arrays with the same dimensionality as the confusion matrix (10 by 10) storing mean and standard deviation. In other words, one matrix could hold the mean and another would hold the standard deviation values. Each row and column represents nine different attack classes along with the Normal class with the same order, similar to the rows and columns of the confusion matrix. These arrays store zeros along their main diagonal. The reason for that is that the aim of Algorithm #1 along with Algorithm #2 is to find the prediction errors or the errors existing in the membership scores. Since the main diagonal is holding true positives in confusion matrix, there is no error to calculate. That is why in the nested list, these entries are eliminated automatically.

Investigating the confusion matrix using Algorithm #1, if class A is misclassified x different times as class B, this algorithm iterates x times to calculate the difference between the probability score of class B and class A as well as class B and the eight remaining classes. If the former difference is smaller than the latter, this difference value (between class B and class A) is stored in a temporary variable (array D) for further calculation. Otherwise, the value is discarded. In the next step, the mean and standard deviation of the stored values are calculated and kept in arrays M and SD, respectively. These two values are placed in the third level of the nested list, which is an output of Algorithm #1, where the class A is the element of the first level of the list and class B is the element of the second level of the list.

Algorithm #1 – Compute Means and Standard Deviations

Mean-Standard-Deviation(CM, PS)
in:
two-dimensional array $\mathrm{CM}_{10\times10}$ holding the confusion matrix
two-dimensional array $\mathrm{PS}_{n\times10}$ holding the membership scores, n = the number of samples of validation subset
out:
two-dimensional array $\mathrm{OLM}_{10\times10}$ initialized with zero
two-dimensional array $\mathrm{OLSD}_{10\times10}$ initialized with zero
local:
empty array DL representing the minimum value of the membership scores difference
empty variable SD representing the computed Standard Deviation value
empty variable μ representing the computed Mean value
empty variable D representing the value obtained by subtracting the membership scores
constant:
array $AL = \{Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Normal, Recon, Shellcode, Worms\}$

```
1: for all x ∈ AL do                          ▷ AL is an ordered list
2:     for all y ∈ (AL − x) do
3:         if CM_x,y > 0 then
4:             j ← 0
5:             for i ← 1 … CM_x,y do
6:                 D ← PS_i,y − PS_i,x        ▷ where x is misclassified as y
7:                 for all z ∈ (AL − (x, y)) do
8:                     DT ← PS_i,y − PS_i,z
9:                     if DT < D then
10:                        count ← 1
11:                    end if
12:                end for
```

Algorithm #1 – Compute Means and Standard Deviations

```
13:                 if count = 0 then
14:                     DL_j ← D
15                      j ← j + 1
16:                 end if
17:             end for
18:             sum ← 0
19:             for h ← 1 … length(DL) do
20:                 sum ← sum + DL_h
21:             end for
22:             μ ← sum / N                          ▷ where N indicates the number of elements in DL
23:             SD ← sqrt( (1/N) Σ_{i=1}^{N} (DL_i − μ)^2 )
24:             OLM_{x,y} ← μ
25:             OLSD_{x,y} ← SD
26:         end if
27:     end for
28: end for
29: return OLM, OLSD
```

To clarify how Algorithm #1 works, we trace its application step by step next. The first row of Table 5.4 (a) represents a confusion matrix for the Analysis attack and Table 5.4 (b) represents the probability scores generated by the Balanced Bagging classifier in a two-dimensional array or matrix form after it is trained and its performance evaluated on the validation subset. Values of these matrices are held by CM and PS, two-dimensional array variables in the Algorithm #1, respectively. Initially, DL, OLM and OLSD are empty lists and eventually holding values for distances, output for mean values, and output for standard deviation values, respectively. AL is another list that initially contains the Normal and all the attack classes.

Now, let's consider the initialization steps in lines 1 through line 3 for the very first iteration of the algorithm:

- Attack class Analysis is removed from the attack list held in AL and maintained in variable x as shown in line 1.

- In line 2, Backdoor attack class is placed in variable y.

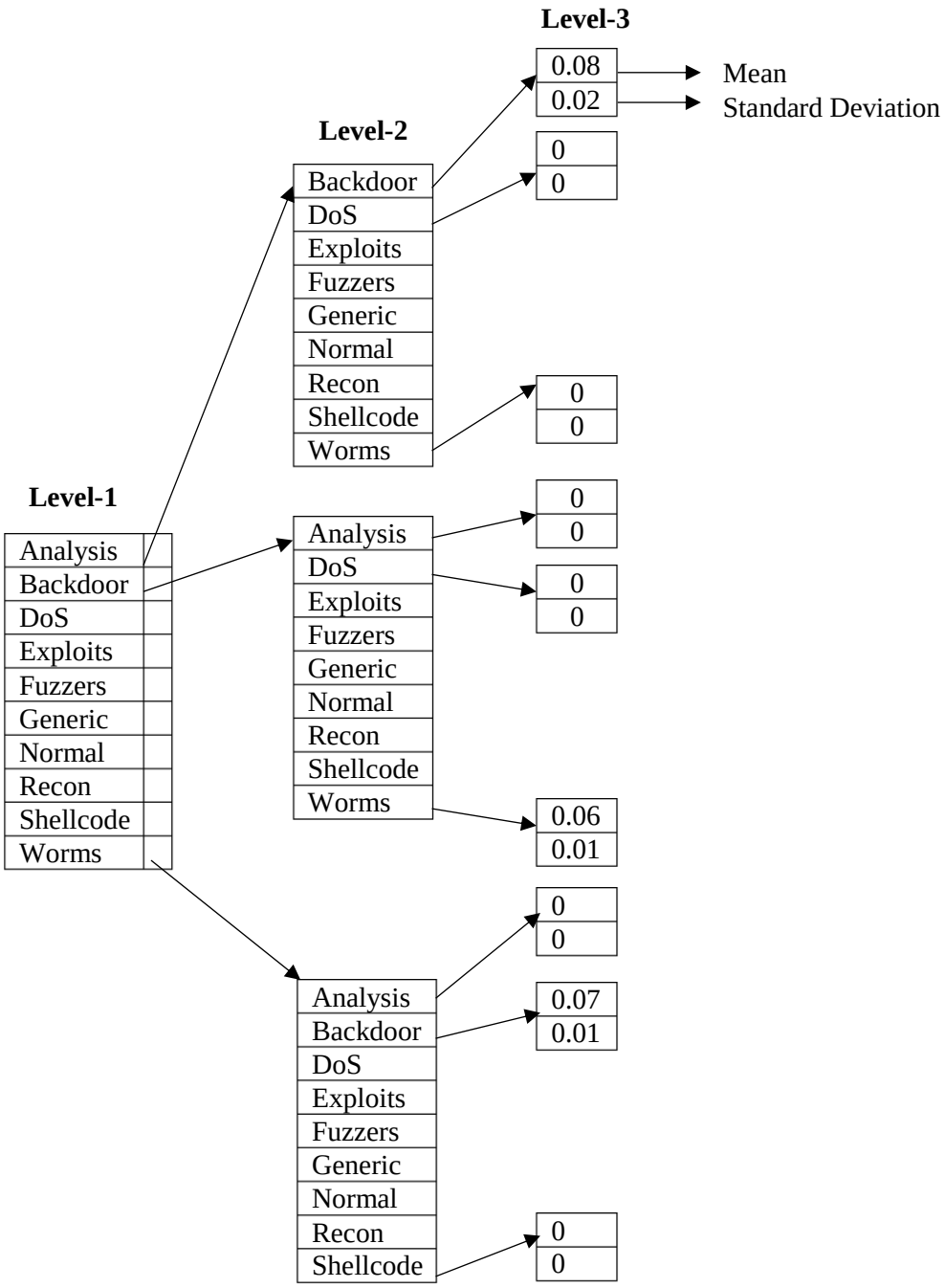


Figure 5-3 Illustration of partial output by Algorithm #1

- To fill out the elements of both mean and standard deviation arrays, the algorithm, in line 3, counts the false negatives where Analysis is the target and Backdoor is the attack class that Analysis is misclassified as. For this example, 124 records of Analysis are misclassified as Backdoor. Since the false negative count is more than zero for Backdoor attack class, the difference between its prediction scores and the Analysis scores are calculated for the corresponding data points. For Analysis and Backdoor we have 124 rows and 10 columns of probability scores generated by the Balanced Bagging model. From the entire 35,068 elements existing in the probability score matrix, 124 element values are extracted where the Analysis attack records are misclassified as Backdoor attack records.

Execution now reaches the for-loop in line 4 during the first iteration.

- In line 4, the algorithm starts from the first row of the probability score matrix in which 0.78, 0.81, 0.02, 0.01, 0.24, 0.08, 0.11, 0.19, 0.08, 0.22 are the likelihood of a new Analysis attack sample being classified as Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Normal, Reconnaissance, Shellcode and Worms, respectively.
- In line 5, the differences between the Backdoor score and the score of other attack classes are calculated.
- In line 6, since the score for Analysis (the target class, score: 0.78) which is deducted from the score for Backdoor (incorrectly predicted class, score: 0.81) is less than other differences (0.81-0.78=0.03) < (0.81-0.02= 0.79 and so on), 0.03 is maintained in the distance variable D.

The for-loop in line 4 repeats for 123 more times. In the second iteration, the algorithm accesses the second row of the probability scores matrix with the values of 0.09, 0.54, 0.01, 0.03, 0.41, 0.04, 0.01, 0.06, 0.12 and 0.14.

