Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department

IMPROVING NETWORK INTRUSION DETECTION SYSTEM BASED ON A PROACTIVE KNOWLEDGE DISCOVERY TECHNIQUE

A Dissertation

Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Technology-Software

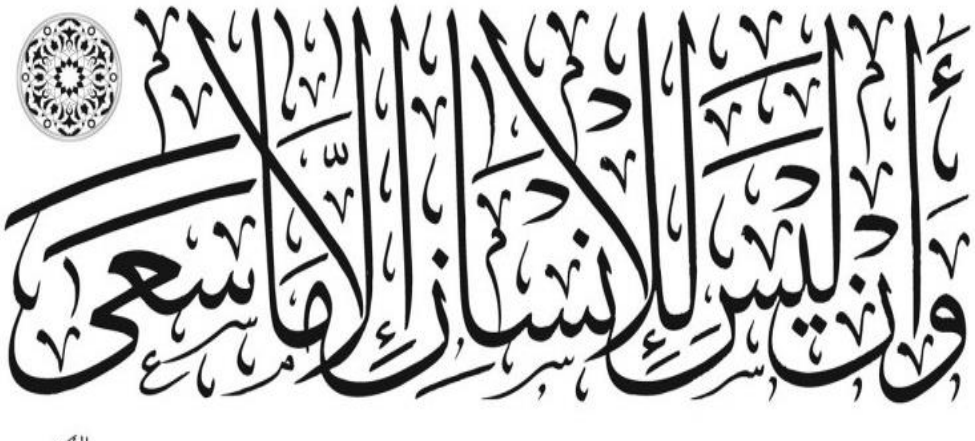
SALAM SAAD MOHAMED ALI ABED ALAAH

Supervised by

Prof. Dr. Rafah Mohammed Kadhum Khudair

2022A.D. **1443 A.H.**

Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department



2022A.D. **1443 A.H.**

Supervisor Certification

I certify that the dissertation entitled (**Improving Network Intrusion Detection System based on A Proactive Knowledge Discovery Technique**) was prepared under my supervision at the department of Software/ College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Doctor of Philosophy in Information Technology-Software.

Signature:

Supervisor Name: **Prof**. **Dr.Rafah M. Almuttairi**

Date: / /2022

The Head of the Department Certification

In view of the available recommendations, I forward the dissertation entitled "**Improving Network Intrusion Detection System based on A Proactive Knowledge Discovery Technique**" for debate by the examination committee.

Signature:

Name**: Asst. Prof. Dr. Ahmed Saleam**

Head of Software Department

Date: / /2022

Certification of the Examination Committee

We, the undersigned, certify that (Salam Saad Mohamed Ali Abed Alaah) candidate for the degree of Doctor of Philosophy in Information Technology-Software, has presented his dissertation of the following title (**Improving Network Intrusion Detection System based on A Proactive Knowledge Discovery Technique**) as it appears on the title page and front cover of the dissertation that the said dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: (12 / 6 /2022).

Signature:
Name: Dr. Sattar B.Sadkhan
Title: Professor
Date: / / 2022
(**Chairman**)

Signature:
Name: Dr. Wesam S.Bhaya
Title: Professor
Date: / / 2022
(**Member**)

Signature:
Name: Dr. Ibrahim Ahmed Saleh
Title: Professor
Date: / / 2022
(**Member**)

Signature:
Name: Dr. Haider K.Hoomod
Title: Professor
Date: / / 2022
(**Member**)

Signature:
Name: Dr. Mehdi Ebady
Title: Assistant Professor
Date: / / 2022
(**Member**)

Signature:
Name: Dr. Rafah M.Almuttairi
Title: Professor
Date: / / 2022
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Dr. Hussein Atiya Lafta
Title: Professor
Date: / / 2022
(**Dean of Collage of Information Technology**)

Declaration

I hereby declare that this dissertation entitled (**A Proactive Knowledge Discovery Techniques for Improving Network Intrusion Detection System)** that submitted to University of Babylon in partial fulfillment of requirements for the degree of Doctorate of Philosophy in Information Technology/Software has not been submitted as an exercise for a similar degree at any other university. I also certify that the work described here is entirely my own except for experts and summaries whose sources are appropriately cited in the references.

