

Presenting an Improved Method for Detecting Computer Attacks Using Feature Reduction Techniques and Machine Learning Algorithms

Hadi Najafi Mohsenabad

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
May 2024
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy in Applied Mathematics and Computer Science.

Prof. Dr. Nazım Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Applied Mathematics and Computer Science.

Asst. Prof. Dr. Mehmet Ali Tut
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev

2. Prof. Dr. Hamza Erol

3. Prof. Dr. Efendi Nasiboğlu

4. Asst. Prof. Dr. Müge Saadetoğlu

5. Asst. Prof. Dr. Mehmet Ali Tut

ABSTRACT

Experts in computer network security have come to view intrusion detection systems (IDS) as a crucial research topic because of the growing use of computer networks and the corresponding increase in security breaches.

In computer network security, the escalating use of computer networks and the corresponding increase in cyberattacks have propelled Intrusion Detection Systems (IDSs) to the forefront of research in computer science. IDSs are a crucial security technology that diligently monitors network traffic and host activities to identify unauthorized or malicious behaviour.

In this work, a machine learning algorithm-based approach is presented that can identify, with 100% accuracy, the detection rate of Brute Force type attacks and secure traffic, as well as the detection rate of Bot type attacks with more than 99% accuracy when the fewest identity features are used.

Thanks to the CFSSubsetEval feature selection mechanism, this method can use fewer features with the same level of intrusion detection accuracy. Developing an intrusion detection model takes less time than other techniques.

Also, this study develops highly accurate models for detecting a diverse range of cyberattacks using the fewest possible features, achieved via a meticulous selection of features. We chose 5, 9, and 10 features, respectively, using the Artificial Bee Colony (ABC), Flower Pollination Algorithm (FPA), and Ant Colony Optimization

(ACO) feature-selection techniques. We successfully constructed different models with a remarkable detection accuracy of over 98.8% (approximately 99.0%) with Ant Colony Optimization (ACO), an accuracy of 98.7% with the Flower Pollination Algorithm (FPA) and an accuracy of 98.6% With the Artificial Bee Colony (ABC). Another achievement of this study is the minimum model Building time achieved in intrusion detection, which was equal to 1 s using the Flower Pollination Algorithm (FPA), 2 s using the Artificial Bee Colony (ABC), and 3 s using Ant Colony Optimization (ACO). Our research leverages the comprehensive and up-to-date CSE-CIC-IDS2018 dataset and uses the pre-processing Discretize technique to discretize data. Furthermore, our research provides valuable recommendations to network administrators, aiding them in selecting appropriate machine-learning algorithms tailored to specific requirements.

Keywords: machine learning, network security, feature selection, intrusion detection systems, deep learning, cyberattack, computer network, decision tree.

ÖZ

Bilgisayar ağı güvenliği uzmanları, bilgisayar ağlarının artan kullanımı ve buna bağlı olarak güvenlik ihlallerindeki artış nedeniyle izinsiz giriş tespit sistemlerini (IDS) önemli bir araştırma konusu olarak görmeye başladı.

Bilgisayar ağı güvenliğinde, bilgisayar ağlarının artan kullanımı ve buna bağlı olarak siber saldırılardaki artış, İzinsiz Giriş Tespit Sistemlerini (IDS'ler) bilgisayar bilimi araştırmalarında ön sıralara taşımıştır. IDS'ler, yetkisiz veya kötü niyetli davranışları belirlemek için ağ trafiğini ve ana bilgisayar etkinliklerini özenle izleyen önemli bir güvenlik teknolojisidir.

Bu çalışmada, Brute Force tipi saldırıların ve güvenli trafiğin tespit oranının yanı sıra Bot tipi saldırıların tespit oranını da %99'un üzerinde doğrulukla %100 doğrulukla tespit edebilen, makine öğrenmesi algoritması tabanlı bir yaklaşım sunulmaktadır. en az kimlik özelliği kullanıldığında.

CFSSubsetEval özellik seçme mekanizması sayesinde bu yöntem, aynı düzeyde izinsiz giriş tespit doğruluğuyla daha az özellik kullanabilir. Bir saldırı tespit modelinin geliştirilmesi diğer tekniklere göre daha az zaman alır.

Ayrıca bu çalışma, çok çeşitli siber saldırıları tespit etmek için mümkün olan en az özelliği kullanarak ve özelliklerin titizlikle seçilmesiyle elde edilen yüksek doğruluklu modeller geliştirmektedir. Yapay Arı Kolonisi (ABC), Çiçek Tozlaşma Algoritması

(FPA) ve Karınca Kolonisi Optimizasyonunu kullanarak sırasıyla 5, 9 ve 10 özelliği seçtik.

(ACO) özellik seçme teknikleri. Karınca Kolonisi Optimizasyonu (ACO) ile %98,8'in üzerinde (yaklaşık %99,0) dikkate değer bir tespit doğruluğu, Çiçek Tozlaşma Algoritması (FPA) ile %98,7'lik bir doğruluk ve Yapay Arı ile %98,6'lık bir doğrulukla farklı modelleri başarıyla oluşturduk. Koloni (ABC). Bu çalışmanın bir diğer başarısı, Çiçek Tozlaşma Algoritması (FPA) kullanılarak 1 saniyeye, Yapay Arı Kolonisi (ABC) kullanılarak 2 saniyeye ve Karınca Kolonisi Optimizasyonu kullanılarak 3 saniyeye eşit olan izinsiz giriş tespitinde elde edilen minimum model oluşturma süresidir (AKO). Araştırmamız kapsamlı ve güncel CSE-CIC-IDS2018 veri kümesinden yararlanıyor ve verileri ayırklaştırmak için ön işleme Ayırklaştırma tekniğini kullanıyor. Ayrıca araştırmamız ağ yöneticilerine değerli tavsiyeler sunarak onların belirli gereksinimlere göre uyarlanmış uygun makine öğrenimi algoritmalarını seçmelerine yardımcı olur.

Anahtar Kelimeler: makine öğrenmesi, ağ güvenliği, özellik seçimi, saldırı tespit sistemleri, derin öğrenme, siber saldırı, bilgisayar ağı, karar ağacı.

DEDICATION

I dedicate my thesis to my parents for their endless love, support, and encouragement throughout my pursuit of education. I hope this achievement will fulfill the dream they envisioned for me.

ACKNOWLEDGMENT

I would like to express my thanks and special appreciation to my supervisor, Assistant Professor Dr. Mehmet Ali Tut. You have been an incredible mentor for me, and I am grateful for your guidance throughout my PhD journey. Thank you for encouraging my research and allowing me to grow as a research scientist.

I am also profoundly grateful to my beloved mother, Touran Shabani. Without your unwavering support and encouragement, I could not have accomplished what I have. I express my heartfelt thanks to my dear sister, Elham Najafi Mohsenabad.

This achievement would not have been possible without my family's support and my supervisor's guidance. I am genuinely thankful for the love and encouragement I have received.

TABLE OF CONTENTS

ABSTRACT	iii
ÖZ	v
DEDICATION	vii
ACKNOWLEDGMENT	viii
LIST OF TABLES	xi
LIST OF FIGURES	xii
1 INTRODUCTION	1
2 PRELIMINARIES AND DEFINITION	6
2.1 Intrusion Detection System (IDS)	6
2.2 Crucial-Elements of System for Detecting Intrusions	7
2.3 Weka and Python Language	8
2.3.1 Weka	8
2.3.2 Python	9
3 INTRUSION DETECTION: BRUTE FORCE AND BOT FOCUS	12
3.1 Dataset	12
3.2 Experimental System Specifications	14
3.3 Attack Using Brute Force	14
3.4 The Bot Attack	15
3.5 The Selection of Features	15
3.6 The Methodology	16
3.7 Time Taken To Build a Model	18
3.8 Result	18
3.8.1 Brute Force Detection: Comparative Algorithm Result	18

3.8.2 Bot Detection: Comparative Result	19
4 CYBERSECURITY WITH BIO-INSPIRED ATTACK DETECTION	21
4.1 Theoretical Background	21
4.2 Methodology	25
4.2.1 Data Cleaning.....	25
4.2.2 Data Filtering	28
4.3 The Selection of Features.....	31
4.3.1 ABC, or Artificial Bee Colony.....	31
4.3.2 The Algorithm for Flower Pollination (FPA)	34
4.3.3 Ant Colony Techniques for Optimization (ACO).....	38
4.4 Performance Evaluation	40
4.5 Time Taken To Build a Model	41
4.6 Experimental Design.....	41
4.6.1 Dataset.....	41
4.6.2 Experimental System Specification	43
4.7 Result	43
5 CONCLUSION	47
REFERENCES.....	49

LIST OF TABLES

Table 3.1: Attack types and secure traffic in the dataset CSE-CIC-IDS2018	14
Table 3.2: The CSE-CIC-IDS2018 dataset's record count	15
Table 3.3: The Results of evaluating different algorithms for Brute Force attack	20
Table 3.4: The outcome of comparing various Bot attack algorithms	21
Table 4.1: List of fields removed prior to feature selection	27
Table 4.2: Traffic class quantity and percentage pre/post data cleaning.	29
Table 4.3: Features selected by applying the Bee algorithm	36
Table 4.4: Featured deleted by applying the Flower Pollination algorithms	39
Table 4.5: Features chosen using an Ant Colony Optimization algorithm	42
Table 4.6: Attack types and secure traffic in the dataset CSE-CIC-IDS2018	45
Table 4.7: The CSE-CIC-IDS2018 dataset's record count	45
Table 4.8: Results of different algorithms using features selected by the Ant Colony Optimization algorithm	47
Table 4.9: Results of different algorithms using features selected by the Flower Pollination algorithm (FPA).....	48
Table 4.10: Results of different algorithms using features selected by the Artificial Bee Colony (ABC).....	48

LIST OF FIGURES

Figure 1.1: The intrusion detection system	4
Figure 1.2: Network based intrusion detection system	5
Figure 3.1: Feature selection steps	17
Figure 4.1: An intrusion detection model with binary symmetry	23
Figure 4.2: An intrusion detection flowchart	39

Chapter 1

INTRODUCTION

Large-scale local area networks use anomaly detection as a primary technique for solving security issues. However, it can be challenging to extract accurate traffic data for anomaly identification [1]. Intrusion detection in computer networks is crucial as it affects communication and security; yet, finding network intrusions remains a challenge [1]. Network intrusion detection is complicated due to the need for significant data for training advanced machine learning models [1].