- In line 5, it calculates the differences of the scores.
- In line 6, since the difference of the Backdoor score and Analysis score (0.54-0.09= 0.45) is not less than the difference of Backdoor score and Fuzzers score (0.54-0.41=0.13), the difference will not be maintained in the distance variable D. This computation repeats for the other 122 data points.
- In lines 10 and 11, the mean and the standard deviation for all those values in distance variable D are calculated and kept in the mean and standard deviation array for the Analysis and in the elements belonging to Backdoor. Now, we have the range of values with which the errors that are caused by data overlap will be minimized. Figure 5.4 shows this range of values, $\text{Mean} - \text{SD} < \text{x} < \text{Mean} + \text{SD}$ where x will be the likelihood of new sample's membership in a class. This value is obtained in test phase.

Table 5.4 (a) Confusion matrix and (b) First two rows of probability scores matrix.

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
Analysis	501	124	0	0	46	0	3	3	0	0
Backdoor	0	302	0	0	0	12	0	0	0	1
DoS	3	0	270	4	0	0	0	0	5	0
Exploits	0	0	0	254	8	0	50	0	0	0
Fuzzers	3	0	0	32	345	0	23	0	0	0
Generic	1	0	0	0	9	138	0	0	0	0
Normal	0	0	0	54	78	0	876	0	0	0
Recon	0	0	0	0	0	0	8	132	0	0
Shellcode	0	0	0	17	0	0	0	1	187	0
Worms	1	0	0	1	0	0	0	0	2	74

(a)

Index	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
1	0.78	0.81	0.02	0.01	0.24	0.08	0.11	0.19	0.08	0.22
2	0.09	0.54	0.01	0.03	0.41	0.04	0.01	0.06	0.12	0.14

(b)

Table 5.5 (a) The confusion matrix, (b) Mean and standard deviation arrays through Algorithm #1 and (c) Mean and standard deviation arrays through Algorithm #2

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
Analysis	501	124	0	0	46	0	3	3	0	0

(a)

Analysis	Mean	0.08	0	0	0.09	0	0.04	1.30	0	0
	SD	0.02	0	0	0.02	0	0.03	0.10	0	0

(b)

Analysis	Mean	0.08	0	0	0	0	0	1.30	0	0
	SD	0.02	0	0	0	0	0	0.10	0	0

(c)

In the next step, we find whether the range of values calculated for each attack class overlap or not. If they overlap, the range with the largest false negative count will be kept and the range with the smallest false negative count will be changed to 0. For example, if the range of values for Backdoor and Analysis, and Backdoor and Worms overlap, and if Analysis data are misclassified as Backdoor for 124 data points and Worms data are misclassified as Backdoor for 2 data points, the Mean and SD calculated for Backdoor and Analysis will be kept in the list, while the Mean and SD will be changed to zero where Worms attack records are misclassified as Backdoor.

In other words, assume x refers to a range of values between $[\text{Mean} - \text{SD} , \text{Mean} + \text{SD}]$ where the records in class A are misclassified as class B, and y indicates a range of values between $\text{Mean} - \text{SD}$ and $\text{Mean} + \text{SD}$ where the records in class C are incorrectly identified as class B;

IF Mean for y is between the $(\text{Mean} - \text{SD})$ and $(\text{Mean} + \text{SD})$ for x;

or

IF (Mean + SD) for y is less than or equal to (Mean + SD) for x and (Mean − SD) for y is greater than or equal to (Mean − SD) for x;

or

IF (Mean − SD) for y is greater than or equal to (Mean − SD) for x and the (Mean + SD) for y is less than or equal to (Mean + SD) for x;

THEN

Ranges represented by x and y overlap.

In this case, the values of Mean and SD for y will remain unchanged if and only if the number of records in class A that are incorrectly classified as class B is greater than the number of records in class C that are misidentified as class B and the values of Mean and SD for x will be converted to zero.

Figure 5.4 shows that the Fuzzers, Backdoor, and Normal overlap. In this figure, Fuzzers represents a range of value between 0.09 -0.02= 0.07 and 0.09 +0.02= 0.11, Backdoor has a range of values between 0.08 -0.02= 0.06 and 0.08 +0.02= 0.1, and Normal is associated with a range of values between 0.04 -0.03= 0.01 and 0.04+ 0.03= 0.07 where Analysis is incorrectly predicted as Fuzzers, Backdoor, and Normal, respectively. These values are taken from Table 5.5 (b) and after finding the overlaps, Table 5.5 (c) would be the final Mean and SD values generated for all the classes in preparation for the test phase. In Table 5.5 (c) the Mean and SD values for Recon remains unchanged since its error range is from 1.3-0.1= 1.2 to 1.3+0.1=1.4 and does not have any overlap with other ranges. Since

Fuzzers, Backdoor, and Normal ranges overlap as well as the greater number of Analysis records are misclassified as Backdoor, Mean and SD values calculated for Backdoor remain unaltered and the Mean and SD values associated with Fuzzers and Normal are changed to zero.

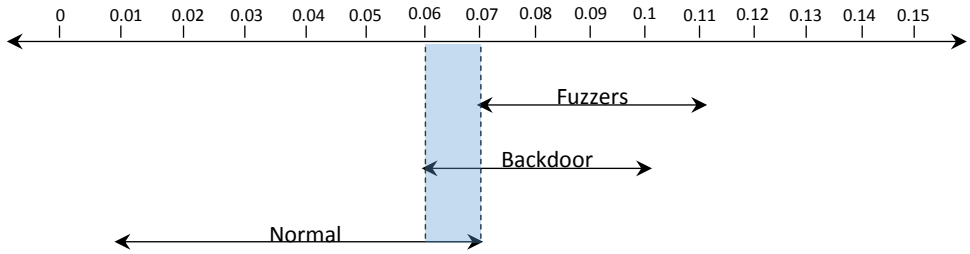


Figure *5-4* An example to show how Algorithm 1 calculates the prediction error range

The process of finding the range overlaps and addressing them takes place for all of 10 classes. This procedure gives us a list of Means and SDs that is shown in Figure 5.3. The list is eventually utilized in the test phase to minimize the prediction errors caused by data overlap.

5.3 Testing methodology

In this phase, Mean and SD values computed by Algorithm #1 and revised after finding the overlaps are used to reduce the errors made by XGBoost and Balanced Bagging classifiers in identifying unseen samples. Due to data overlap issue associated with UNSW-NB15, classifiers may tend to predict class membership for certain new samples incorrectly. A new sample mimicking the behavior of data points belonging to another attack class is most likely to be misclassified. The probability scores matrix generated by the classifiers for a new sample in class ‘A’ contains error if this sample is incorrectly classified as class ‘B’. In this case, the probability matrix score for class ‘A’ is lower than that of class ‘B’, while it must be just the opposite. We consider the difference between the

probability scores for classes ‘A’ and ‘B’ as error. To reduce this error, we apply Algorithm #2 on the probability score matrices generated through XGBoost and Balanced Bagging classifiers.

Algorithm #2 Membership Score Modification

Membership-Score-Modification(PS, OLM, OLSD)

in:

two-dimensional array $PS_{n\times 10}$ holding the membership scores, n = the number of samples of test subset

two-dimensional array $OLM_{10\times 10}$

two-dimensional array $OLSD_{10\times 10}$

out:

two-dimensional array $PS_{n\times 10}$ holding the (modified) membership scores, n = the number of samples of test subset

1: **for** i ← 1 … n **do**
2: max ← 0 ▷ holding zero in max to find the maximum membership score from line 3 to 8
3: **for** j ← 1 … 10 **do**
4: **if** $PS_{i,j} > \max$ **then**
5: max ← $PS_{i,j}$
6: index_max ← j
7: **end if**
8: **end for**
9: min ← 10^{10} ▷ holding a very big constant value in min to find the minimum values obtaining from line 10 to 15
10: **for** j ← 1 **…** 10 **do**
11: **if** $\left((\max - PS_{i,j}) < \min\right)$ **and** (index_max ≠ j) **then**
12: min ← max − $PS_{i,j}$
13: index_min ← j
14: **end if**
15: **end for**
16: **if** $(\min \geq (OLM_{index\_min,index\_max} - OLSD_{index\_min,index\_max}))$ **and** $(\min \leq (OLM_{index\_min,index\_max} + OLSD_{index\_min,index\_max}))$ **then**
17: $PS_{i,index\_min} \leftarrow (PS_{i,index\_min} + OLM_{index\_min,index\_max})$
18: **end if**
19: **end for**
20: **return** PS

Since only the test set is utilized to evaluate the performance of our model, it is definitely unknown to the model. So, the model does not know the real target variable and we cannot calculate the errors using ground truth. This algorithm is designed to go through the membership scores generated by XGBoost and Balanced Bagging classifiers for the

test subset in order to calculate the errors regardless of real target variables. The following steps discuss the functionality of Algorithm #2:

- Lines 3 to 8, the maximum (class) membership score using the probability scores matrix for each testing sample is found among ten different scores generated by the estimator such as XGBoost and Balanced Bagging. This membership score is stored in the temporary variable max and its index is kept by the temporary variable index_max.
- The algorithm calculates the difference values of max with nine other membership scores in lines 10 through 15.It computes the differences of probability score matrix values for other classes residing across the columns of probability scores matrix with the variable max which represents the maximum class membership score computed in lines 3 through 8. It picks the smallest such difference to store in variables min and index_min the difference value and class membership, respectively. These lines represent the computations to find the membership score corresponding to a class which is the second probable class.
- In lines 16 and 17, If min is equal to and greater than the difference of the corresponding mean with SD, and if min is equal to and less than the summation of the corresponding mean with SD, then the corresponding probability score matrix value is modified using the associated mean value. Variables index_min and index_max represent the indexes for two matrixes OLM and OLSD generated by Algorithm #1 and refer to the attack classes with the same order as in AL. For instance, $\text{index\_min} = 1$ means Analysis. Line

17 increases the likelihood of a sample becomes a member of the correct class type by adding the corresponding mean value calculated by Algorithm #1.