Signature:

Name: Salam Saad Mohamed Ali

Date: / / 2022

Dedication

To The Messenger of Mercury and Humanity
(Allah's Blessings be upon him and his household))

To his cousin, Imam Ali Ibn Abi Talib (AS)
To his daughter, Fatima Al Zahraa (AS)
To his grandsons, Imam Al Hassan and Imam Al Hussein (A.S)

All Loved by Allah
Allah's Blessings be upon him and his household

Acknowledgement

First and foremost, all praise and thanks be to almighty **Allah**, the most gracious and the most merciful, for making everything possible by giving me strength, confidence, and courage to accomplish this work. Extending thanks be to His prophet ***"Mohammed"*** (Peace be upon him and his household). Next, I wish to express my sincere gratitude to my supervisor ***Prof. Rafah M. Almuttairi****, Ph.D.,* for her guidance and direction, in this work. She gave me many interesting, valuable and sincere feedbacks throughout her supervision. I have greatly benefited from her detailed comments and insights that helped me clarify ideas in *"A Proactive Knowledge Discovery Techniques for Improving Network Intrusion Detection System".*

I would also like grant my big thanks to all lecturers and staff members, at IT-College / University of Babylon, who were kind enough to give me their precious time and assistance, without which I would not have been able to complete my Ph.D. work. I am indebted and thankful to the Dean of IT College. In addition, I wish to thank the ministry of higher education and scientific research of Iraq for their assistance.

Last but not least, words cannot express my gratitude and indebtedness to my father, and to my family, especially my sympathetic, compassionate and beloved mother, my dear brother, my sisters, my faithful wife. Words cannot describe their constant love, care, concern, patience, and direction in every aspect of my life throughout the years of my study. I'm forever thankful, grateful, and indebted to them. May Allah bless them! I dedicate the accomplishment of this dissertation to my father, my affectionate mother, and to the twin of my spirit, my wife.

Salam Saad Mohamed Ali

Abstract

Any method, process, or means of maliciously degrading network security can be defined as a network attack. Network Intrusion Detection Systems (NIDS) are critical in detecting network intrusions in the domain of national security. Network Intrusion Detection Systems are based on traditional models such as machine learning or clustering algorithms that suffer from low accuracy of multi classifications and rely heavily on reducing the imbalance of low rate attack in network data.

In this dissertation, a Network Intrusion Detection System developed based on combination techniques of unsupervised data mining clustering and supervised machine learning are implemented to detect multiple types of attacks in network flow.

The proposed system is composed of four main stages. The first stage is preprocessing, consisting of two main sub-steps: One-Hot Encoding and Min-Max normalization methods.

The second stage is selecting the relevant features using a proactive BAT swarm intelligence as a feature selection method. The duties of the feature selection algorithm are to eliminate redundant and irrelevant attributes and reduce the impact of these attributes on the homogeneity of distribution of the training data in the clustering and the final prediction process.

The third stage is an anomaly detection model, which employs an improved density peak clustering algorithm to group data samples as regular connections from attacks to break the dataset's imbalance. Then using the artificial neural network classifier to predict the inflexible and low rate attack classes with the help of fractal fuzzy membership degree.