The emergence of new threats that older systems cannot detect presents significant difficulties in network intrusion detection [2].

Therefore, this study aimed to compare multiple techniques to develop an effective network Intrusion Detection System.

The increasing complexity and volume of real-time traffic, driven by the rapid development and adoption of technologies such as 5G, IoT, and Cloud Computing, have led to more sophisticated and varied cyberattacks, posing significant challenges to cyberspace security [3].

Network Intrusion Detection Systems (NIDSs), which serve as the firewall's backup line of defense, function to develop strategies, provide real-time monitoring, swiftly identify malicious network attacks, and implement dynamic security measures [3].

Cybersecurity research has become essential given how much networks are used in the modern world [4]. Current Intrusion Detection Systems (IDSs) have failed to identify new attacks, improve detection accuracy, and lower false alarm rates despite years of research. To solve these issues, many scholars have concentrated on developing IDSs with machine learning methods [4]. Machine learning offers accurate and automatic identification of differences between normal and abnormal data, including recognizing unidentified attacks, while providing excellent generalizability [4].

Feature Selection (FS) techniques play a crucial role in data preprocessing by enabling effective data reduction and supporting the identification of precise data models [5]. Although extensive searching for the ideal feature subset is often impractical, the literature offers numerous search algorithms for overcoming this difficulty [5]. Four widely used techniques employed in feature selection for effective network intrusion detection are the filter, wrapper, embedding, and hybrid methods [5].

The dataset KDD CUP'99 challenge results demonstrated the superiority of Long Short-Term Memory (LSTM) classifiers over other static classifiers, underscoring the significance of LSTM networks for intrusion detection [6]. LSTM networks can learn from past data and establish connections between connection records, making them compelling in intrusion detection [6].

Hyperparameters govern the learning process and model quality in almost all machine learning algorithms. Adjusting these hyperparameters can lead to the generation of infinite models with varying performance levels and learning rates, depending on the specific techniques used for machine learning [7].

Exponential data growth in recent times has posed significant challenges in the classification process [8]. Feature selection solves this problem, improving data classification accuracy while reducing data complexity [8] because many time-consuming features play a role in the detection process.

In order to increase the efficacy of Network Intrusion Detection Systems (NIDSs), feature selection is essential [9]. The feature selection approach influences the time needed to observe traffic behavior and the degree of accuracy improvement. The four approaches to feature selection include wrapper, hybrid techniques, filtering, and embedding [9].

Finding a subset of features with the fewest features is the goal of this study. It takes the least time and has the fewest risks to identify attacks and traffic of the Brite Force and Bot types with the highest accuracy. In order to identify safe traffic and brute force attacks, five characteristics out of 80 were used for the CFSSubsetEval feature reduction technique in this investigation. This technique swiftly and precisely identified 100% of safe traffic and brute force attacks. It has also selected only four features from among all the features to recognize attacks of this kind, and it has effectively detected traffic with an accuracy of more than 99.5%.

Moreover, another goal of this research's primary contribution is its attempt to pinpoint a subset of features most effective at distinguishing between various attacks and regular network traffic while maintaining the fewest overall features possible. We used a variety of search techniques to perform feature selection using Ant Colony

Optimization (ACO), the Artificial Bee Colony (ABC), and Flower Pollination Algorithms (FPAs) in order to accomplish this goal.

As a result, during the feature selection process, subsets of 5, 9, and 10 features were chosen from a pool of 67 features. During the data-cleaning phase, 13 features out of the initial 80 were removed, which is noteworthy. Figure 1 visually depicts the Intrusion Detection System approach covered in this work using a flowchart.

Figure 2 depicts the network-based Intrusion Detection System following the rephrased active voice.

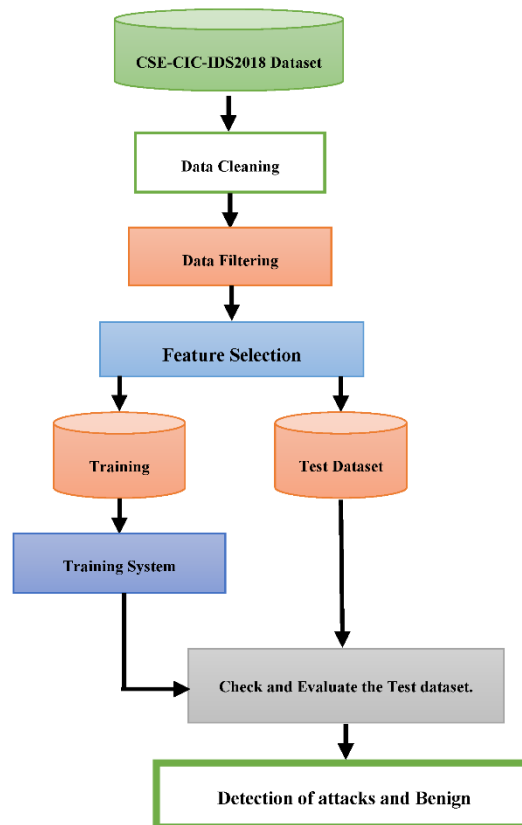


Figure 1.1: The intrusion detection system [40,41].

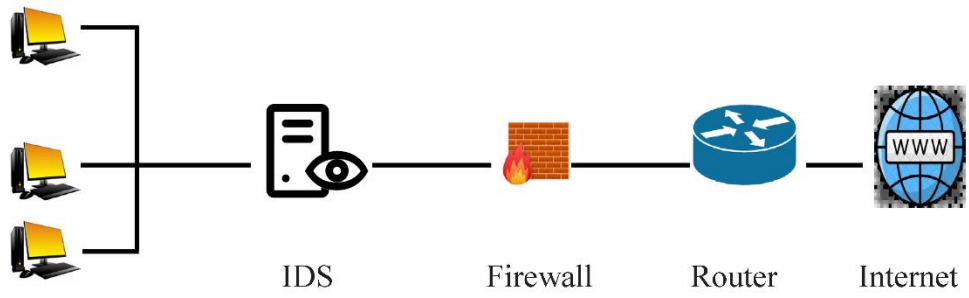


Figure 1.2: Network-based intrusion detection system [40,41].

Chapter 2

PRELIMINARIES AND DEFINITION

Within the confines of this chapter, a comprehensive exploration awaits as we delve into an insightful review and elucidation of indispensable tools. This encompassing discussion will provide concise definitions and meticulously unravel the intricate properties inherent in these tools, shedding light on their nuanced characteristics and functionalities.

2.1 Intrusion Detection System (IDS)

An intrusion detection system (IDS) is a security tool used to keep an eye on system or network activity for unusual or malicious activity and identify and address security issues. Finding unauthorized access, misuse, or other unusual activity that can point to a security breach is the primary goal of an intrusion detection system (IDS).

Intrusion Detection Systems come in two primary varieties:

Intrusion Detection System (NIDS) that is network-based:

- i. Tracks network activity in real-time.
- ii. Examines, via network analysis, packets passing through and compares them to specified signatures or anomalies in behavior.
- iii. It can be positioned carefully to monitor traffic at various locations throughout a network.

Intrusion Detection System (HIDS) depending on the host:

- i. Keeps an eye on what is happening on specific PCs or hosts.

- ii. Checks system calls and log files for odd actions or trends.
- iii. Provide a more thorough perspective of the actions taking place on a particular host.

2.2 Crucial Elements of Systems for Detecting Intrusions

There are six key features of intrusion detection system:

Detection based on signatures:

- i. Utilizes a database containing well-known attack patterns.
- ii. Checks these signatures against system activity or network traffic.
- iii. Efficient in identifying recognized hazards.

Anomaly-based Detection:

- i. Creates a baseline of typical system or network behavior.
- ii. Identifies any departures from this baseline as possibly unauthorized.
- iii. It helps identify novel or unidentified hazards.

Conductual Examination:

- i. Keeps track of system and user behavior throughout time.
- ii. Finds trends that can point to nefarious activity.
- iii. Aids in the detection of complex attacks that can lack a predetermined signature.

Monitoring in real-time:

- i. Functions in real-time, sending out alerts as soon as possible as possible risks are identified.
- ii. Permits prompt reaction to security-related situations.

Record-keeping and reporting:

- i. Creates reports and logs of incidents that are found.
- ii. Aids in forensic analysis and contributes to the enhancement of security regulations.

Reaction Systems:

- i. Specific intrusion detection systems can automatically block or isolate malevolent entities in response to threats they detect.

With their early detection and alerting capabilities, intrusion detection systems enable administrators to swiftly address possible security problems, thereby improving the overall security posture of a network or system.

2.3 Weka and Python Language

In the context of this study, our analytical framework involved utilizing two distinct tools: Weka and Python.

2.3.1 Weka

Weka is a robust and adaptable set of software tools for machine learning that makes it easier to create, use, and assess machine learning algorithms. Waikato Environment for Knowledge Analysis is what we stand for, an acronym created by the University of Waikato in New Zealand. It offers a set of machine learning algorithms and an intuitive graphical user interface for tasks including association rule mining, data preparation, regression, clustering, and classification.

Some of its salient qualities are Weka's broad range of algorithms, adaptability when working with different data types, and graphical user interface that caters to both expert

and inexperienced users. Weka is widely used for research, teaching, and practical applications in academic and industrial contexts. It can handle a wide range of data mining jobs.

Before using machine learning methods, users can clean, filter, and alter datasets using the platform's extensive toolkit for data pretreatment. Weka also facilitates the display of datasets to help with the exploration process, which improves comprehension of data structures and patterns.

One of Weka's noteworthy features is its open-source design, which allows the user community to contribute to its development and customization. Because it is Java-based and has APIs for other program integration, it can be used in various software contexts.

Weka continues to play a crucial role in machine learning as a flexible and approachable platform that enables practitioners, researchers, and students to investigate, test, and implement machine learning solutions in various domains.