Using the revised mean and standard deviation values obtained by implementing Algorithm #1, Algorithm #2 is able to realize and correct the errors in the probability matrices where the errors arise from data overlap. Although, errors are not zeroed out using this algorithm, they may be reduced appreciably.

The modified membership scores were used along with the membership scores obtained by the augmented Random Forest in the testing phase to make the final prediction using the majority vote. Using this method, the most voted prediction wins and taken as a final prediction. In other words, if more than two classifiers cast a vote for a particular class, the class will gain the final vote.

Chapter 6

6 Simulation Results

In this section, performance of the proposed design is first evaluated for binary and then multi-class cases on the UNSW-NB15 data set.

6.1 Binary Classification

Table 6.1 shows that 790 attack records are misclassified among the entire 45,332 attack records. It means that less than 2% of the overall attacks are misclassified as non-attack or Normal which leads to 0.017 missed alarm rate. In other words, the sensitivity for the attack class is 98.26%. On the other hand, 1022 of 37,000 Normal records are incorrectly classified as attacks which indicates less than 3% false alarm rate. Several evaluation metric values are shown in Table 6.2 in order to comprehensively assess the performance of the proposed classifier design.

Table 6.1 Confusion Matrix on test data subset for binary classification

Predicted / Actual	Normal	Attack
Normal	35978	1022
Attack	790	44542

Table 6.2 Performance evaluation of the proposed model for binary classification

Metrics / Class Type	Sensitivity (%)	Specificity (%)	FPR	FNR	Precision (%)	F-measure (%)
Normal	97.24	98.26	0.017	0.028	97.85	97.75
Attack	98.26	97.24	0.028	0.017	97.76	98.01

The main objective of any intrusion detection system (IDS) is to identify the pattern of the network traffic that may imply a suspicious activity. Accordingly, the performance of proposed IDS on UNSW-NB15 data is competitive in comparison with the other studies reported in the literature as shown in Table 6.3. Two columns are dedicated to present the performance of the proposed classifier design. The first column indicates the average of calculated metrics in Table 6.2 for both the attack and Normal records. This column is used to compare the proposed model with [140]. On the other hand, the second column shows the values of performance metrics associated with the attack class in Table 6.2, which suggests that the performance of the proposed classifier design.

Table 6.3 Comparison of the proposed classifier design with four others cited (NR indicates Not Reported)

Metrics	Proposed Design (Average)	Proposed Design	[137] (Average)	[138]	[139]	[140]
Sensitivity (%)	97.75	98.26	79.12	85.00	NR	98.47
FNR (%)	02.25	01.74	NR	15.00	NR	NR
FPR (%)	02.25	02.76	NR	02.00	08.60	02.18
Precision (%)	97.81	97.76	NR	99.00	NR	NR
F-measure (%)	97.88	98.01	77.87	91.00	NR	NR
Accuracy (%)	97.80	97.80	NR	89.00	91.31	94.11

Kumar et al. [189] developed the unified intrusion detection system (UIDS) by generating the new training and test subsets out of UNSW-NB15. They utilized the k-means clustering algorithm to increase the attack sensitivity as the k-means clustering

algorithm was able to identify the similarities between different attack classes. In each cluster, the number of records in some type of attack classes was more than the rest. The authors randomly picked 65% of the records of the dominating class categories to form a training dataset. The remaining 35% of the instances were used to build the test set. They also used information gain algorithm for the feature selection phase. 13 features out 47 were selected due to the improvement of accuracy scores by C5, Chi-Squared Automatic Inference Detection (CHAID), Classification and Regression Tree (CART) is also known as Decision Tree (DT) and Quick Unbiased Efficient Statistical Tree (QUEST) algorithms. These algorithms were used to form the proposed UIDS model. Their study reported 77.87% and 79.12% for the average sensitivity and F-measure, respectively. It also offered 3.80% false alarm rate for Normal instances and 86.15% attack sensitivity.

The authors in [190] implemented MLP as an anomaly detection system for binary classification. They employed RFE along with the Random Forest classifier for the purpose of dimensionality reduction. This method selected the top four informative features. The original training and testing subsets proposed by [131], are used for training and evaluating the model. The MLP-based IDS scored 85% for sensitivity and 89% for accuracy on the test subset of UNSW-NB15: 15% of the attack traces and 2% of the normal records were misclassified.

Bayu et al. [191] applied Gradient Boosted Machine (GBM) on three datasets including UNSW-NB15. No feature selection technique was implemented. All 47 features were kept for training and testing phases. The results showed that GBM outperformed four other algorithms, namely RF, Deep Neural Network (DNN), SVM and CART, with the average accuracy value of 93.64% and missed alarm rate of 0.0206 where GBM

performance was evaluated on NSL-KDD, UNSW-NB15 and GPRS datasets. While running GBM on UNSW-NB15 alone provided 95.08% accuracy and 2.97% false alarm rate using 10-fold cross-validation and the accuracy of 91.31% and false alarm rate of 8.60% using the Hold-Out method on the original UNSW-NB15 train and test subsets.

In [192], nominal features were converted to numerical and then the Min-Max normalization method was utilized to scale down the values to the range of 0 to 1. They used 5-fold cross-validation without resampling to generate the test and training subsets. They calculated the average of the sensitivity, false alarm rate and accuracy of 5 folds. The authors utilized SVM by taking advantage of hyper clique property of hypergraph to improve the performance of SVM. This optimization technique implemented the feature selection as well. The SVM algorithm was trained with the entire 47 features and then the results were compared with the case when SVM was trained with the optimal number of feature subsets. The optimal feature subset is not reported. However, the number of optimal features is in the range of 30 to 35. They concluded that the feature selection had significant influence on the proposed model which delivered 98.47% sensitivity and 2.18% false alarm rate. However, 94.11% accuracy and 2.18% false alarm rate suggests a relatively large value for the missed alarm rate, which is not reported.

6.2 Multiclass Classification

In this research, the performance of different estimators were evaluated separately by implementing them on the refined dataset. The results are shown in Table 5.1. We combined the first three estimators produced satisfactory performances to form an ensemble method along with Elastic Net and Sequential Forward Selection while Min-Max scaler had already implemented on the refined dataset. The results of the model evaluation

is shown in Table 6.4 in terms of confusion matrix. Although the outcome of the ensemble method has effectively improved the performance of each classifier contributed in the method, the model still suffered from generating large number of false negatives. In order to cover this issue, we proposed two algorithms utilizing the basic statistic methods such as mean and standard deviation. Comparing Table 6.4 with the performance of the proposed model depicted in detail for the multi-class case in Tables 6.5, represents the significant improves in most classes, such as Normal class to shrink the number of false negatives. This improvement verifies the effectiveness of Algorithm #1 and Algorithm #2. Although we observe the increasing number of false negatives in some elements in confusion matrix, such as Fuzzers incorrectly classified as Backdoor, when the proposed algorithms implemented on the final decision, they are mostly seen between two attack class rather than an attack class and Normal records. On the other words, false negatives in one attack class may be increase due to the attack class to attack class misclassification.

In Table 6.5, the share of attack records which are incorrectly categorized as Normal traces is 4.14% for the 28% overall missed alarm rate. The remaining 23.86% missed alarm rate is associated with misclassification among attack classes which is not equally as problematic for network transactions. Although, 4.14% missed alarm rate for attack records alone could be detrimental for an intrusion detection system, our design achieves better results in comparison with the classifiers in other studies as shown in Table 6.6. Normal traces are also misclassified as Shellcode, Fuzzers and Analysis which suggests approximately 3% false alarm rate and 97.24% sensitivity. This is because these attack types mimic the behavior of Normal records [135, 193, 194]. This is the main reason that some attacks are also incorrectly predicted as Normal activity. In the associated confusion

matrix, we can see that 19.20% of Analysis, 6.69% of Backdoor, 4.35% of DoS, 3.04% of Exploits, 0.33% of Generic, 0.88% of Reconnaissance, 2.38% of Shellcode, and 4.55% of Worms attack records are confused with Normal records.

Table 6.4 Confusion matrix showing the performance of the ensemble method

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
Analysis	327	193	4	1	18	0	130	0	4	0
Backdoor	298	154	8	8	23	0	76	1	15	0
DoS	1477	737	775	435	221	0	228	45	164	7
Exploits	1322	1583	51	6197	257	0	667	490	420	145
Fuzzers	651	386	8	14	1837	0	2604	7	540	15
Generic	15	30	37	390	99	18145	63	7	72	13
Normal	842	1	3	28	1427	0	34545	0	154	0
Recon	97	210	0	35	14	0	31	2967	128	14
Shellcode	11	0	0	0	7	0	9	3	347	1
Worms	0	0	0	1	0	0	1	0	3	39

Table 6.5 Confusion matrix showing the performance of the proposed design

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
Analysis	327	193	4	1	18	0	130	0	4	0
Backdoor	126	405	0	0	13	0	39	0	0	0
DoS	1604	625	1110	228	92	0	178	59	164	29
Exploits	779	830	39	8511	57	0	338	221	213	144
Fuzzers	482	491	4	34	4380	0	0	10	646	15
Generic	15	30	37	390	99	18145	63	7	72	13
Normal	200	0	0	0	668	0	35978	0	154	0
Recon	103	207	0	35	14	0	31	2967	126	13
Shellcode	0	0	0	0	7	0	9	3	358	1
Worms	0	0	0	1	0	0	2	0	5	36

Table 6.6 Performance of the proposed design for multi-class case

Metric / Class	Accuracy	Sensitivity	Specificity	FPR	FNR	Precision	F-measure
Analysis	95.56	48.30	95.95	0.041	0.51	0.15	64.25
Backdoor	96.89	69.47	97.09	0.029	0.31	0.84	80.99
Dos	96.27	27.15	99.89	0.001	0.73	0.42	42.70
Exploits	95.98	76.46	99.03	0.009	0.24	0.84	86.29
Fuzzers	96.78	72.25	98.73	0.012	0.28	0.77	83.44
Generic	99.12	96.15	100.0	0.000	0.28	1.00	98.04
Normal	97.81	97.24	98.26	0.017	0.02	0.97	97.75
Reconnaissance	95.39	84.87	95.86	0.041	0.15	0.61	90.03
Shellcode	99.31	94.71	98.33	0.017	0.05	0.34	96.49
Worms	99.73	81.82	99.74	0.003	0.18	0.24	89.90

Performance comparison of the proposed model with those studies reported in the literature is presented in Tables 6.7 and 6.8. Many of the relevant performance metrics including the missed alarm rate, which is one of the most critical ones, are not reported in these studies by others. Consequently, performance comparison is done only for accuracy and sensitivity as these are the only metrics commonly reported in the cited studies.