System evaluation is the fourth stage, in which the proposed system is evaluated using performance indicators like accuracy, detection rate (recall), precision and F1 score. Two network traffic datasets are used: NSL_KDD and real-world UNSW_NB15. Each of them consists of both regular and multi-type

attacks packets. The obtained results have shown that the best detection system is achieved by using combination techniques based on improved Density peak cluster (IDPC) algorithm and ANN classification algorithm with the help of fractal fuzzy membership degree function. The overall accuracy for NSL_KDDTest-21 dataset is 82.47%,the precision 83.44%,the recall(detection rate) is 82.47%, the f1.score 82.95% and the overall accuracy for NSL_KDDTest+ dataset is 92.24%, the precision 92.68%,the recall(detection rate) is 92.24%, the f1.score 92.46%. While the overall accuracy in UNSW_NB15 dataset is 95.59%, the precision 96.68%, the recall (detection rate) is 95.59%, the f1.score 96.13%.This done in an efficient time (in seconds).

Declaration Associated with this Thesis

Some of the works presented in this dissertation have been published as listed below.

1. **"A Proactive Model for Optimizing Swarm Search Algorithms for Intrusion Detection System"**. Academics Syndicate International Conference for Pure and Applied Sciences (IICPS), Babylon Branch, Babylon, Iraq at 5-6 December 2020. **IOP Conference Series.**
   (Scopus - Q3) (Publication)

2. **"Enhancing Density Peak Clustering Technique for Intrusion Detection System".** Periodicals of Engineering and Natural Sciences, Vol. 9, No. 2, June 2021, pp.965-975, ISSN 2303-4521. **PEN Journal**.
   (Scopus – Q2) (Publication)

Table of Contents

List of Algorithms

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Description
AID	Anomaly Intrusion Detection
ANN	Artificial Neural Network
BA	Bat Algorithm
CIA	Confidentially, Integrity, and Availability
DPC	Density Peak Clustering
DT	Decision Tree
Dos	Denial of Service
DR	Detection Rate
FFM	Fractal Fuzzy Membership
FN	False Negative
FP	False Positive
FTP	File Transfer Protocol
HTTP	Hyper-Text Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IDS	Intrusion Detection System
KDD	The Knowledge Discovery and Database
KNN	K-Nearest Neighbors
MID	Misuse Intrusion Detection
ML	Machine Learning
NIDS	Network Intrusion Detection System

Abbreviation	Description
NB	Naïve Bayes
R2L	Remote to Local
SMTP	Simple Mail Transfer Protocol
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
U2R	User to Root
UDP	User Datagram Protocol
UNSWNB15	University of New South Wales Network Based 2015

Chapter One

Introduction

Chapter One
Introduction

1.1 Introduction

The publicity of the Internet led to the need for sharing and accessing information, and such popularity resulted in increasing the amount of information and consuming resources. Data is stored in the cloud or different databases; thus, it is crucial to protect them and ensure that it is secure [1]. Personal information is an essential part that must be secured against attacks [2]. Nowadays, many organizations adopt various strategies to authenticate and authorize access for data, which should be kept secure and confidential [3].

Network attacks are growing in number and variety: ransomware is increasing like never before, and zero-day exploits are becoming so critical that they are receiving media attention [4].

Antivirus and firewalls are no longer enough to protect a company network, which needs to be covered with several levels of security. On the other hand, the massive increase in computer network usage and the creation of applications that run on various platforms has drawn attention to network security [5].

1.1.1 Network Attacks

Many attacks infect the network and try to access their information and essential sources. To gain unauthorized access to a system, attackers typically take five stages [6]. Figure (1.1) illustrates the main steps of an attacking approach.