2.3.2 Python

Python is a high-level, interpreted, multipurpose programming language known for being simple to understand and comprehend. Python was developed by Guido van Rossum and was made available in 1991. Python has become one of the most popular and commonly used programming languages.

Among Python's salient characteristics are:

i. Readability and Ease of Use:

Because of its simple, legible syntax, Python is easy to learn and encourages program maintainability.

ii. Broad Standard Library:

A vast standard library of modules and packages covering everything from file management to web development is included with Python.

iii. Flexibility:

Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Because of its adaptability, it can be used in various settings.

iv. Understanding and Adaptive Typing:

Since Python is an interpreted language, testing and development may happen quickly. It is dynamically typed, which offers flexibility by determining variable types at runtime.

v. Big and Busy Community:

Python boasts a thriving and dynamic developer community that complements the language's vast array of libraries, frameworks, and tools.

vi. Open-Source:

Python is open-source, which promotes creativity and teamwork. Because the language is open-ended, developers can add to and expand upon it.

vii. Cross-Platform Harmoniousness:

Python is a cross-platform language because it works with multiple operating systems, such as Windows, macOS, and Linux.

viii. Machine Learning and Data Science:

Python is a favored option for data analysis and modeling because libraries like NumPy, pandas, sci-kit-learn, and TensorFlow have increased their appeal in data science and machine learning.

ix. Web Design:

Python is frequently used alongside frameworks like Flask and Django for web development. These frameworks make scalable and maintainable web applications more accessible.

x. Development is driven by the community:

The Python Software Foundation is in charge of the language's development, and community input shapes decision-making to ensure the language changes to meet the demands of its users.

Python's adaptability and large ecosystem can benefit web development, data analysis, artificial intelligence, automation, scientific computing, and many other fields. Because of its readability and simplicity, it is an excellent option for novice and seasoned coders.

Chapter 3

INTRUSION DETECTION: BRUTE FORCE AND BOT FOCUS

This chapter introduces a machine learning-based method capable of achieving 100% accuracy in detecting secure traffic and Brute Force attacks and more than 99% accuracy in detecting Bot attacks using minimal identity features. The proposed approach is trained and tested on the CSE-CIC-IDS2018 dataset, a recent standard for evaluating computer network intrusion detection techniques.

Implementing the CFSSubsetEval feature selection method reduces feature usage while maintaining high intrusion detection accuracy. Additionally, this method facilitates faster model development compared to alternative approaches.

3.1 Dataset

The Communications Security Establishment (CSE) and the Canadian Institute for Cyber Security (CIC) have jointly developed the CSE-CIC-IDS2018 dataset for attack detection and prevention. This open-source dataset provides traffic patterns and various attacks found in network architecture [11].

There are 80 features in the dataset, including tags, most consisting of data in terms of statistics regarding internet traffic. The CICFlowMeter tool obtains these features by generating information based on network traffic flow. The dataset includes continual characteristics such as labels, timestamps, destination ports, and protocols. It includes

six scenarios and fourteen types of assaults, such as botnet, denial of service (DoS), infiltration, Web attack, and DDoS [12]. Amazon Web Services (AWSs) offers the CSE-CIC-IDS2018 dataset for download; included are ten CSV files totaling over 16.2 million samples, representing ten recorded network flow days. The CICFlowMeter program extracted more than 80 features to produce the dataset. The CSE-CIC-IDS2018 dataset’s Table 1 [13, 14] offers comprehensive details on the different kinds of assaults and safe traffic.

Table 3.1: Attack types and secure traffic in the dataset CSE-CIC-IDS-2018.

Category	Attack Types
Benign	
DDoS	HOIC
	LOIC-HTTP
	LOIC-UDP
Botnet	Bot
Brute Force	FTP
	SSH
Infiltration	Infiltration
	Hulk
Dos	SlowHTTPTest
	GoldenEye
	Slowloris
Web Attack	Web
	XSS
	SQL Injection

Table 2 shows the record count and attack percentage of the CSE-CIC-IDS2018 dataset.

Table 3.2: The CSE-CIC-IDS2018 dataset’s record count.

Traffic Class	Record Count	Share (%)
Benign	13,484,708	83.070%
HOIC	686,012	4.226%
LOIC-HTTP	576,191	3.550%
Hulk	461,912	2.846%
Bot	286,191	1.76%

FTP—BruteForce	193,360	1.191%
SSH—BruteForce	187,589	1.156%
Infiltration	161,934	0.998%
SlowHTTPTest	139,890	0.862%
GoldenEye	41,508	0.256%
Slowloris	10,990	0.068%
LOIC-UDP	1730	0.011%
Brute Force—Web	611	0.004%
Brute Force—XXS	230	0.001%
SQL Injection	87	0.0005%

There are 16,232,943 records in the CSE-CIC-IDS2018 dataset, with 83.07 percent (or records) consisting of benign traffic, while the remaining records represent malicious activity. This study used 80 percent of the records for training, while the remaining 20 percent was allocated for testing [12-14].

3.2 Experimental System Specifications

The experimental design for this study involved implementing the test procedure on a system comprising an AMD Ryzen 7 1800X eight-core processor operating at a clock speed of 3.60 GHz, paired with 32 GB of RAM and 128 GB allocated as swap space on the hard disk. The system employed an X64-based processor and ran on Windows 10 Enterprise as the operating system.

3.3 Attack using Brute Force

A Brute-Force attack involves the attacker submitting numerous passwords or passphrases and methodically checking all possibilities until the correct one is found. It is an attempt to crack a password or username using an "exhaustive search" or trial-and-error method in cryptography.

Typically, the goal of brute force assaults is to get personal data, including usernames, passwords, passphrases, and Personal Identification Numbers (PINS).

3.4 The Bot Attack

A bot attack uses an automated online request to fool, mislead, or disrupt an application, website, API, or end users in some other way. Bot attacks started as simple spamming campaigns but have grown into complex international criminal networks with their economies and infrastructures.

3.5 The Selection of Features

Feature selection is crucial in machine learning. The optimal selection of significant feature sets achieves better classification performance.

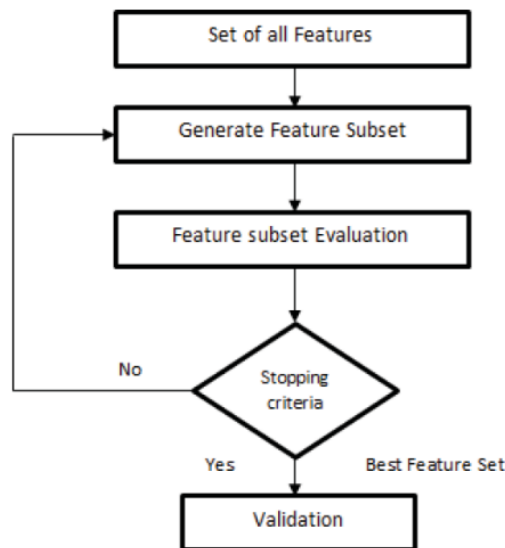


Figure 3.1: Feature selection steps [10].

This research uses the CFSSubsetEval feature selection method. It evaluates the worth of a subset of attributes by considering each feature's individual predictive ability and the degree of redundancy between them. Subsets of features highly correlated with the class while having low inter correlation are preferred.

3.6 The Methodology

We first need a way to sort through the 80 features and select the most relevant to our research's secure network traffic and attack type. Weka software, one of the best packages for machine learning techniques, was also used in this investigation. CFSSubsetEval was the feature selection method used. Five features that can identify brute force attacks were extracted using the CFSSubsetEval feature selection approach. They are listed in the following order:

- 1) Dst Port : Port number of destination,
- 2) Flow IAT Mean : average duration of two flows,
- 3) Init Fwd Win Byts : amount of bytes transmitted in the forward direction in the first window,
- 4) Init Bwd Win Byts : amount of bytes transferred backwards in the first window,
- 5) Fwd Seg Size Min: minimum segment size seen in the forward.

The following table displays the findings of the evaluation of the algorithms, which was conducted using the evaluation criteria of the eleven different algorithms: Twenty percent of this dataset was utilized for machine testing. In contrast, the remaining eighty percent was used to train the machine using these five features.

Our study employed a confusion matrix to evaluate a binary classification problem. Such problems often involve classes of interest, including a minority class and its opposite class, commonly called positives and negatives. In this context, several key performance metrics are used [11, 14, and 15]:

- **True Positive (TP):** These correctly anticipated positive results suggest that the actual and forecasted class values are accurate;
- **True Negative (TN):** the fact that neither the actual class value nor the expected class value is positive is one of the precisely predicted negative values;
- **False Positive (FP):** the situation where the expected class is valid but the actual class is false;
- **False Negative (FN):** a situation where the actual class is positive but the projected class is negative;
- **Accuracy:** The percentage of related samples among the retrieved samples is known as accuracy, another name for a positive value:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- **Precision:** precision is the ratio of accurately expected observations to all anticipated positive observations concerning positive observations:

$$\text{Precision} = \frac{TP}{TP + FN} \quad (2)$$

- **Recall (Sensitivity):** By dividing the total number of true positives by the sum of the true and false negatives, the sensitivity is quantitatively calculated:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

- **F-measure (F1 score):** The F-measure functions as a sort of average between the parameters P (Precision) and R (Recall). P represents the system's accuracy when compared to the predicted data. We express the number of predicted data to all expected data for prediction as a ratio or r:

$$\text{F1Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

3.7 Time Taken to Build a Model

The algorithm's duration is required to build the model using the training data.

3.8 Result

3.8.1 Brute Force Detection: Comparative Algorithm Results

The results in the table below were achieved using various machine-learning methods.

Table 3.3: The result of evaluating different algorithms for Brute Force attack [40, 41].