In [195], the authors proposed the ensemble extreme learning machine (ELM) along with one-vs-all method to generate multi-class classification model. The proposed algorithm is a combination of a single hidden layer feedforward neural network and a softmax layer to make a multi-class prediction out of an ensemble of single output which was 0 or 1. This algorithm scored 95.66% average accuracy. Also, the authors implemented ExtraTree classifier in order to reduce the dimensionality of feature space. Accordingly, 21 features were selected. In the final stage, weighted extreme learning machine (WELM) was implemented and the accuracy of each attack type increased as shown in Table 6.7. The training was done on 80% of the original training set and the remaining 20% was used for validation to avoid overfitting. The entire UNSW-NB15 test subset was utilized for the test phase.

D. Papamartzivanos et al. [196] combined the decision tree and genetic algorithm to generate classification rules and called their model Dendron. Wrapper technique was used for feature selection, which resulted in 23 being selected as informative features. The authors reported the sensitivity of 97.39% for Normal records and the average false alarm rate of 2.61% as presented in Table 6.8. In this study, 10% of the instances for each of the 9 classes were considered for building the training set and the remaining 90% was kept for the test set. To address the imbalance problem of multi-class classification in UNSW-

NB15, 50% of Worms attack class records were included in the training subset and remaining 50% was kept to test the model.

The integrated rule-based model in [193] is trained to detect five class types to avoid overlapping. These rules were generated from four tree-structured classification algorithms, C5, CHAID, CART and QUEST. Training and test sets were built by eliminating some instances from the original training and testing subsets using k-means clustering. These instances belong to Analysis, Backdoor, Fuzzers, Shellcode and Worms attack types. These attack types suffer from overlapping problem and their presence may cause poor results. For the feature selection phase, the genetic algorithm was used and 22 features were picked accordingly. They reported good accuracy and sensitivity values for the Normal records. Also, the reconnaissance is successfully detected with an accuracy of 99.10% as shown in Table 6.6. However, the average accuracy and sensitivity are 93.94% and 65.21%, respectively.

Table 6.7 The accuracy of proposed model vs. the accuracy of different models (NR: Not Reported)

Attack Type	Accuracy	Accuracy [143]	Accuracy [144]	Accuracy [141]	Difference
Analysis	95.56	99.26	**99.30**	NR	-3.74
Backdoor	96.89	**99.11**	97.93	NR	-2.22
Dos	**96.27**	94.90	95.71	94.52	+0.56
Exploits	**95.98**	90.12	93.58	89.72	+2.40
Fuzzers	**96.78**	91.47	95.04	NR	+1.74
Generic	**99.12**	98.23	98.70	87.70	+0.42
Normal	97.81	93.54	94.59	**98.64**	-0.30
Reconnaissance	95.39	95.33	96.18	**99.10**	-3.71
Shellcode	99.31	**99.40**	98.33	NR	-0.09
Worms	99.73	**99.92**	99.78	NR	-0.14

Table 6.8 The sensitivity of proposed model vs. the sensitivity of different models (NR: Not Reported)

Attack Type	Sensitivity	Sensitivity [144]	Sensitivity [141]	Difference
Analysis	**48.30**	20.45	NR	+27.85
Backdoor	**69.47**	67.32	NR	+02.15
Dos	**27.15**	14.29	5.0	+12.86
Exploits	**76.46**	76.22	54.64	+00.24
Fuzzers	**72.25**	64.42	NR	+07.83
Generic	96.15	81.37	**96.72**	**-00.57**
Normal	97.24	97.39	**98.00**	**-00.76**
Reconnaissance	**84.87**	46.04	71.70	+13.17
Shellcode	**94.71**	36.39	NR	+58.32
Worms	**81.82**	18.37	NR	+63.45

Figure 6.1 depicts the performance of our model in comparison with two models, Integrated and Dendron [193, 196] that are proposed recently in terms of the F-measure. The proposed model in this study outperforms the other two given the F-measure values. The main reason is that the imbalance and overlapping problems in our model are addressed. We have a combination of ensemble methods to handle the imbalance and if-then-else rules to mitigate the adverse effects of overlapping issue and using Hellinger distance criterion to choose the best split considering the imbalance problem.

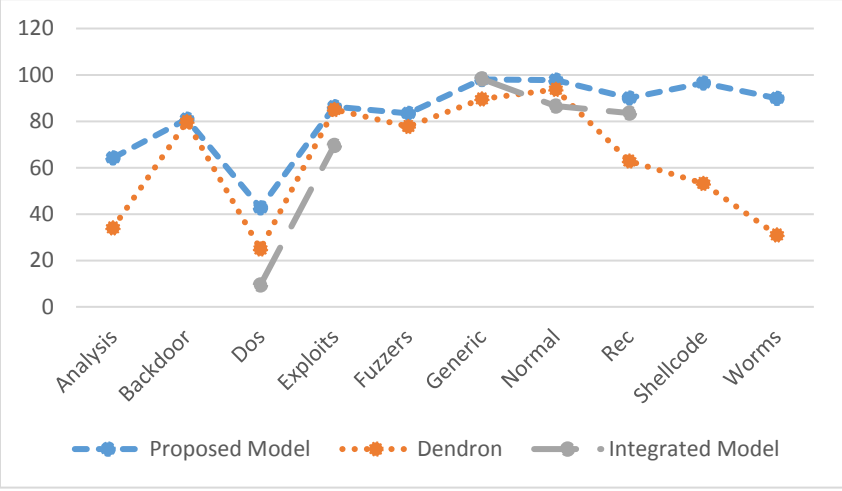


Figure 6-1 F-measure comparison

Chapter 7

7 Conclusions & Future Work

7.1 Conclusion

This study presents design and performance evaluation of an intrusion detection (and identification) system using machine learning for the UNSW-NB15 dataset. We evaluated the performance of classifier design which employs three ensemble classifiers and two proposed algorithms where the latter is developed for minimizing the errors due to one of the twoissues inherent to the UNSW-NB15 dataset, namely the class overlap and class imbalance. To deal with the imbalanced data, we utilized the Balanced Bagging and the XGBoost ensemble classifiers which offer a set of hyper-parameters that, through judicious adjustments of the same, help contribute to improved performance in the presence of the imbalanced data. To address the class overlap issue, we proposed two algorithms and utilized them to process and modify the classification outputs from the Balanced Bagging and the XGBoost ensembles. Outputs of three ensemble classifiers, namely Random Forest, Balanced Bagging and XGBoost, were provided as inputs to a combiner that

implemented majority voting to determine the final class membership of an input data record under test.

Although the literature reports high level of classification performance for the Random Forest (RF) algorithm for a variety of problem domains, its performance on the UNSW-NB15 dataset was somewhat lacking particularly with respect to the missed alarm rate metric. Therefore, in order to address this problem associated with the RF classifier performance, we leveraged the Hellinger distance as a split criterion in the construction of trees rather than the entropy value which is used as the default.

A multistep preprocessing approach was implemented for the UNSW-NB15 dataset. In the first step, we removed a number of features which potentially did not help with the prospect of developing a classifier that would be applicable in a general sense: hence source IP address, source port number, destination IP address, destination port number, record_start_time, and record_last_time were removed from the dataset. The next step entailed empirical assessment of performances of more than ten different classifiers on the dataset with the remaining 42 features. Among those tested, Balanced Bagging, XGBoost, and Random Forest showed the higher performance. We assessed the performances of these classifiers by employing six different normalization methods on the modified UNSW-NB15. The results showed that min-max scaler enhanced the performance of the classifiers in terms of accuracy. Min-max scaler as a normalization method helped increase the distances between the data points of two different attack classes reducing the degree of class overlap.

We evaluated the classifiers with a variety of feature selectors as well. Utilizing Elastic Net and SFS increased the performance of the estimators independently. The Elastic Net feature selection algorithm identified 24 features which were then further processed by the SFS feature selection algorithm to reduce the selection to 8 features. Balanced Bagging and XGBoost performed better with 24 features selected by the Elastic Net while Random Forest performed better with only 8 features selected by the SFS.

The classifiers were trained with 90% of the training subset and the remaining 10% were kept as a validation subset in order to measure the probable classification error using the class membership scores. Since the dataset suffers from the overlap issue, it is then expected that the classifiers will commit classification errors. We calculated the mean and standard deviation of the class membership scores to help reduce the probable misclassification.

This was done by Algorithm #1. This algorithm records the smallest membership score errors made by Balanced Bagging and XGBoost through incorrectly estimating the likelihood of the training samples being the member of a class type. This algorithm calculates the mean and standard deviation of the errors using the validation subset. The calculated values were utilized in Algorithm #2 to minimize the membership errors may occur during the testing phase. In fact, Algorithm #1 estimated the probability of misclassification for each class and recorded them in a list. This list was then engaged in testing phase to avoid misclassification using Algorithm #2.