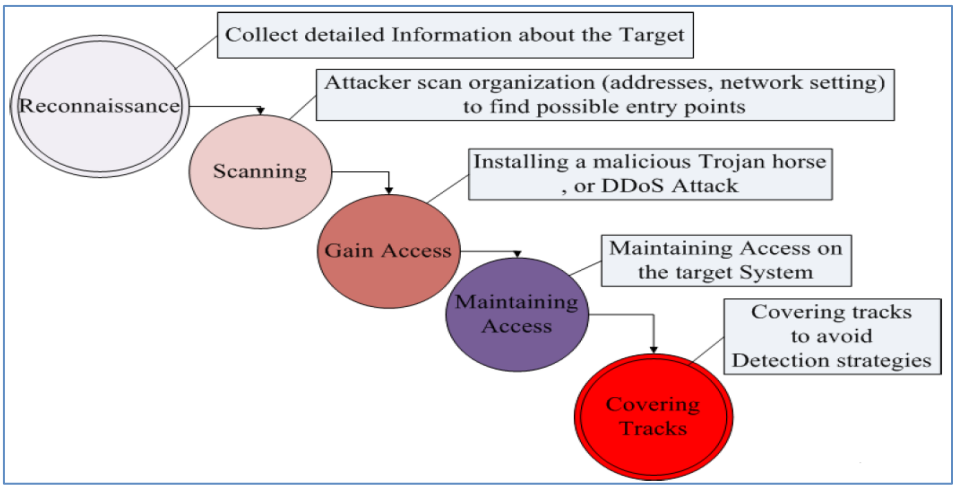


Figure (1.1): Five Steps for Attacking Approach

One of these attacks is flooding attacks, often referred to as SYN flooding, which works at the transport protocol (TCP) layer. Flooding attacks have become a severe issue since they cause a crisis of confidence and privacy, and they result in criminal actions being done against a company. Flooding attacks are intended to take down a network or service in a short period of time, causing the host memory buffer to fill and putting the host inaccessible for normal operations [7]. Figure (1.2) shows flooding attacks classification.

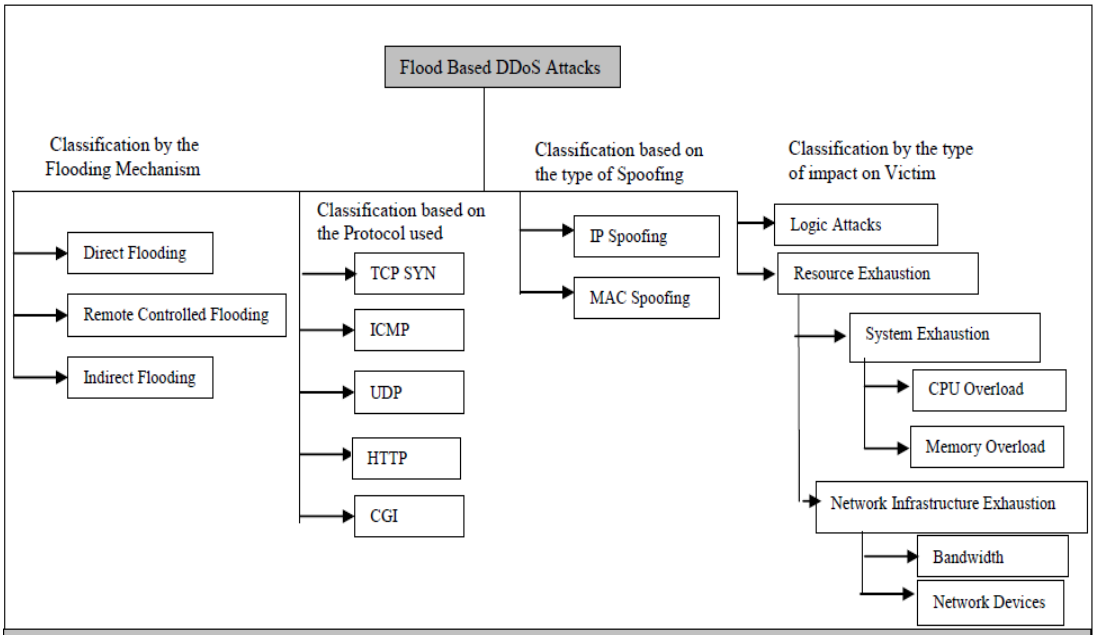


Figure (1.2): Flooding Attacks Classification

1.1.2 Network Intrusion Detection System (NIDS)

Network Intrusion detection systems (NIDS) are becoming increasingly important tools as the number of attacks on network transactions increases. NIDS monitors and analyzes events during data transactions in network traffic to examine a network for potential intrusions [8]. In order to detect intrusions, efficient network intrusion detection systems must collect massive amounts of data transactions, which may contain sensitive information [9].