Algorithm	Time taken to build model	Accuracy	Precision	Recall	F-measure
J48	5.39s	1	1	1	1
Random Forest	133.95s	1	1	1	1
BayesNet	5.32s	1	1	1	1
NaiveBayes	1.05s	99.98	1	1	1
SMO	21.8s	99.83	1	0.997	0.999
Logistic	21.33s	99.96	1	0.999	1
Classification via Regression	813.71s	99.99	1	1	1
LogitBoost	14.31s	99.99	1	0.999	1
JRip	8.64s	1	1	1	1
OneR	1.2s	99.96	1	0.999	1
DTNB	32.6s	1	1	1	1

The best strategy is the BayesNet method, as you can see in the above table, which produced the model rapidly and identified Brute Force attacks with 100% accuracy. If speed is more important to us, the OneR technique provides the fastest detection time for Brute Force attacks. It takes 1.2 seconds to develop a model to identify Brute Force attacks. Using the feature selection method (CFSSubsetEval) mentioned in the previous step, we were able to extract the four most valuable traits to identify Bot type assaults:

- 1) Dst port: port number of destination,

- 2) Fwd Pkt len Min: quantity of packets forwarded,
- 3) Fwd PSH Flags: number of forward-moving packets in which the PSH flag was set,
- 4) Active Max: Max was not active when the flow was idle.

We used 80% of the data to train the system and the remaining 20% to assess and test it using these four attributes. The table below summarizes the outcomes of 12 machine learning algorithms that we obtained.

3.8.2 Bot Detection: Comparative Algorithm Results

Review the results achieved by deploying various machine learning algorithms in Table 4 below:

Table 3.4: The outcome of comparing various Bot attack algorithms [40, 41].

Algorithm	Time taken to build model	Accuracy	Precision	Recall	F-measure
Decision Table	8.77 s	99.59	0.994	1	0.997
Bagging	40.31s	99.59	0.994	1	0.997
AdaBoostM1	14.71s	99.57	0.994	1	0.997
REPTree	40.31s	99.59	0.994	1	0.997
NaiveBayes	0.78s	62.49	0.998	0.489	0.654
BayesNet	2.64s	99.57	0.994	1	0.997
JRip	10.25s	19.57	0.994	1	0.997
Classification via Regression	1407.7s	99.57	0.994	1	0.997
DTNB	24.2s	99.59	0.994	1	0.997
Random Forest	184.1s	99.58	0.994	1	0.997
J48	5.23s	99.59	0.994	1	0.997
Random Tree	2.34s	99.58	0.994	1	0.997

When comparing the machine learning algorithms shown in the above table, the J48 algorithm achieves the best accuracy (99.59% accuracy) in 5.23 seconds for detecting

Bot type attacks; however, if speed is of the essence, the NaiveByes algorithm takes the shortest amount of time (0.78 seconds) to build the model. The BayesNet algorithm can develop a model in 2.64 seconds with 99.57% accuracy, the lowest time with decent accuracy. However, the accuracy obtained is relatively low, at 62.49%.

Chapter 4

CYBERSECURITY WITH BIO-INSPIRED ATTACK DETECTION

4.1 Theoretical Background

Everyday events rely on monitored input data. The IDS generates accurate alarms as informative output, reflecting the events under surveillance. Its primary function involves analyzing the input data [13, 16].

Alerts are generated and streamed to show whether intrusions are present. The categorization of every input data stream unit as either standard or invasive represents a random variable X . A low value ($X = 0$) indicates regular and non-intrusive traffic. In contrast, a high value ($X = 1$) indicates an incursion. Similarly, a random variable Y represents the output information from the IDS, with $Y = 1$ denoting an intrusion warning and $Y = 0$ denoting no signal [13, 16].

Intrusion detection can be patterned using a binary symmetric channel, which incorporates information theory and provides a framework for understanding intrusion detection. The base rate, or prior probability (B), denotes the chance of intrusions in the input data that the Intrusion Detection System (IDS) has detected. Denoted as γ , the false negative rate (FN) indicates the likelihood of an intrusion event being categorized as usual ($p(Y = 0 | X = 1)$); on the other hand, the false positive rate (FP), represented by α , is the likelihood that a typical event will mistakenly be classified as

an incursion ($p(Y = 1 | X = 0)$). In this instance, X stands for the IDS input's random variable. For IDS output, Y stands for the random variable. It is, therefore, feasible to characterize intrusion detection capabilities accurately [13, 16].

Figure 4 illustrates the application of Binary Symmetry in an Intrusion Detection Model.

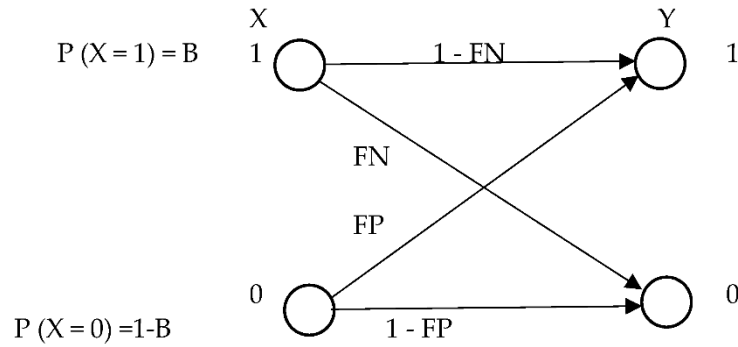


Figure 4.1: An intrusion detection model with binary symmetry.

We suggest a new metric called Intrusion Detection Capability, or CID. It is the simple ratio of the input's entropy to the mutual information between the IDS's input and output.

Entropy and mutual information are related— $H(X) = I(X; Y) + H(X|Y)$, for instance. The entropy $H(Y)$ is significantly greater than $H(X)$ on the right. This represents a plausible IDS scenario in which $H(X)$ is almost zero due to a minimal base rate that is almost zero. However, the IDS may generate a lot of false positives. Therefore, $H(Y)$ may be greater than $H(X)$.

Let X and Y be the random variables representing the IDS input and output, respectively.

The definition of Intrusion Detection Capability is Equation (5):

$$C_{ID} = \frac{I(X; Y)}{H(X)} \quad (5)$$

Equation (6) allows us to articulate the idea of C_{ID} (Conditional Intrusion Detection) based on our knowledge of mutual information and information theory:

$$C_{ID} = \frac{H(X) - H(X|Y)}{H(X)} \quad (6)$$

When considering this, we quantify the decrease in ambiguity about the Intrusion Detection System input using mutual information.

Equation (6) illustrates that the C_{ID} represents the proportion of uncertainty reduction in the Intrusion Detection System input due to the Intrusion Detection System output. Practically speaking, the C_{ID} value is between 0 and 1. An increased C_{ID} value indicates an improved ability of the Intrusion Detection System to classify events [13, 16].

The mutual information $H(X)$ formula (Equation (7)) defines the mutual information of an event [17].

$H(X|Y)$ is expressed in Equation (8):

$$H(X) = -\sum_x p(x) \log p(x) = -B \log B - (1 - B) \log(1 - B) \quad (7)$$

$$H(X|Y) = \frac{-\sum_x \sum_y p(x) p(y|x) \log[p(x)p(y|x)]}{p(y)} = -B(1 - \gamma) \log PPV - B\gamma \log(1 - NPV) - (1 - B)(1 - \alpha) \log NPV - (1 - B)\alpha \log(1 - PPV) \quad (8)$$

Equation (9) is obtained by substituting the equations and is the expression for C_{ID} :

$$C_{ID} = \frac{-B \log B - (1-B) \log(1-B)}{-B(1-\gamma) \log PPV - B\gamma \log(1-NPV) - (1-B)(1-\alpha) \log NPV - (1-B)\alpha \log(1-PPV)} \quad (9)$$

The Intrusion Detection Capability (CID) formula, which gauges how well a system detects intrusions, is shown in Equation (9). This equation defines several parameters: the symbols for base rate (B), false negative (FN) rate (γ), false positive (FP) rate (α), positive predictive value (PPV), and negative predictive value (NPV), which are as follows [1]:

The base rate (B) characterizes the Intrusion Detection System's (IDS) operational environment. A value of $B = 0$ or $B = 1$ indicates that the input solely consists of regular or intrusive events, respectively. On the other hand, figuring out or managing the base rate in practice could be difficult for practitioners. This challenge stems from the fact that they frequently utilize base rate as an operational metric to assess the IDS environment. References [16, 18] provide a detailed discussion on estimating base rate B and prior probabilities.

The probability that an alert would sound without an intrusion is an Intrusion Detection System's false positive (FP) rate (IDS).

The false negative (FN) rate represents the likelihood that the Intrusion Detection System fails to raise the alarm when an intrusion occurs [16, 18].

When the IDS raises an alarm, the positive predictive value (PPV) measures the likelihood that an intrusion is present. It shows the percentage of IDS alarms that are related to real incursions. The PPV can be expressed mathematically as Equation (10):

$$PPV = \frac{B(1-\gamma)}{B(1-\gamma) + (1-B)\alpha} \quad (10)$$

A negative predictive value (NPV) shows the probability of no incursion when the Intrusion Detection System (IDS) does not sound an alarm. Put otherwise, it quantifies the probability that the absence of Intrusion Detection System (IDS) alerts accurately implies the absence of invasions. Equation (11) provides the mathematical expression for NPV [16, 18]:

$$NPV = \frac{(1 - B)(1 - \alpha)}{(1 - B)(1 - \alpha) + B\gamma} \quad (11)$$

4.2 Methodology

4.2.1 Data Cleaning

We implemented several steps to preprocess the CSE-CIC-IDS2018 dataset during the data-cleansing stage. Firstly, we removed specific fields that were deemed redundant or irrelevant. We eliminated the protocol column since each destination port's Dst_Port (Destination Port) field previously provided equivalent protocol values.

Additionally, we removed the Timestamp parameter to ensure unbiased learning and avoid time-based distinctions, allowing learners to differentiate between high-volume and covert attacks accurately. Removing the timestamp field also facilitated convenient dataset combinations or divisions for our experimental frameworks.

Furthermore, we identified and removed 59 record header row duplicates by filtering the data using an empty list of acceptable label values. Among the downloaded files, 'Tuesday-20-02-2018_TrafficForML_CICFlowMeter.csv'

(<https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv> accessed on 20 October 2023) stood out as distinct from the other nine files. From this file, we excluded four additional fields: FlowID, Source Internet Protocol (SrcIP), Source Port (SrcPort), and Destination Internet Protocol (DstIP).