Application of the combination of preprocessing, feature selectors, tree-based ensemble classifiers, and the proposed algorithms for our design resulted in superior

performance when compared to seven other classifiers, reported in the recent literature, implemented on the UNSW-NB15 dataset for both multi-class and binary classification cases. Performance of the proposed model was compared with both the binary classifiers and multi-class classifiers cited in the literature. In the binary class classification case, our model could classify more than 98% of the attack classes correctly and therefore less than 2% of the attack classes were misclassified. Our model also performed highly for the classification of Normal records. More than 97% of the Normal records were classified correctly with less than 3% false alarm rate. Among all those cited in the literature studies, only one reported the missed alarm rate, which was rather high at 8%.

For the multi-class case, 28% of the attack classes were misclassified: 4.14% of the attack classes were mislabeled as Normal class and 23.86% of the remaining attack classes were misclassified as other attacks where the latter scenario is not as problematic as the former scenario. Also, 97.24% of Normal traces are classified correctly and the remaining 2.76% were misclassified as Shellcode, Fuzzers and Analysis. The main reason behind the Normal class misclassification is the data overlap or mimicry. Almost 100% of Generic and Reconnaissance data points were classified correctly using our proposed model. Also, 19.20% of Analysis, 6.69% of Backdoor, 4.55% of Worms, 4.35% of DoS, 3.04% of Exploits, and 2.38% of Shellcode attack records were incorrectly labeled as Normal records.

In comparison with other studies reported in the current literature on the UNSW-NB15 dataset, our model achieved impressive results. It addresses two major issues that a dataset may suffer from, overlap and imbalance. We employed Balanced Bagging and XGBoost offering a range of hyperparameters in order to address the dataset imbalance. Also, we

utilized the Hellinger distance for the Random Forest for the same reason. We further proposed two new post-processing algorithms for the outputs of training models to minimize the errors caused by the large number of impure nodes generted during the training phase due to the data overlap issue. These algorithms are generic and therefore can be used along with any other machine learners where their base classifiers are decision trees.

7.2 Future work

We plan to use Hellinger distance as split criterion for both Balanced Bagging and XGBoost to enhance the performance of our model. We are also aimed at utilizing our proposed algorithms, known as Algorithm #1 and Algorithm #2, along with Random Forest. This algorithm can minimize the errors of membership scores generating in the test phase. So, it also works with Random Forest classifier. Moreover, we intend to combine binary and multi-class classification with which the Normal and attack classes can be classified with less error and the attack classes can be classified with different multi-class classifiers. The attack classes can be categorized in advance in different classes based on similarity measure. This method can help us find the attacks that mimic their behaviors. We can then separate these attack classes to train our algorithms separately and avoid any confusion that may be caused by data overlap. We can also use fuzzy if-rules to find the similarities between the attack classes and the attack classes and Normal class.

References

[1] V. Zlomislić, K. Fertalj, and V. Sruk, "Denial of service attacks: an overview," in *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, 2014, pp. 1-6.

[2] A. Bendovschi, "Cyber-attacks–trends, patterns and security countermeasures," *Procedia Economics and Finance*, vol. 28, pp. 24-31, 2015.

[3] R. Bhandari, R. Swapnil, T. Vishwa, P. Jaydip, and D. Sagar, "Survey on Cyber Attacks," *International Journal of Computer Applications*, vol. 975, p. 8887.

[4] M. Omar, "A World of Cyber Attacks (A Survey)," 2019.

[5] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 2027-2051, 2016.

[6] A. Mallik, A. Ahsan, M. Shahadat, and J. Tsou, "Man-in-the-middle-attack: Understanding in simple words," *International Journal of Data and Network Science*, vol. 3, pp. 77-92, 2019.

[7] V. Sousa, "A Review on Cyber Attacks and Its Preventive Measures," in *Proceedings of the Digital Privacy and Security Conference*, 2019.

[8] K. M. Jain, M. V. Jain, and J. L. Borade, "A Survey on Man in the Middle Attack," *IJSTE-International J. Sci. Technol. Eng*, vol. 2, pp. 277-280, 2016.

[9] M. Raza, M. Iqbal, M. Sharif, and W. Haider, "A survey of password attacks and comparative analysis on methods for secure authentication," *World Applied Sciences Journal*, vol. 19, pp. 439-444, 2012.

[10] M. Baykara and Z. Z. Gürel, "Detection of phishing attacks," in *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 2018, pp. 1-5.

[11] A. Qamar, A. Karim, and V. Chang, "Mobile malware attacks: Review, taxonomy & future directions," *Future Generation Computer Systems*, vol. 97, pp. 887-909, 2019.

[12] E. G. Dada, J. S. Bassi, Y. J. Hurcha, and A. H. Alkali, "Performance Evaluation of Machine Learning Algorithms for Detection and Prevention of Malware Attacks," *IOSR Journal of Computer Engineering*, vol. 21, pp. 18-27, 2019.

[13] A. K. Pandey, A. K. Tripathi, G. Kapil, V. Singh, M. W. Khan, A. Agrawal, *et al.*, "Trends in Malware Attacks: Identification and Mitigation Strategies," in *Critical Concepts, Standards, and Techniques in Cyber Forensics*, ed: IGI Global, 2020, pp. 47-60.

[14] R. Sharp, "An Introduction to Malware," 2017.

[15] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, "Paybreak: Defense against cryptographic ransomware," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 599-611.

[16] G. O'Gorman and G. McDonald, *Ransomware: A growing menace*: Symantec Corporation, 2012.

[17] M. T. Ahvanooey, Q. Li, M. Rabbani, and A. R. Rajput, "A survey on smartphones security: software vulnerabilities, malware, and attacks," *arXiv preprint arXiv:2001.09406,* 2020.

[18] A. O. Prokofiev, Y. S. Smirnova, and V. A. Surov, "A method to detect Internet of Things botnets," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2018, pp. 105-108.

[19] J. Milosevic, F. Regazzoni, and M. Malek, "Malware threats and solutions for trustworthy mobile systems design," in *Hardware Security and Trust*, ed: Springer, 2017, pp. 149-167.

[20] H. S. Brar and G. Kumar, "Cybercrimes: A proposed taxonomy and challenges," *Journal of Computer Networks and Communications,* vol. 2018, 2018.

[21] R. Tahir, "A study on malware and malware detection techniques," *International Journal of Education and Management Engineering,* vol. 8, p. 20, 2018.

[22] A. K. zDalai and S. K. Jena, "Neutralizing SQL injection attack using server side code modification in web applications," *Security and Communication Networks,* vol. 2017, 2017.

[23] S. Gupta and B. B. Gupta, "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art," *International Journal of System Assurance Engineering and Management,* vol. 8, pp. 512-530, 2017.

[24] R. Lu, X. Lin, T. H. Luan, X. Liang, and X. Shen, "Pseudonym changing at social spots: An effective strategy for location privacy in vanets," *IEEE transactions on vehicular technology,* vol. 61, pp. 86-96, 2011.

[25] J. M. Hatfield, "Social engineering in cybersecurity: The evolution of a concept," *Computers & Security*, vol. 73, pp. 102-113, 2018.

[26] H. Aldawood and G. Skinner, "An Advanced Taxonomy for Social Engineering Attacks," *International Journal of Computer Applications*, vol. 975, p. 8887.

[27] J. M. Biju, N. Gopal, and A. J. Prakash, "CYBER ATTACKS AND ITS DIFFERENT TYPES," 2019.

[28] V. Delgado-Gomes, J. F. Martins, C. Lima, and P. N. Borza, "Smart grid security issues," in *2015 9th International Conference on Compatibility and Power Electronics (CPE)*, 2015, pp. 534-538.

[29] N. Tariq, "Impact of cyberattacks on financial institutions," *Journal of Internet Banking and Commerce*, vol. 23, pp. 1-11, 2018.

[30] A. Bouveret, *Cyber risk for the financial sector: A framework for quantitative assessment*: International Monetary Fund, 2018.

[31] B. B. Gupta, N. A. Arachchilage, and K. E. Psannis, "Defending against phishing attacks: taxonomy of methods, current issues and future directions," *Telecommunication Systems*, vol. 67, pp. 247-267, 2018.

[32] T. S. Reed, "CYBERCRIME AND TECHNOLOGY LOSSES: CLAIMS AND POTENTIAL INSURANCE COVERAGE FOR MODERN CYBER RISKS," *Tort Trial & Insurance Practice Law Journal*, vol. 54, pp. 153-209, 2019.

[33] J. L. Cebula and L. R. Young, "A taxonomy of operational cyber security risks," Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst2010.

[34] E. C. Thompson, *Cybersecurity Incident Response: How to Contain, Eradicate, and Recover from Incidents*: Apress, 2018.

[35] I. Agrafiotis, M. Bada, P. Cornish, S. Creese, M. Goldsmith, E. Ignatuschtschenko, *et al.*, "Cyber harm: concepts, taxonomy and measurement," *Saïd Business School WP*, vol. 23, 2016.

[36] L. F. Sikos and K.-K. R. Choo, *Data science in cybersecurity and cyberthreat intelligence*: Springer, 2020.

[37] M. Wu and Y. B. Moon, "Taxonomy of cross-domain attacks on cybermanufacturing system," *Procedia Computer Science*, vol. 114, pp. 367-374, 2017.

[38] A. E. Elhabashy, L. J. Wells, J. A. Camelio, and W. H. Woodall, "A cyber-physical attack taxonomy for production systems: a quality control perspective," *Journal of Intelligent Manufacturing*, vol. 30, pp. 2489-2504, 2019.

[39] E. Kopp, L. Kaffenberger, and N. Jenkinson, *Cyber risk, market failures, and financial stability*: International Monetary Fund, 2017.

[40] M. Meisner, "Financial consequences of cyber attacks leading to data breaches in healthcare sector," *Copernican Journal of Finance & Accounting*, vol. 6, pp. 63-73, 2017.