In general, NIDS is classified into two types: the first type is a **S**ignature-based or misuse **I**ntrusion **D**etection **S**ystem (**SIDS**) that detects attacks in network traffic based on recognized patterns. These patterns revealed a series of activities that were collected and saved in a database. The pattern database is in charge of identifying network behavior by detecting only attacks whose patterns have already been saved in this database [10].

Anomaly-based **I**ntrusion **D**etection **S**ystems (**AIDS**) are the second type. This system is based on network behavior as the primary parameter for analyzing network transactions. The learned system will accept network transactions with predefined behavior; otherwise, the system will issue an alert. The system administrator is responsible for the system's accepted behavior specifications, which can be predefined or realized by the system [11]. The drawback of signature-based SIDSs is that they cannot detect unknown attacks, so AIDS is recommended in many studies. Figure (1.3) shows the NIDS (classifications, techniques, and detection mechanism) [12].

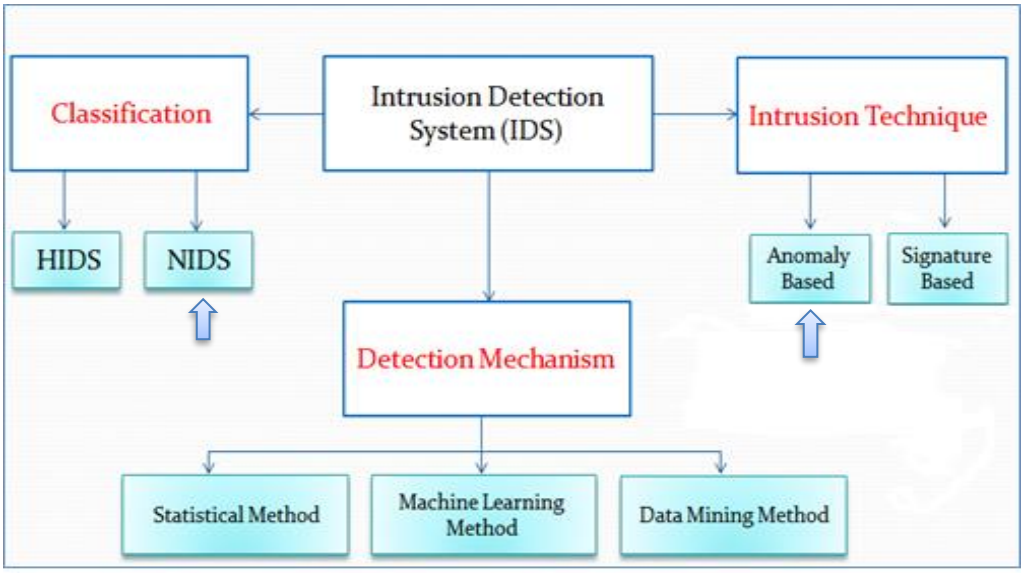


Figure (1.3): IDS Classification, Techniques, Detection Mechanism

1.1.3 Anomaly Detection based on Combination Techniques

In **A**nomaly-**N**etwork **I**ntrusion **D**etection **S**ystems (**ANIDS**), a model of legitimate network or host profile of events is designed first. Then prediction based on the model is compared with actual measurements [13].

In this dissertation, a combination system based on unsupervised clustering technique and supervised artificial neural network technique have been designed and implemented to analyze and classify low rate attacks in many steps. Figure (1.4) shows NIDS detection strategies with Hybrid Techniques**.** On the other hand, the unsupervised technique extract new usable knowledge from an extensive data set. It will group similar objects together and separate different objects using a defined dissimilarity measure. In contrast, supervised technique are used to predict the inflexible and low rate attack classes with the help of fractal fuzzy membership degree.