Moreover, we eliminated all occurrences of negative values in the Fwd_Header_Length, Flow_Duration, and Flow-IAT-Min fields, as these values were consistent with their respective fields. Negative values in the fwd_Header_Length field that correspond with extreme values in other fields can distort outlier-prone statistics. Additionally, we removed records where the values in eight specific fields were zero. Table 5 provides the complete list of removed fields before feature selection.

Table 4.1: List of fields removed prior to feature selection.

No	Feature
1	Bwd _ Avg _ Bulk _ Rate
2	Fwd _ Avg _ Bytes _ Bulk
3	Bwd _ Avg _ Packets _ Bulk
4	Fwd _ Avg _ Bulk _ Rate
5	Bwd _ PSH _ Flags
6	Fwd _ Avg _ Packets _ Bulk
7	Bwd _ Avg _ Bytes _ Bulk
8	Bwd _ URG _ Flags

We did not include the Init_Win_bytes_backward and Init_Win_bytes_forward fields due to approximately 50 percent having negative numbers corresponding to these variables.

Similarly, we chose not to utilize the Flow_Duration fields because some values were unreasonably low or zero. Less than 0.6 percent of entries in the Flow Packets/s and Flow Bytes/s columns included “NaN” or “Infinity” values. We eliminated cases when the flow bytes or packets per second had such values.

After completing the data cleaning stage and before feature selection, we applied an unsupervised approach and the discretize filter method to filter our data; this involved using an instance filter called “Discrete” to convert various numerical properties of the dataset into nominal attributes.

The CSE-CIC-IDS2018 dataset consists of 10 files collected over consecutive days. The resulting dataset contained 16,232,943 records and 80 features upon merging these files. Following the data cleaning process, 16,137,168 records and 67 features remained in the dataset. For the data filtering step, we utilized this enhanced dataset. Of the entire amount of records collected (16,137,168), we reserved 3,227,433 records (20 percent) for system testing and assessment, and we assigned 12,909,735 records (80%) for system training.

Table 6 provides a visual representation of the quantity and percentage of each traffic type before and after the completion of the data-cleaning process.

Table 4.2: Traffic class quantity and percentage pre/post data cleaning [40, 41].

Traffic Class	Before Data Cleaning	Percentage before Data Cleaning	After Data Cleaning	Percentage after Data Cleaning
Benign	13,484,708	83.070%	13,390,234	82.978%
DDoS	1,263,933	7.787%	1,263,933	7.832%
Bot	286,191	1.76%	286,191	1.773%
Brute Force	380,949	2.347%	380,943	2.361%
Infiltration	161,934	0.998%	160,639	0.995%
Dos	654,300	4.032%	654,300	4.055%
Web Attack	928	0.0055%	928	0.006%
Total	16,232,943		16,137,168	

4.2.1 Data Filtering

In data mining, ensuring data quality for subsequent mining stages is of utmost importance. Data preprocessing plays a vital role in accomplishing this objective. Its primary aim is to improve the quality of mining outcomes by cleaning the collected data, encompassing activities such as data transformation and dimensionality reduction. In discretization, one transforms continuous or numerical variables into discrete or nominal attributes by dividing them into a finite number of intervals [19, 20].

Among the top ten data mining algorithms are machine learning approaches like naive Bayes, Apriori, decision trees, and leveraged data discretization [21–23]. By employing data discretization, these techniques construct more effective and efficient models. Moreover, transforming continuous features into discrete counterparts provides a representation that aligns with knowledge-level understanding, making them easier to comprehend, utilize, and explain than continuous attributes [21].

In a comparison study, Garcia et al. [23] examined the classification performance of 30 discretized algorithms utilizing a slow, rule-based decision tree and Bayesian learning techniques. Their work provides information on discretizations appropriate for a particular classifier type, those that work with all classifier types, and those that strike a decent balance between the number of intervals generated and the precision of classification. Data discretization divides the range into short intervals showing strong class coherence to determine a set of cut points for continuous features.

Additionally, discretization techniques should reduce the number of intervals while maintaining a high degree of mutual dependency between classes and attributes.

Because of this, we partitioned continuous features into k intervals, or portions, with a maximum of $k - 1$ cut points. The trade-off between the arity (number of intervals) and its effect on accuracy must be considered [21].

The subsequent section summarizes the data discretization process applied to a dataset containing N examples and C target classes. We apply an algorithm for discretization to transform the continuous attribute A into m discrete intervals represented by $D = \{[d_0, d_1], [d_1, d_2], \dots, [d_{m-1}, d_m]\}$, where d_0 is the minimum value, d_m is the maximum value, and $d_i < d_{i+1}$ for $i = 0, 1, \dots, m - 1$. The resulting discrete representation D is a discretization scheme for attribute A , and $P = \{d_1, d_2, \dots, d_{m-1}\}$ represents attribute A 's set of cut points [21]. Sorting the continuous feature values, identifying and assessing cut issues for splitting or merging intervals, dividing or merging constant intervals depending on particular criteria, and stopping at a specific point are the four general processes of the discretization process [38].

Discretization algorithms can be categorized based on different characteristics [21, 22, and 24], as described below:

Dynamic vs. Static: Static discretization is performed before and does not depend on the learning algorithm and the learning task; while creating a model, dynamic discretization occurs. The most commonly used discretizations are static, whereas the discretized ID3 decision tree is an example of a dynamic discretization.

Comparing univariate and multivariate discretization, the former examines a single continuous feature, while the latter considers several features simultaneously.

Supervised versus unsupervised discretization methods differ in their approach to determining appropriate intervals during the discretization process. Supervised methods utilize class information to guide the selection of intervals, considering the class membership of the data. On the other hand, unsupervised methods do not consider class membership when discretizing the data. It is important to remember that the majority of discretization methods require supervision.

Splitting vs. merging: Splitting methods choose a cut-point from all accessible boundary points to divide the domain into two intervals. In contrast, merging methods start with a pre-defined partition and eliminate a potential cut-point to join close intervals.

Local vs. global: Local discretization bases its partition decisions solely on local data, whereas global discretization handles the discretization of every numeric property during preprocessing. Global discretization techniques typically perform better than local ones.

Incremental vs. Direct: Since direct approaches divide the range into k intervals concurrently, they require an extra criterion to find k . Conversely, incremental approaches begin with a straightforward discretization and proceed through an improvement process; however, they need an extra criterion to decide when to end the process. We used an unsupervised strategy to filter the data after the data-cleaning stage before selecting features. In particular, we applied a discretized filter approach called “Discrete,”—an instance filter. The many numerical properties of the dataset were converted into notional qualities using this filter.

4.3 The Selection of Features

An essential component of machine learning is feature selection, which directly impacts classification performance. A well-chosen feature set leads to higher accuracy in classification tasks. This chapter employs three feature selection approaches: the Ant, Bee, Bat, and Wolf algorithms, each selecting various features.

After the data cleaning stage, our dataset consisted of 67 features. We employed three algorithms (Ant et al.). Tables 4.3–4.5 show the outcomes of using three distinct algorithms.

4.3.1. ABC, or Artificial Bee Colony (ABC)

The Artificial Bee Colony (ABC) algorithm takes cues from honey bee activities. Finding the best answers to various issues is made possible by a swarm intelligence system known as colonies [25]. This represents a potential food supply by each point in the search space of the ABC algorithm, which depicts the surroundings of a colony of honey bees. Each food source has a different amount of nectar, and those indicate the fitness value of the food supply. Researchers use three different types of bees in the ABC algorithm: scout, employed, and observation bees [25, 26].

The algorithm only plans to provide a working bee with food. Worker bees figure out how much nectar there is. There are the same number of workers in the colony and observers of artificial bees. The task of collecting high-quality data and forwarding it to observer bees falls to employed bees.

Consequently, nectar-rich food sources will be directed toward observers among the bee population. We continued these steps until we obtained the limit value.

Researchers have developed numerous ABC algorithms in the literature [27] for various purposes. Researchers have created numerous enhanced ABC variations to address global optimization issues [28].

To create test cases, for example, two novel combinatorial ABC algorithms with many requirements have been devised [29]. These algorithms improve both intensification and variety in the bee population by optimizing coverage criteria.

There are five primary phases to the ABC algorithm:

First Step: Based on the amount of food sources indicated by SN, the initialization randomly creates the initial food sources. Two times SN (SN employed bees + SNonlooker bees) is the colony size NP. We generate food sources between the variables' top and lower boundaries. Equation (12) generates the food source positions during the startup stage:

$$X_{ij} = X_{minj} + \text{rand}(0,1)(X_{maxj} - X_{minj}) \quad (12)$$

In dimension j , where $j = 1, 2, 3, \dots, D$, X_{minj} , and X_{maxj} represent the lower and upper bounds. SN defines the number of worker or observer bees. For every solution, we construct a D -dimensional vector x_i ($i = 1, 2, \dots, SN$), where D is the quantity of optimization factors.

Following the initialization stage, we repeat cycles with the populations of the scout, employed, and onlooker bees $C = 1, 2, \dots, MCN$ used in the solution search strategies. MCN is the maximum count of iterations.

Second Step: Evaluate the food sources' fitness value (i.e., nectar amount). We use

$$fit_i = \begin{cases} \frac{1}{1+f_i}, & \text{if } (f_i \geq 0) \\ 1 + abs(f_i), & \text{otherwise} \end{cases} \quad (13)$$

Equation (13) to compute the attributes of every food source individually:

Where fit_i is the value of the specific objective function and f_i is the fitness of the i th food source.

Third Step: After initialization and fitness calculations, the employed bees, following the bee method, explore new food sources with better nectar. Every worker bee updates the locations of their relevant food sources. This way, we create a new food source position v_{ij} :

$$v_{ij} = X_{ij} + \phi_{ij}(X_{ij} - X_{kj}) \quad (14)$$

The letters $j \in \{1, 2, \dots, d\}$ and $k \in \{1, 2, \dots, SN\}$ represent the randomly selected index values. The uniform random number j_{ij} , in this instance, is located in the interval $[-1, 1]$. In dimension j , the reference and randomly chosen food sources are at X_{ij} and X_{kj} , respectively.