[41] E. Ozkaya and M. Aslaner, *Hands-On Cybersecurity for Finance: Identify vulnerabilities and secure your financial services from security breaches*: Packt Publishing Ltd, 2019.

[42] S. Morgan, "Official annual cybercrime report," *Sausalito: Cybersecurity Ventures*, 2019.

[43] C. Ventures. (2020). *Global Cybercrime Damages Predicted To Reach $6 Trillion Annually By 2021*. Available: https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

[44] S. Sibi Chakkaravarthy, D. Sangeetha, M. Venkata Rathnam, K. Srinithi, and V. Vaidehi, "Futuristic cyber-attacks," *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 22, pp. 195-204, 2018.

[45] J. W. Pfeifer, "Preparing for Cyber Incidents with Physical Effects," *The Cyber Defense Review*, vol. 3, pp. 27-34, 2018.

[46] A. Thaduri, M. Aljumaili, R. Kour, and R. Karim, "Cybersecurity for eMaintenance in railway infrastructure: risks and consequences," *International Journal of System Assurance Engineering and Management*, vol. 10, pp. 149-159, 2019.

[47] I. Agrafiotis, J. R. Nurse, M. Goldsmith, S. Creese, and D. Upton, "A taxonomy of cyber-harms: Defining the impacts of cyber-attacks and understanding how they propagate," *Journal of Cybersecurity*, vol. 4, p. tyy006, 2018.

[48] M. Bada and J. R. Nurse, "The social and psychological impact of cyberattacks," in *Emerging Cyber Threats and Cognitive Vulnerabilities*, ed: Elsevier, 2020, pp. 73-92.

[49] M. L. Gross, D. Canetti, and D. R. Vashdi, "The psychological effects of cyber terrorism," *Bulletin of the Atomic Scientists*, vol. 72, pp. 284-291, 2016.

[50] D. Bianchi and O. K. Tosun, "Cyber attacks and stock market activity," *Available at SSRN 3190454*, 2019.

[51] R. Williams, "Cultural safety—what does it mean for our work practice?," *Australian and New Zealand journal of public health*, vol. 23, pp. 213-214, 1999.

[52] R. Anderson, C. Barton, R. Bölme, R. Clayton, C. Ganán, T. Grasso, *et al.*, "Measuring the changing cost of cybercrime," 2019.

[53] C. L. Ryan and J. M. Lewis, *Computer and internet use in the United States: 2015*: US Department of Commerce, Economics and Statistics Administration, US …, 2017.

[54] K. White, "The rise of cybercrime 1970 through 2010. A tour of the conditions that gave rise to cybercrime and the crimes themselves," ed, 2013.

[55] J. X. Li, "Cyber crime and legal countermeasures: A historical analysis," *International Journal of Criminal Justice Sciences*, vol. 12, pp. 196-207, 2017.

[56] InternetLiveStats.com. (2011, May 17, 2020). *Total number of Websites*. Available: https://www.internetlivestats.com/total-number-of-websites/

[57] B. Wible, "A site where hackers are welcome: Using hack-in contests to shape preferences and deter computer crime," *Yale LJ*, vol. 112, p. 1577, 2002.

[58] S. KEMP. (2019, May 17, 2020). *DIGITAL 2019: GLOBAL INTERNET USE ACCELERATES*. Available: https://wearesocial.com/blog/2019/01/digital-2019-global-internet-use-accelerates

[59] Netcraft. (2020, May 29, 2020). *May 2020 Web Server Survey*. Available: https://news.netcraft.com/archives/category/web-server-survey/

[60] fbi.gov. (2020, May 29, 2020). *2019 Internet Crime Report Released*. Available: https://www.fbi.gov/news/stories/2019-internet-crime-report-released-021120

[61] J. Armin, B. Thompson, D. Ariu, G. Giacinto, F. Roli, and P. Kijewski, "2020 cybercrime economic costs: No measure no solution," in *2015 10th International Conference on Availability, Reliability and Security*, 2015, pp. 701-710.

[62] A. Brashares, *Steve Jobs: thinks different*: Twenty-First Century Books, 2001.

[63] M. Gitlin and M. J. Goldstein, *Cyber Attack*: Twenty First Century Books, 2015.

[64] T. C. Greene, "Chapter one: Kevin Mitnick's story," *The Register*, 2003.

[65] B. Middleton, *A history of cyber security attacks: 1980 to present*: CRC Press, 2017.

[66] G. Benford, "Future Tense: Catch me if you can," *coMMunications of the acM*, vol. 54, pp. 112-ff, 2011.

[67] E. Cloner. (1982, May 29, 2020). *Elk Cloner (circa 1982)*. Available: http://www.skrenta.com/cloner/

[68] R. Skrenta, "Elk cloner," ed, 2013.

[69] S. Bosworth and M. E. Kabay, *Computer security handbook*: John Wiley & Sons, 2002.

[70] F. Cohen, "Computer viruses: theory and experiments," *Computers & security*, vol. 6, pp. 22-35, 1987.

[71] P. Warren and M. Streeter, "Cyber alert: How the world is under attack from a new form of crime," 2005.

[72] J. Kremling and A. M. S. Parker, *Cyberspace, cybersecurity, and cybercrime*: SAGE Publications, 2017.

[73] R. Nagpal, "Evolution of cyber Crimes," *Asian School of Cyber laws*, vol. 2, 2008.

[74] G. Dayanandam, T. Rao, D. B. Babu, and S. N. Durga, "DDoS Attacks—Analysis and Prevention," in *Innovations in Computer Science and Engineering*, ed: Springer, 2019, pp. 1-10.

[75] M. D. Cavelty, "Cyber-security," *The routledge handbook of new security studies*, pp. 154-162, 2010.

[76] J. Schwarb, "Evaluation and Analysis of Cyber Security Threats and Their Impact on an Evolving Technological Society," Utica College, 2018.

[77] E. v. Asperen, C. Barker, T. Berners-Lee, R. Cailliau, D. Connolly, P. Dobberstein, *et al.* (1991, June 3, 2020). *World Wide Web*. Available: http://info.cern.ch/hypertext/WWW/TheProject.html

[78] P. Beaucamps, "Advanced polymorphic techniques," *International Journal of Computer Science*, vol. 2, pp. 194-205, 2007.

[79] P. Szor, *The Art of Computer Virus Research and Defense: ART COMP VIRUS RES DEFENSE _p1*: Pearson Education, 2005.

[80] N. Goldmann, N. Goldmann, N. R. S. NRS, and N. Undoubtedly, "The Citibank Affair: A Purely Russian Crime?."

[81] M. Reinikainen, "Computer Viruses," *Computer Science*, 2019.

[82] K. Cesare, "Prosecuting computer virus authors: The need for an adequate and immediate international solution," *Transnat'l Law.*, vol. 14, p. 135, 2001.

[83] V. Arutyunov, "Cloud computing: Its history of development, modern state, and future considerations," *Scientific and Technical Information Processing*, vol. 39, pp. 173-178, 2012.

[84] L. Garber, "Melissa virus creates a new type of threat," *Computer*, pp. 16-19, 1999.

[85] G. Pogrebna and M. Skilton, "Cybersecurity Threats: Past and Present," in *Navigating New Cyber Risks*, ed: Springer, 2019, pp. 13-29.

[86] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud computing: An overview," in *IEEE International Conference on Cloud Computing*, 2009, pp. 626-631.

[87] R. S. D. M. Purohit, "Cyber Attacks That Shook the World."

[88] M. S. Asish and R. Aishwarya, "Cyber Security at a Glance," in *2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, 2019, pp. 240-245.

[89] T. Chen, "Phillimon Mwape Mumba," Department of Computer Science, Swansea University, 2012.

[90] S. Specht and R. Lee, "Taxonomies of distributed denial of service networks, attacks, tools and countermeasures," *CEL2003-03, Princeton University, Princeton, NJ, USA*, 2003.

[91] A. Verma, M. Arif, and M. S. Husain, "Analysis of DDoS attack detection and prevention in cloud environment: A review," *International Journal of Advanced Research in Computer Science*, vol. 9, p. 107, 2018.

[92] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," *IEEE Security & Privacy*, vol. 1, pp. 33-39, 2003.

[93] A. Sharma, "Bluetooth security issues: threats and consequences," in *Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008)*, 2008, pp. 78-80.

[94] E. Willems, "Thirty Years of Malware: A Short Outline," in *Cyberdanger*, ed: Springer, 2019, pp. 1-12.

[95] A. Al-Bataineh and G. White, "Analysis and detection of malicious data exfiltration in web traffic," in *2012 7th International Conference on Malicious and Unwanted Software*, 2012, pp. 26-31.

[96] A. Hutchings and R. Clayton, "Configuring Zeus: A case study of online crime target selection and knowledge transmission," in *2017 APWG Symposium on Electronic Crime Research (eCrime)*, 2017, pp. 33-40.

[97] F. BOSATELLI, "Zarathustra: detecting banking trojans via automatic, platform independent WebInjects extraction," 2013.

[98] justice.gov. (2008, June 3, 2020). *Retail Hacking Ring Charged for Stealing and Distributing Credit and Debit Card Numbers from Major U.S. Retailers*. Available: https://www.justice.gov/archive/opa/pr/2008/August/08-ag-689.html

[99] T. M. Chen and S. Abu-Nimeh, "Lessons from stuxnet," *Computer*, vol. 44, pp. 91-93, 2011.

[100] P. Shakarian, "Stuxnet: Cyberwar revolution in military affairs," MILITARY ACADEMY WEST POINT NY2011.

[101] K. Zetter, "DigiNotar files for bankruptcy in wake of devastating hack," *Wired magazine, September*, 2011.

[102] C. Bronk and E. Tikk-Ringas, "The cyber attack on Saudi Aramco," *Survival*, vol. 55, pp. 81-96, 2013.

[103] R. Sherstobitoff and M. Itai Liba, "Dissecting Operation Troy: Cyberespionage in South Korea," *Korea*, vol. 2009, p. 10, 2013.

[104] H. YAPAR, "RUSSIA'S HYBRID WARFARE," 2020.

[105] M. McQuade, "The untold story of NotPetya, the most devastating cyberattack in history," ed: Wired, 2018.

[106] R. Morris and K. Thompson, "Password security: A case history," *Communications of the ACM*, vol. 22, pp. 594-597, 1979.

[107] D. Ferbrache, *A pathology of computer viruses*: Springer Science & Business Media, 2012.

[108] Z. Trabelsi, S. Zeidan, K. Shuaib, and K. Salah, "Improved session table architecture for denial of stateful firewall attacks," *IEEE Access*, vol. 6, pp. 35528-35543, 2018.

[109] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz, "A data mining analysis of RTID alarms," *Computer Networks*, vol. 34, pp. 571-577, 2000.

[110] B. Kannaiyaraja and A. Senthamaraiselvan, "Routers Sequential Comparing Two Sample Packets for Dropping Worms‖," *IJ Computer Network and Information Security*, vol. 9, pp. 38-46, 2012.

[111] W. Venema, "TCP WRAPPER," in *UNIX Security Symposium III: proceedings: Baltimore, MD, September 14-16, 1992*, 1992, p. 85.

[112] W. Park and S. Ahn, "Performance comparison and detection analysis in snort and suricata environment," *Wireless Personal Communications*, vol. 94, pp. 241-252, 2017.

[113] H. Abedsoltan, "Meso-Scale Wetting of Paper Towels," Miami University, 2017.

[114] H. Abedsoltan, G. Wood, and D. S. Keller, "Characterization of Paperboard Formation using Soft X-radiography and Image Analysis."

[115] A. R. SA, M. Shafiee, H. Abedsoltan, and A. Shafiee, "Gas barrier and mechanical properties of crosslinked ethylene vinyl acetate nanocomposites," *Journal of composite materials,* vol. 47, pp. 2987-2993, 2013.

[116] A. Moosaie, N. Shekouhi, N. Nouri, and M. Manhart, "An algebraic closure model for the DNS of turbulent drag reduction by Brownian microfiber additives in a channel flow," *Journal of Non-Newtonian Fluid Mechanics,* vol. 226, pp. 60-66, 2015.

[117] M. Nemati, J. Ansary, and N. Nemati, "Machine Learning Approaches in COVID-19 Survival Analysis and Discharge Time Likelihood Prediction using Clinical Data," *Patterns,* p. 100074, 2020.

[118] N. Shekouhi, "Towards A Standard Clinically Relevant Testing Protocol For the Assessment of Growing Rods," Bioengineering Department, The University of Toledo, 2020.

[119] M. Nemati, "Machine Learning Approaches in Kidney Transplantation Survival Analysis Using Multiple Feature Representations of Donor and Recipient," Electrical Engineering and Computer Science, University of Toledo, 2020.

[120] M. Fathi, M. Nemati, S. M. Mohammadi, and R. Abbasi-Kesbi, "A MACHINE LEARNING APPROACH BASED ON SVM FOR CLASSIFICATION OF LIVER DISEASES," *Biomedical Engineering: Applications, Basis and Communications,* vol. 32, p. 2050018, 2020.

[121] N. Shekouhi, D. Dick, M. W. Baechle, D. K. Kaeley, V. K. Goel, H. Serhan, *et al.*, "Clinically Relevant Finite Element Technique Based Protocol to Evaluate Growing Rods for Early Onset Scoliosis Correction," *JOR Spine,* 2020.

[122] V. K. Goel, D. K. Kaeley, M. W. Baechle, N. Shekouhi, D. Dick, and H. Serhan, "Finite element based test protocol to evaluate growth rods used in pediatric scoliosis patients," presented at the Biomedical Engineering Society (BMES) Annual Meeting, 2019.

[123] L. LABORATORY. (1998, June 3, 2020). *1998 DARPA Intrusion Detection Evaluation Dataset*. Available: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset

[124] I. University of California. (1999, June 3, 2020). *KDD Cup 1999 Data*. Available: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[125] J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection," *Results from the JAM Project by Salvatore,* pp. 1-15, 2000.

[126] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, 2009, pp. 1-6.

[127] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, 2013, pp. 4487-4492.

[128] E. Aghaei and G. Serpen, "Host-based anomaly detection using Eigentraces feature extraction and one-class classification on system call trace data," *arXiv preprint arXiv:1911.11284*, 2019.

[129] E. Aghaei and G. Serpen, "Ensemble classifier for misuse detection using N-gram feature vectors through operating system call traces," *International Journal of Hybrid Intelligent Systems*, vol. 14, pp. 141-154, 2017.

[130] G. Serpen and E. Aghaei, "Host-based misuse intrusion detection using PCA feature extraction and kNN classification algorithms," *Intelligent Data Analysis*, vol. 22, pp. 1101-1114, 2018.

[131] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, 2015, pp. 1-6.

[132] N. Moustafa, "Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic," University of New South Wales, Canberra, Australia, 2017.

[133] N. Moustafa. (2018, June 3, 2020). *The UNSW-NB15 Dataset Description*. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

[134] T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," in *2017 IEEE 26th international symposium on industrial electronics (ISIE)*, 2017, pp. 1881-1886.

[135] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, vol. 25, pp. 18-31, 2016.

[136] Ixia. (1998). Available: https://www.ixiacom.com/products/perfectstorm

[137] Argus. (June 3, 2020). Available: https://openargus.org/

[138] Zeek. (June 3, 2020). Available: https://zeek.org/

[139] V. Bolón-Canedo, D. Rego-Fernández, D. Peteiro-Barral, A. Alonso-Betanzos, B. Guijarro-Berdiñas, and N. Sánchez-Maroño, "On the scalability of feature selection methods on high-dimensional data," *Knowledge and Information Systems*, vol. 56, pp. 395-442, 2018.

[140] H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," in *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, 1995, pp. 388-391.

[141] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, pp. 273-324, 1997.

[142] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern recognition letters*, vol. 15, pp. 1119-1125, 1994.

[143] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, pp. 267-288, 1996.

[144] T. Y. Young, *Handbook of pattern recognition and image processing (vol. 2): computer vision*: Academic Press, Inc., 1994.

[145] A. Hoerl and R. Kennard, "Encyclopedia of Statistical Sciences, 8, chap. Ridge Regression, 129–136," ed: Wiley, New York, 1988.

[146] L. E. Frank and J. H. Friedman, "A statistical view of some chemometrics regression tools," *Technometrics*, vol. 35, pp. 109-135, 1993.

[147] P. Cao, D. Zhao, and O. Zaiane, "An optimized cost-sensitive SVM for imbalanced data learning," in *Pacific-Asia conference on knowledge discovery and data mining*, 2013, pp. 280-292.

[148] S. Köknar-Tezel and L. J. Latecki, "Improving SVM classification on imbalanced data sets in distance spaces," in *2009 ninth IEEE international conference on data mining*, 2009, pp. 259-267.

[149] F. Nelli, "Machine Learning with scikit-learn," in *Python Data Analytics*, ed: Springer, 2018, pp. 313-347.

[150] D. A. Cieslak and N. V. Chawla, "Learning decision trees for unbalanced data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2008, pp. 241-256.

[151] D. A. Cieslak, T. R. Hoens, N. V. Chawla, and W. P. Kegelmeyer, "Hellinger distance decision trees are robust and skew-insensitive," *Data Mining and Knowledge Discovery*, vol. 24, pp. 136-158, 2012.

[152] T. R. Hoens, Q. Qian, N. V. Chawla, and Z.-H. Zhou, "Building decision trees for the multi-class imbalance problem," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2012, pp. 122-134.

[153] Y. Liu, N. V. Chawla, M. P. Harper, E. Shriberg, and A. Stolcke, "A study in machine learning from imbalanced data for sentence boundary detection in speech," *Computer Speech & Language*, vol. 20, pp. 468-494, 2006.

[154] R. Barandela, R. M. Valdovinos, and J. S. Sánchez, "New applications of ensembles of classifiers," *Pattern Analysis & Applications*, vol. 6, pp. 245-256, 2003.

[155] R. Johnson and T. Zhang, "Learning nonlinear functions using regularized greedy forest," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, pp. 942-954, 2013.

[156] M. Greenwald and S. Khanna, "Space-efficient online computation of quantile summaries," *ACM SIGMOD Record*, vol. 30, pp. 58-66, 2001.

[157] P. Li, Q. Wu, and C. J. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Advances in neural information processing systems*, 2008, pp. 897-904.

[158] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, pp. 367-378, 2002.

[159] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5-32, 2001.

[160] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99-109, 1943.

[161] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Diagnosis of multiple cancer types by shrunken centroids of gene expression," *Proceedings of the National Academy of Sciences*, vol. 99, pp. 6567-6572, 2002.

[162] P. C. Mahalanobis, "On the generalized distance in statistics," 1936.