Equation (13), once we have established v_{ij} , is used to calculate its fitness value. An employed bee stores the position of the new food source when its fitness exceeds that of the previous one.

Fourth Step: The onlooker bee method: Each chooses a food source based on the fitness values determined in Step 3. High-fitness food sources are more likely to be chosen than others in this selection mechanism [30]. We compute the probability as follows:

$$p_i = \frac{fit_i}{\sum_{k=1}^{SN} fit_k} \quad (15)$$

Upon determining the probability value p_i , each food source produces a random integer known as its fitness value or fit_i . If there is a higher chance of locating the food source than this random number, the spectator bee is given a current food source for the searching–exploiting process. Bystander bees then identify the precise food sources. Similar to the previous bee stage, we evaluated the new technique.

Fifth Step: Process of the scout bee: The Artificial Bee Colony algorithm states that the number of food sources (i.e., solutions) and trial counts determine the incidence of scout bees. A deployed bee’s trial counter value will depart from the food source if it exceeds the upper limit. After that, the working bee changes into a scout bee. The scout bee looks for new food sources.

Using a simulated bee colony, we selected the characteristics in Table 7.

Table 4.3: Features selected by applying the Bee algorithm [40, 41].

No	Feature
1	Dst Port
2	Fwd IAT Min
3	Bwd Pkts/s
4	Fwd Seg Size Min
5	Fwd Act Data Pkts

4.3.2 The Algorithm for Flower Pollination (FPA)

The FPA is a population-based global optimization algorithm.

Yang et al. summarized the features of the Flower Pollination Algorithm procedure into the following four perfect guidelines [31].

Rule 1: Cross-pollination by pollinators, like birds, represents a global pollination stage. Pollinators behave like light birds;

Rule 2: the local pollination process starts with self-pollination on nearby flowers;

Rule 3: the reproduction rate, or floral constancy, is a straight ratio comparable to the two flowers under consideration;

Rule 4: a switch probability implements local and global pollination.

According to the previously mentioned principles, the FPA has a local and a global pollination operator. Every pollen item in the FPA is considered a solution SX_i ; the initialization of solutions within the feasible search space involves the utilization of random vectors. Presented below is the original formula [32]:

$$SX_i = \text{Lower} + R_v(\text{Upper} - \text{Lower}) \quad (16)$$

Where R_v is a D -dimensional random vector in $[0, 1]$ D ; the bottom limit of the search space is bottom = $[l1, ..., ld, ..., lD]$, and the upper limit is Upper = $[u1, ..., ud, ..., uD]$.

Where $i \in \{1, ..., NP\}$, NP is the population size. Pollinators, like birds, have a comparatively wide range of motion and can transport pollen across great distances as part of the global pollination process. Thus, the following is how Rules 1 and 3 are formulated [32]:

$$SX_i^{t+1} = SX_i^t + \gamma L(\lambda)(gbest - SX_i^t) \quad (17)$$

By using the levies distribution denoted by (λ) in (17), we may numerically simulate the flight behavior of birds. Furthermore, (λ) can be regarded as a variable step factor for pollination intensity measurement. As a step factor, γ is the solution i at iteration

t , and $abest$ is the current global best solution. When $L > 0$, the levy distribution can

$$L \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, s \gg s_0 > 0, s_0 = 0.1, \quad (18)$$

be explained as follows [32]:

Using the two Gaussian distributions, U and V , and the standard gamma function, $\Gamma(\lambda)$, whose $\lambda = 1.5$ [32], we compute s :

$$s = \frac{U}{|V|^{1/\lambda}}, U \sim N(0, \delta^2), V \sim N(0, 1) \quad (19)$$

$$\delta^2 = \left\{ \frac{\Gamma(1 + \lambda)}{\lambda \Gamma[(1 + \lambda)/2]} \cdot \frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}} \right\}^{1/\lambda}$$

The conventional normal distribution is shown by $(0, 1)$, and the normal distribution with mean value 0 and variance δ^2 is denoted by $(0, \delta^2)$. Local pollination disperses pollen to a nearby neighbor if used in pollination activities. We can create the model as follows [32], using Rules 2 and 3:

$$SX_i^{t+1} = SX_i^t + \xi(SX_j^t - SX_k^t) \quad (20)$$

Here, j and $k \in \{1, \dots, NP\}$ and ξ is a random vector in $[0, 1]^D$ of dimension D , and SX_j and SX_k are the pollen that has been chosen at random from different flowers on the same plant. Moreover, based on Rule 4, the randomness of the two pollination acts is determined by a probability p . In contrast, the opposite is true if a random value rand in $[0, 1]$ is minor; otherwise, p . Figure 4 displays the FPA flowchart [32].

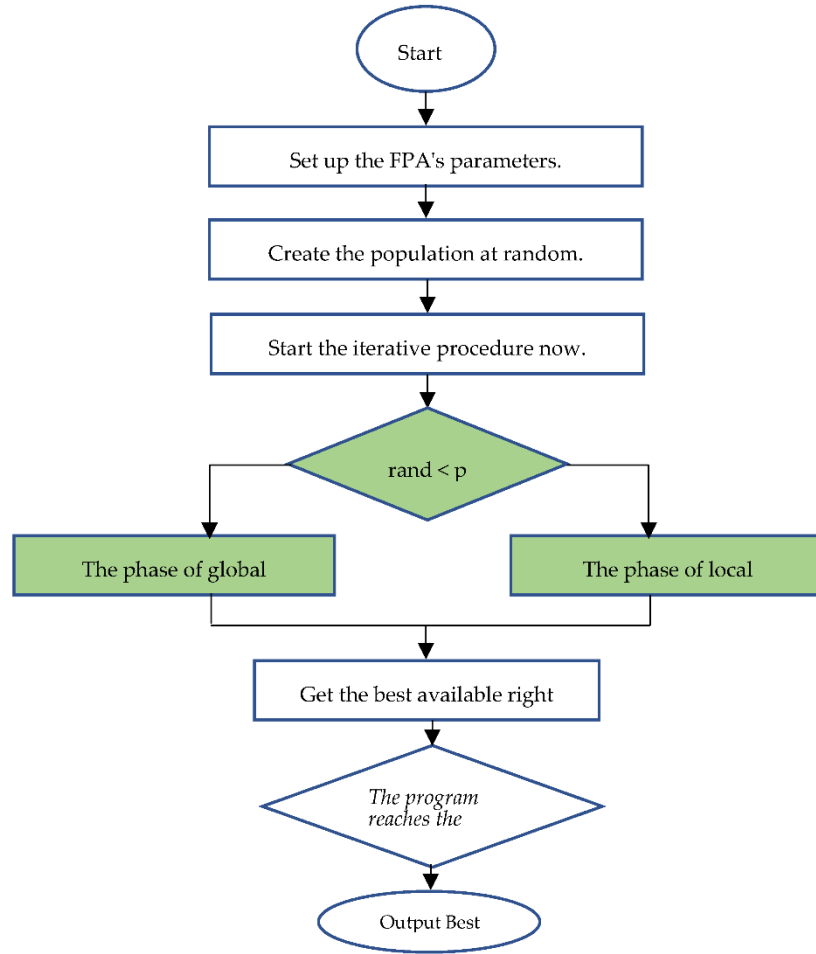


Figure 4.2: An intrusion detection flowchart [32].

Table 8 shows the features selected using a Flower Pollination Algorithm.

Table 4.4: Features deleted by applying the Flower Pollination Algorithm [40, 41].

No	Feature
1	Dst Port
2	Fwd Pkt Len Min
3	Flow Pkts/s
4	Bwd Pkts/s
5	Fwd IAT Min
6	ECE Flag Cnt
7	ACK Flag Cnt
8	Fwd Seg Size Min
9	Fwd Act Data Pkts

4.3.3 Ant Colony Techniques for Optimization (ACO)

Dorigo and colleagues introduced the Ant System (AS) algorithm in the early 1990s as a cutting-edge meta-heuristic technique for resolving combinatorial optimization issues that drew inspiration from nature. The first problem to be solved by the algorithm was the traveling salesperson problem. Recently, this has been improved and modified to enhance its usefulness and apply it to other optimization problems. Ant Colony System (ACS), AS-rank, MAX-MIN, and AS are the upgraded versions of AS [28].

Ant Colony Optimization mimics the cooperative and adaptive mechanisms of natural ant behavior on an individual level. Real ants' foraging habits inspire ACO [28]. We based our computational paradigm for the ACO algorithm on actual ant colonies and their functioning. Our goal was to involve as many constructive computational ants as possible. Each ant follows the generated solution, which relies on its dynamic memory structure recording outcomes from past trials.

Ethologists' findings derived the paradigm regarding the pheromone trail system ants employ to communicate information regarding the most direct paths to food. Upon finding a previously constructed road, an ant can identify it, decide to follow it with a high chance of success, and reinforce the route with its pheromones. A solitary ant, on the other hand, travels around practically at random. What appears is a type of autocatalytic process whereby a track becomes more appealing to ants to follow the more they follow it. As a result, a positive feedback loop defines the process where the likelihood of selecting a path increases in proportion to the number of ants that have

already selected that path. The ACO family of algorithms draws inspiration from the abovementioned mechanism [33].

It is also possible to consider a FS period before the training phase. Which traits are more discriminative than others is determined by the FS process. It removes duplicated and unnecessary functionality, improving system performance. Generally speaking, FS is a rare IDS operation. Nonetheless, several researchers have conducted their experiments using various FS techniques. This suggests that FS may raise IDS classification accuracy. The problem in using FS lies in finding a minimal feature subset of size S ($S < N$) given a feature set of size N while maintaining the prior accuracy. As a result, the FS problem needs a solution path [11, 34].

A partial solution, sometimes referred to as a subset, suggests that there is no connection between the elements or characteristics of the solution, and the decision of which element to add to the partial solution after the previous one needs not be influenced [11, 34]. FS problems can occasionally have solutions of varying proportions. Redefining the use of the ACO representation graph is the first stage in FS employing ACO.