[163] T. S. community. (2008). *scipy.stats.chisquare*. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chisquare.html

[164] s.-l. developers. (2007). *Feature selection*. Available: https://scikit-learn.org/stable/modules/feature_selection.html

[165] P. S. Foundation. *ReliefF feature selection algorithms*. Available: https://pypi.org/project/ReliefF/

[166] P. S. Foundation. *Python3 binding to mRMR Feature Selection algorithm*. Available: https://pypi.org/project/pymrmr/

[167] s.-l. developers. *sklearn.feature_selection.RFE*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

[168] s.-l. developers. *3.2.4.3.1. sklearn.ensemble.RandomForestClassifier*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[169] s.-l. developers. *sklearn.ensemble.BaggingClassifier*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

[170] G. Lemaitre, F. Nogueira, D. Oliveira, and C. Aridas. *imblearn.ensemble.BalancedBaggingClassifier*. Available: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.ensemble.BalancedBaggingClassifier.html

[171] s.-l. developers. *sklearn.ensemble.AdaBoostClassifier*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[172] s.-l. developers. *3.2.4.3.5. sklearn.ensemble.GradientBoostingClassifier*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[173] s.-l. developers. *3.2.4.3.3. sklearn.ensemble.ExtraTreesClassifier*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

[174] G. Lemaitre, F. Nogueira, D. Oliveira, and C. Aridas. *imblearn.ensemble.EasyEnsemble*. Available: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.ensemble.EasyEnsemble.html

[175] s.-l. developers. *sklearn.naive_bayes.GaussianNB*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[176] s.-l. developers. *sklearn.neural_network.MLPClassifier*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[177] T. S. W. Contributors. Available: https://www.spyder-ide.org/

[178] P. S. Foundation. Available: https://www.python.org/

[179] *pandas*. Available: https://pandas.pydata.org/

[180] *NumPy*. Available: https://numpy.org/

[181] *math — Mathematical functions*. Available: https://docs.python.org/3/library/math.html

[182] J. Hunter, D. Dale, E. Firing, M. Droettboom, and t. M. d. team. *Matplotlib*. Available: https://matplotlib.org/users/installing.html

[183] S. developers. *SciPy*. Available: https://www.scipy.org/

[184] M. Waskom. *seaborn: statistical data visualization*. Available: https://seaborn.pydata.org/

[185] s.-l. developer. *scikit-learn*. Available: https://scikit-learn.org/stable/

[186] G. Lemaitre, F. Nogueira, D. Oliveira, and C. Aridas. *Install and contribution*. Available: https://imbalanced-learn.readthedocs.io/en/stable/install.html

[187] C. Aridas. (2019). *imbalanced-learn*. Available: https://github.com/scikit-learn-contrib/imbalanced-learn/

[188] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, 1995, pp. 278-282.

[189] V. Kanimozhi and P. Jacob, "UNSW-NB15 Dataset Feature Selection and Network Intrusion Detection using Deep Learning."

[190] B. A. Tama and K.-H. Rhee, "An in-depth experimental study of anomaly detection using gradient boosted machine," *Neural Computing and Applications*, vol. 31, pp. 955-965, 2019.

[191] M. G. Raman, N. Somu, S. Jagarapu, T. Manghnani, T. Selvam, K. Krithivasan, *et al.*, "An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm," *Artificial Intelligence Review*, pp. 1-32, 2019.

[192] V. Kumar, A. K. Das, and D. Sinha, "UIDS: a unified intrusion detection system for IoT environment," *Evolutionary Intelligence*, pp. 1-13, 2019.

[193] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset," *Cluster Computing*, pp. 1-22, 2019.

[194] Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks," *Applied Sciences*, vol. 9, p. 238, 2019.

[195] J. Sharma, C. Giri, O.-C. Granmo, and M. Goodwin, "Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation," *EURASIP Journal on Information Security,* vol. 2019, p. 15, 2019.

[196] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Dendron: Genetic trees driven rule induction for network intrusion detection systems," *Future Generation Computer Systems,* vol. 79, pp. 558-574, 2018.

Appendix A

Python Code for Attribute Processing

```
#=============================================================================
Finding Uncorrelated Attributes
#=============================================================================
def correlation_detection(data):
    correlated_features = set()      #creates a set
    correlation_matrix = data.corr()  #creates the correlation matrix
    for i in range(len(correlation_matrix .columns)):
        for j in range(i):
            if abs(correlation_matrix.iloc[i, j]) > 0.8: #finds the uncorrelated attributes
                colname = correlation_matrix.columns[i]
                correlated_features.add(colname)
    return correlated_features
#=============================================================================
Eliminating Uncorrelated Attributes
#=============================================================================
def keep_related_columns(data, features): #gets correlated_features and the dataset as the inputs
    data_x.drop(labels=features, axis=1, inplace=True) #drops uncorrelated attributes from the dataset
```

```
#================================================================================
Genetic Algorithm
#================================================================================
def genetic_algo(data_x, data_y):
    kappa_scorer = make_scorer(f1_score, average=None) #uses f1-score to evaluate the attributes
    selector = GeneticSelectionCV(DecisionTreeClassifier(), #sets parameters for Genetic Algorithm
                                  cv=5,
                                  scoring="accuracy",
                                  max_features=5,
                                  n_population=50,
                                  crossover_proba=0.5,
                                  mutation_proba=0.2,
                                  n_generations=40,
                                  crossover_independent_proba=0.5,
                                  mutation_independent_proba=0.05,
                                  tournament_size=3,
                                  n_gen_no_change=10,
                                  caching=True,
                                  n_jobs=-1)
    selector = selector.fit(data_x, data_y) #fits the Genetic algorithm on the dataset


    return data_x.columns[selector.support_] #returns the dataset with the selected attributes


#================================================================================
Sequential Feature Selection
#================================================================================
def step_forward_wrapper(data_x, data_y):
    kappa_scorer_f1 = make_scorer(f1_score, average=None) #uses f1-score to evaluate the attributes
    feature_selector = SequentialFeatureSelector(DecisionTreeClassifier(),  #sets parameters
        k_features=8,
```

```
        forward=True,
        floating=False,
        verbose=2,
        scoring= kappa_scorer_f1,
        cv=0)
    features = feature_selector.fit(data_x, data_y)  #fits the sequential selector on the dataset
    return data_x.columns[list(features.k_feature_idx_)] #returns the dataset with the selected attributes
```

Appendix B

Appendix CMiscellaneous Content and Tables

Three ensemble estimators with different split criterion are utilized in the proposed algorithm. Each has different parameters needed to set with which the final result has been improved. All the estimators make use of decision tree as their base estimator. 101 number of the base estimators are used in parallel with each other. All the three classifiers sample the training subset uniformly with an equal probability of its instances being randomly selected. Since sampling method is employed along with replacement, this method of sampling is called bootstrapping. Due to the imbalanced issue affected our dataset, we utilized Balanced Bagging resampling all classes except for the majority one. Data resampling is not designed for XGBoost and Random Forest. The parameters along with their assigned values for this project are tabulated as Table B.1,

Table B.1 The values that the parameters of estimators are set by

Parameters	BB	XGBoost	RF
Base Estimator	Decision Tree	Decision Tree	Decision Tree
N. of Estimator	101	101	101
Sampling Method	Uniform	Uniform	Uniform
Resampling Method	Not majority	NA	NA

Parameters	BB	XGBoost	RF
Replacement	True	True	True
N. of Jobs	-1	maximum number of processors	-1
Random State	0	0	0
Split Criterion	gini	percentiles of feature distribution	Hellinger Distance
Booster	Averaging of trees	gbtree	Averaging of trees
Size of Buffer	Size of trees	Number of training samples	Size of trees
Learning Rate	0.01	0.1	0.01

Table B.2 shows the features selected by five studies cited from the current literature. The optimal feature subset in [36] is not listed in the table since it is not reported in the paper. Also, in [35] all the features are considered for both training and testing phase. In this table, each column represents the features selected by the feature selection algorithms proposed in the papers compared to the feature selection method that we proposed.

Table 7.2 Feature subset selected by the different literature

Original Attributes	Proposed Model	[33]	[34]	[25]	[26]	[32]
Rate	x			x		
Srcip						
Sport						
Dstip						
Dstport						
State	x					x
Sload	x		x	x		
dload	x			x		x
Swin	x					
Dwin	x			x		x
Stcpb	x			x		
Dtcpb	x					
Trans_depth						x
Res_bdy_len						x
Djit	x					
Stime						
Ltime						
Sinpkt	x			x		
Tcprtt	x					
Synack						
Ackdat				x		
Is_sm_ips_sport						
Ct_ftw_http_mthd						x
Is_ftp_login						
Ct_ftp_cmd						

Original Attributes	Proposed Model	[33]	[34]	[25]	[26]	[32]
Ct_src_ltm				x		
Proto	x	x			x	x
Dur	x	x		x	x	
Sbytes	x	x	x	x	x	x
Dbytes	x	x		x	x	x
Sttl	x	x	x	x	x	x
Dttl	x	x			x	x
Sloss	x	x			x	
Dloss	x	x			x	x
Service	x	x		x	x	x
Spkts	x	x			x	
Dpkts	x	x			x	x
Smean		x		x	x	x
Dmean		x		x	x	x
Sjit	x	x			x	x
Dinpkt	x	x			x	
Ct_state_ttl		x			x	x
Ct_srv_src		x		x	x	x
Ct_srv_dst		x		x	x	x
Ct_dst_ltm		x		x	x	
Ct_src_dport_ltm		x		x	x	x
Ct_dst_sport_ltm		x		x	x	x
Ct_dst_src_ltm		x	x	x	x	x

Table B.3 summarizes the scaling (SC), feature selecting (FS), Train-Test splitting (TTS), and classification methodologies (CL) employed by all seven papers cited from the current literature. We compare their methods with ours in this table.

Table 7.3 *Different strategies that were utilized in seven literature in comparison with the proposed model*

	Proposed Model	[25]	[26]	[32]	[33]	[34]	[35]	[36]
SC	Min-Max	NR	NR	NR	NR	NR	NR	Min-Max
TTS	Hold-Out	Hold-Out	Hold-Out	Hold-Out	Hold-Out	Hold-Out	CV	CV
FS	SFS Elastic Net	Extra Tree classifier	Information gain (IG)	DT Genetic Algorithm	IG	RFE	Not Used	SVM
CL	Balanced Bagging XGBoost RF-HDDT	ELM	C5 CHAID CART QUEST	Dendron	C5 CART CHAID QUEST	Deep MLP	GBM	HC-IBGSA SVM