Using an Ant Colony Optimization algorithm, we selected the attributes shown in Table 9.

Table 4.5: Features chosen using an Ant Colony Optimization algorithm [40, 41].

No	Feature
1	Dst Port
2	Flow Pkts/s
3	Fwd IAT Max
4	Flow, IAT Min,
5	Fwd Pkts/s

6	Fwd Header Len
7	Bwd Pkts/s
8	RST Flag Cnt
9	Fwd Seg Siz Min
10	Idle Std

4.4 Performance Evaluation

Our study employed a confusion matrix to evaluate a binary classification problem. Such problems often involve classes of interest, including a minority class and its opposite class, commonly called positives and negatives. In this context, several key performance metrics are used [35–37]:

True Positive (TP): these correctly anticipated positive results suggest that the actual and forecasted class values are accurate;

True Negative (TN): the fact that neither the actual class value nor the expected class value is positive is one of the precisely predicted negative values;

False Positive (FP): the situation where the expected class is valid but the actual class is false;

False Negative (FN): a situation where the actual class is positive but the projected class is negative;

Accuracy: the percentage of related samples among the retrieved samples is known as accuracy, another name for a positive value:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (21)$$

Precision: precision is the ratio of accurately expected observations to all anticipated positive observations concerning positive observations:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (22)$$

Recall (Sensitivity): By dividing the total number of true positives by the sum of the true and false negatives, the sensitivity is quantitatively calculated:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{TN}} \quad (23)$$

F-measure (F1 score): The F-measure functions as a sort of average between the parameters P (Precision) and R (Recall). P represents the system's accuracy when compared to the predicted data. We express the number of predicted data to all expected data for prediction as a ratio or r:

$$\text{F1Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (24)$$

4.5 Time Taken to Build a Model

The algorithm's duration is required to build the model using the training data.

4.6. Experimental Design

4.6.1 Dataset

The Communications Security Establishment (CSE) and the Canadian Institute for Cyber Security (CIC) have jointly developed the CSE-CIC-IDS2018 dataset for attack detection and prevention. This open-source dataset provides traffic patterns and various attacks found in network architecture [35].

There are 80 features in the dataset, including tags, most consisting of data in terms of statistics regarding internet traffic. The CICFlowMeter tool obtains these features by generating information based on network traffic flow. The dataset includes continual characteristics such as labels, timestamps, destination ports, and protocols. It includes six scenarios and fourteen types of assaults, such as botnet, denial of service (DoS), infiltration, Web attack, and DDoS [35,38].

Amazon Web Services (AWSs) offers the CSE-CIC-IDS2018 dataset for download; included are ten CSV files totaling over 16.2 million samples, representing ten recorded network flow days. The CICFlowMeter program extracted more than 80 features to produce the dataset. The CSE-CIC-IDS2018 dataset's Table 10 [35, 39] offers comprehensive details on the different kinds of assaults and safe traffic.

Table 4.6: Attack types and secure traffic in the dataset CSE-CIC-IDS-2018 [40, 41].

Category	Attack Types
Benign	
DDoS	HOIC
	LOIC-HTTP
	LOIC-UDP
Botnet	Bot
Brute Force	FTP
	SSH
Infiltration	Infiltration
Dos	Hulk
	SlowHTTPTest
	GoldenEye
	Slowloris
Web Attack	Web
	XSS
	SQL Injection

Table 11 shows the record count and attack percentage of the CSE-CIC-IDS2018 dataset.

Table 4.7: The CSE-CIC-IDS2018 dataset's record count [40, 41].

Traffic Class	Record Count	Share (%)
Benign	13,484,708	83.070%
HOIC	686,012	4.226%
LOIC-HTTP	576,191	3.550%
Hulk	461,912	2.846%
Bot	286,191	1.76%
FTP—BruteForce	193,360	1.191%
SSH—BruteForce	187,589	1.156%

Infiltration	161,934	0.998%
SlowHTTPTest	139,890	0.862%
GoldenEye	41,508	0.256%
Slowloris	10,990	0.068%
LOIC-UDP	1730	0.011%
Brute Force—Web	611	0.004%
Brute Force—XXS	230	0.001%
SQL Injection	87	0.0005%

There are 16,232,943 records in the CSE-CIC-IDS2018 dataset, with 83.07 percent (or records) consisting of benign traffic, while the remaining records represent malicious activity. This study used 80 percent of the records for training, while the remaining 20 percent was allocated for testing [38, 39, 35–37].

4.6.2. Experimental System Specifications

The experimental design for this study involved implementing the test procedure on a system comprising an AMD Ryzen 7 1800X eight-core processor operating at a clock speed of 3.60 GHz, paired with 32 GB of RAM and 128 GB allocated as swap space on the hard disk. The system employed an X64-based processor and ran on Windows 10 Enterprise as the operating system.

4.7. Results

Utilizing a range of machine learning algorithms, we performed experiments by dividing 80% of the dataset for training the system while allocating the remaining 20% to evaluate its performance. The feature selection techniques included Ant Colony Optimization (ACO) algorithms, the Flower Pollination Algorithm (FPA), and Artificial Bee Colony (ABC). Subsequently, these selected features were tested and evaluated with various machine learning algorithms. Tables 12–14 (values rounded to three decimal places) display the outcomes of these tests.

Table 4.8: Results of different algorithms using features selected by the Ant Colony Optimization (ACO) algorithms [40, 41].

Algorithm	Time Spent on a Model Build	Accuracy	Precision	Recall	F-Measure
AdaBoost M1	802.44 s	0.862	0.961	0.883	0.920
Bagging	13,969.96 s	0.988	0.996	0.989	0.992
BayesNet	638.07 s	0.965	0.968	0.989	0.979
CDT	1132.89 s	0.987	0.997	0.988	0.993
DTNB	18,733.06 s	0.986	0.996	0.988	0.992
IB1	9 s	0.978	0.986	0.988	0.987
J48	2858.5 s	0.988	0.997	0.988	0.993
KStar	3 s	0.979	0.989	0.987	0.988
Local Knn	324,075 s	0.984	0.993	0.988	0.990
Logistic	35,662 s	0.960	0.981	0.972	0.976
LogitBoost	2752 s	0.982	0.993	0.985	0.989
Multilayer perceptron	17,072 s	0.945	0.968	0.965	0.967
Random Tree	481.24 s	0.983	0.991	0.989	0.990
SMO	48,562	0.891	0.993	0.889	0.938

As can be seen in Table 8, after applying 14 different machine learning algorithms, according to the results obtained, we obtained the best accuracy in intrusion detection by applying the Bagging and J48 machine learning algorithms, of which 98.8% were able to detect different types of cyber threats and distinguish them from regular traffic; furthermore, regarding the shortest time to build the model, the construction time was three seconds using the Ant Colony Optimization feature selection method.

Table 4.9: Results of different algorithms using features selected by the Flower Pollination Algorithm (FPA) [40, 41].

Algorithm	Time Spent on a Model Build	Accuracy	Precision	Recall	F-Measure
AdaBoostM1	593.74 s	0.862	0.961	0.883	0.920
Bagging	14,089.45 s	0.987	0.996	0.988	0.992
BayesNet	438.8 s	0.978	0.988	0.986	0.987
CDT	991.14 s	0.987	0.997	0.988	0.992
DTNB	9329.23 s	0.986	0.996	0.988	0.992
IB1	7 s	0.978	0.986	0.987	0.987

J48	3003.97 s	0.987	0.997	0.988	0.992
KStar	1 s	0.980	0.990	0.986	0.988
Local Knn	445,905 s	0.983	0.992	0.988	0.990
Logistic	20,578 s	0.946	0.971	0.964	0.968
LogitBoost	2565 s	0.975	0.987	0.983	0.985
Multilayer perceptron	16,290 s	0.956	0.974	0.974	0.974
Random Tree	445.08	0.982	0.991	0.988	0.989
SMO	35,787 s	0.885	0.997	0.880	0.935

According to the results obtained after selecting features using the Flower Pollination Algorithm by applying different machine learning algorithms, the best result was an accuracy of 98.7 in detecting regular traffic and different cyberattacks. Moreover, the shortest model-building time was one second using the KStar algorithm.

Table 4.10 Results of different algorithms using features selected by the Artificial Bee Colony (ABC) [40, 41].

Algorithm	Time Spent on a Model Build	Accuracy	Precision	Recall	F-Measure
AdaBoostM1	430.44 s	0.862	0.961	0.883	0.920
Bagging	6679.11 s	0.966	0.995	0.988	0.992
BayesNet	230.1 s	0.970	0.978	0.985	0.982
CDT	475.84 s	0.986	0.996	0.988	0.992
DTNB	2941.84 s	0.983	0.996	0.984	0.990
IB1	5 s	0.976	0.985	0.986	0.986
J48	1299.04 s	0.986	0.996	0.988	0.992
KStar	2 s	0.977	0.987	0.985	0.986
Local Knn	3,430,073 s	0.982	0.991	0.987	0.989
Logistic	7212.36 s	0.892	0.998	0.887	0.939
LogitBoost	1287.00 s	0.973	0.984	0.985	0.984
Multilayer perceptron	11,008 s	0.926	0.998	0.920	0.957
Random Tree	394.52 s	0.981	0.990	0.987	0.989
SMO	12,496 s	0.890	0.992	0.888	0.938

Table 10 shows that the DTNB and J48 machine learning algorithms, combined with the Artificial Bee Colony feature selection approach, yielded the maximum accuracy

of 98.6, and that this strategy took the smallest amount of time to develop the model at two seconds.

Readers are encouraged to explore the following charts (Figures 5 and 6), portraying the accuracy and time needed for model construction employing various machine learning algorithms alongside diverse feature selection techniques. As shown in Figure 5, the highest accuracy among the three different feature selection methods, including Artificial Bee Colony (ABC), the Flower Pollination Algorithm (FPA), and Ant Colony Optimization (ACO), was that related to Ant Colony Optimization (ACO) with 98.8% using the Bagging and J48 machine learning algorithms. Moreover, the minimum model-building time was related to the Flower Pollination Algorithm (FPA), which was only 1 s using the KStar machine learning algorithm.

Chapter 5

CONCLUSION

This research aimed to build a model with minimal features from the 80 available in the CSE-CIC-IDS2018 dataset, achieving high accuracy in detecting Brute Force and Bot attacks while ensuring quick network traffic security. The CFSSubsetEval feature selection method identified five features for Brute Force and four for Bot attacks. Utilizing 11 machine learning techniques and five features, the BayesNet algorithm demonstrated 100% accuracy, taking 5.32 seconds to build the model.

The study introduced novel methodologies, combining intrusion detection approaches and employing Ant Colony Optimization, Flower Pollination Algorithm, and Artificial Bee Colony with other techniques for the first time. The research achieved over 98% intrusion detection accuracy using only five features and approximately 99% with ten features. This efficiency is highlighted by the model-building time, which is as short as 1 second with the KStar algorithm.

The effectiveness of feature selection and data-cleaning approaches in Intrusion Detection Systems was demonstrated. These approaches drastically reduced model-building time while achieving unprecedented accuracy levels. The study's insights provide valuable considerations for computer science researchers aiming to create and enhance Intrusion Detection Systems.

Using the CSE-CIC-IDS2018 dataset with 80 features and over 16 million records, the study showcased the impact of feature selection on accuracy, achieving 98% accuracy with only five features. Future research recommendations include refining feature selection methods and exploring data cleaning to optimize system performance based on network priorities—whether emphasizing time or accuracy.

In the context of computer networks, especially data centers, efficient Intrusion Detection Systems are crucial for timely and accurate identification of intrusions. The study emphasizes the significance of reducing the number of features for enhanced system efficiency, preventing slowdowns, reducing resource consumption, and increasing detection speed. The ultimate goal is to detect network assaults promptly, ensuring higher threat detection accuracy and instilling confidence in-network specialists.

REFERENCES

- [1] Ma C, Du X, Cao L. Analysis of multi-types of flow features based on hybrid neural network for improving network anomaly detection. *IEEE Access*. 2019; 7:148363–148380.
- [2] Ahmad I, Ul Haq QE, Imran M, Alassafi MO, AlGhamdi RA. An efficient network intrusion detection and classification system. *Mathematics*. 2022;10:530-539
- [3] Liu L, Wang P, Jun L, Liu L. Intrusion detection of imbalanced network traffic based on machine learning and deep learning. *IEEE Access*. 2021; 9:7550-7563.
- [4] Liu H, Lang B. Machine learning and deep learning methods for intrusion detection systems: a survey. *Appl. Sci*. 2019; 9:4396-4409.
- [5] Jovic A, Brkic K, Bogunovic N. A review of feature selection methods with applications. In: *Proceedings of the International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*; 2015 May 25–29; Opatija, Croatia.
- [6] Althubiti S, Nick W, Mason J, Yuan X, Esterkine A. Applying long short-term memory recurrent neural networks to intrusion detection. In: *Proceedings of SoutheastCon 2018*; 2018 Apr 19–22; St. Petersburg, FL, USA.

- [7] Esmaeili A, Ghorrati Z, Matson ET. Agent-based collaborative random search for hyperparameter tuning and global function optimization. *Systems*. 2023; 11:228-239.

- [8] Nisioti A, Mylonas A, Yoo P, Katos V. From intrusion detection to attacker attribution: a comprehensive survey of unsupervised methods. *IEEE Commun. Surv. Tutor*. 2018; 20:3369-3388.

- [9] Chockwanich N, Visoottiviseth V. Intrusion detection by deep learning with TensorFlow. In: *Proceedings of the 21st International Conference on Advanced Communication Technology (ICACT)*; 2019 Feb 17–20; Pyeongchang, Republic of Korea.

- [10] Pushpalatha KR, Karegowda AG. CFS based feature subset selection for enhancing classification of similar looking food grains: a-filter approach. In: *Proceedings of the 2017 2nd International Conference on Emerging Computation and Information Technologies (ICECIT)*; 2017. DOI: 10.1109/ICECIT.2017.8453403.

- [11] A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018). Registry of Open Data on AWS. Available online: <https://registry.opendata.aws/cse-CIC-ids2018>. Accessed 2 February 2023.

- [12] Arsalan S. FastTrafficAnalyzer: An efficient method for intrusion detection systems to analyze network traffic. *DUJE (Dicle Univ. J. Eng.)*. 2021; 12:565-572.

- [13]Government of Canada. Communications Security Establishment. Available online: <https://www.cse-cst.gc.ca/en>. Accessed 2 February 2023.
- [14]Canadian Institute for Cybersecurity. University of New Brunswick est.1785. Available online: www.unb.ca/cic/. Accessed 2 February 2023.
- [15]Alzughairi S, El Khediri S. A cloud intrusion detection system based on DNN using backpropagation and PSO on the CSECIC-IDS2018 dataset. *Appl. Sci.* 2023; 13:145-155.
- [16]Agbotiname LI, Oyedare T, Otuokere ME, Shettym S. Software intrusion detection evaluation systems: a cost-based evaluation of intrusion detection capability. *Commun Netw.* 2018; 10:211-229.
- [17]Shapoorifard H, Shamsinejad P. Intrusion detection using a novel hybrid method incorporating an improved KNN. *Int J Comput Appl.* 2017; 173:5-9.
- [18]Gu, G., Fogla, P., Dagon, D., Lee, W., & Skoric, B. (2006). Measuring Intrusion Detection Capability: An Information-Theoretic Approach. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security* (pp. 90–101). Taipei, Taiwan: ACM.
- [19]Yang Y, Webb GI. Discretization for naïve-Bayes learning: Managing discretization bias and variance. *Mach Learn.* 2009; 74:39-74.

- [20] Parsad A, Chandra SH. Machine learning to combat cyberattack: A survey of datasets and challenges. *J Def Model Simul*. 2022; 20:577-588.
- [21] Kiran MS, Babalik A. Improved artificial bee colony algorithm for continuous optimization problems. *J Comput Commun*. 2014; 2:108-116.
- [22] Yan X, Zhu Y, Zou W, Wang L. A new approach for data clustering using a hybrid artificial bee colony algorithm. *Neurocomputing*. 2012; 97:241-250.
- [23] Brajević I, Stanimirović P. An improved chaotic firefly algorithm for global numerical optimization. *Int J Comput Intell Syst*. 2018; 12:131–148.
- [24] Chu X, Cai F, Gao D, Li L. An artificial bee colony algorithm with adaptive heterogeneous competition for global optimization problems. *Appl Soft Comput*. 2020; 93:106391.
- [25] Sahin O, Akay B, Karaboga D. Archive-based multi-criteria Artificial Bee Colony algorithm for whole test suite generation. *Eng Sci Technol*. 2021; 24:806–817.
- [26] Yang X, Karamanoglu M, He X. Flower pollination algorithm: A novel approach for multiobjective optimization. *Eng Optim*. 2014; 46:1222–1237.
- [27] Cui W, He Y. Biological Flower Pollination Algorithm with Orthogonal Learning Strategy and Catfish Effect Mechanism for Global Optimization Problems. *Math Probl Eng*. 2018; 2018:6906295.

- [28]Dorigo M, Blum C. Ant colony optimization theory: A survey. *Theor Comput Sci.* 2005; 334:243–278.
- [29]Montemanni R, Gambardella LM, Rizzoli AE, Donati AV. Ant Colony System for a Dynamic Vehicle Routing Problem. *J Comb Optim.* 2005; 10:327–343.
- [30]Sahin O, Akay B, Karaboga D. Archive-based multi-criteria Artificial Bee Colony algorithm for whole test suite generation. *Eng Sci Technol.* 2021; 24:806–817.
- [31]Blum C, Dorigo M. The hyper-cube framework for ant colony optimization. *IEEE Trans Syst Man Cybern.* 2004; 34:1161–1172.
- [32]Leguizamon G, Michalewicz Z. A New Version of Ant System for Subset Problems. In: *Proceedings of IEEE Congress on Evolutionary Computation*; 1999 Jul 6–9; Washington, DC, USA.
- [33]Zhang L, Xu C. An Intrusion Detection Model Based on Convolutional Neural Network and Feature Selection. In: *Proceedings of the 5th International Conference on Artificial Intelligence and Big Data (ICAIBD)*; 2022 May 27–30; Chengdu, China. pp. 162–167.
- [34]Abdulhammed R. Intrusion Detection: Embedded Software Machine Learning and Hardware Rules Based Co-Designs. [Doctoral Dissertation]. Bridgeport, CT, USA: University of Bridgeport; 2019.

- [35] Alzughhaibi S, El Khediri S. A Cloud Intrusion Detection Systems Based on DNN Using Backpropagation and PSO on the CSECIC-IDS2018 Dataset. *Appl Sci.* 2023; 13:145–155.
- [36] Canadian Institute for Cybersecurity. University of New Brunswick est.1785. Available from: www.unb.ca/cic/ (accessed on 2 February 2023).
- [37] Canadian Institute for Cybersecurity. CSE-CIC-IDS2018 on AWS. Available from: www.unb.ca/cic/datasets/ids-2018.html (accessed on 2 February 2023).
- [38] Arsalan S. FastTrafficAnalyzer: An Efficient Method for Intrusion Detection Systems to Analyze Network Traffic. *DUJE (Dicle Univ. J Eng)*. 2021; 12:565–572.
- [39] Government of Canada. Communications Security Establishment. Available from: <https://www.cse-cst.gc.ca/en> (accessed on 2 February 2023).
- [40] Najafi Mohsenabad, H., & Tut, M. A. (2024). Optimizing cybersecurity attack detection in computer networks: A comparative analysis of bio-inspired optimization algorithms using the CSE-CIC-IDS 2018 dataset. *Applied Sciences* (Basel, Switzerland), 14(3), 1044. doi:10.3390/app14031044.
- [41] Mohsenabad, H. N., & Tut, M. A. (2023). Achieving the Highest Detection Rate and the Shortest Time in Building a Model for Intrusion Detection in Computer Network Intrusion Detection Systems. *International Journal of Advance*

Computational Engineering and Networking (IJACEN), 55–60. doi: IJACEN-IRAJ-DOIONLINE-19501.