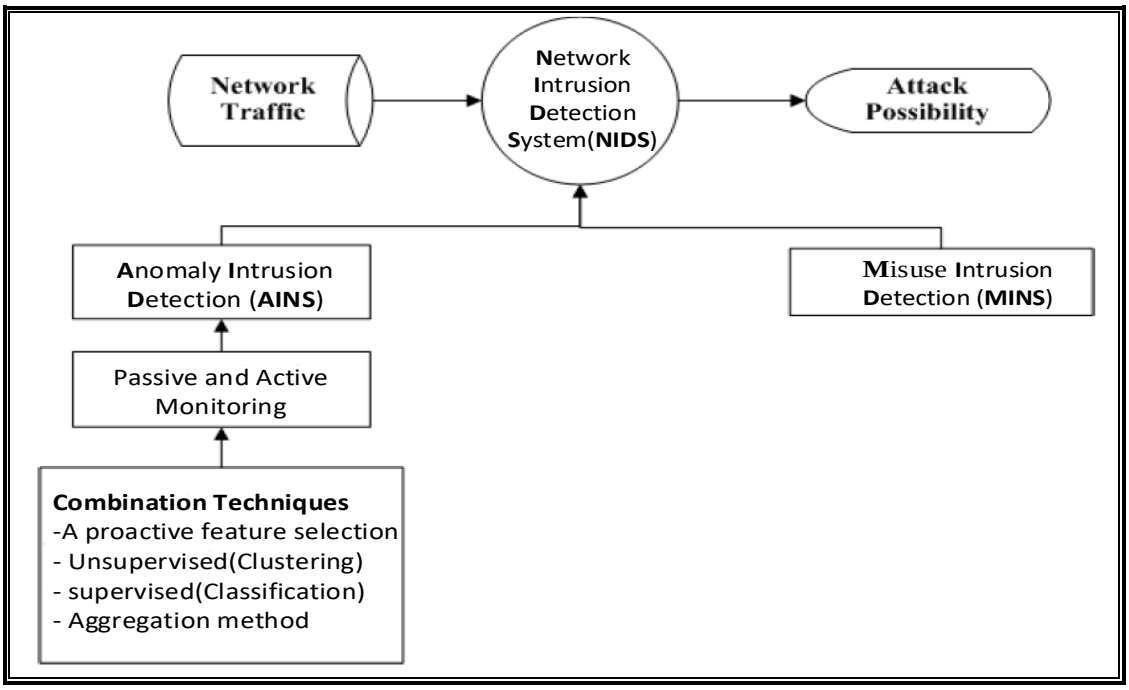


Figure (1.4) NIDS Detection Strategies with Combination Techniques

1.2 Problem Statement

The problems based on the NIDS components are divided into the following:

1. The attack packets are sent within the normal range which make the detection is difficult at early stage in the firewall device. Moreover, the firewall passes this attack as genuine packets which lead to early damage of an organization resources. The **F**alse **A**larm **R**ate (FAR) and **D**etection **R**ate (DR) are not changed during the attacking time which causes the defect in the overall system accuracy measurements[.
2. Choosing the relevant network features is a significant challenge in today's network environments. Most machine-learning algorithms suffer from high data dimensionality. logically, increasing data dimensionality will increase process complexity, time and impact system accuracy.

3. The multi-classification and Detection rate of low rate attacks is a big challenge because the actual network environment is imbalanced, which implies that threats records in network traffic are less than regular records; at the same time, the classifier is biased toward more commonly occurring data, the detection rate of low-rate attack recordings will be lowered .

1.3 Research Aims

The main objectives of this research are to address the issues of taking security consideration into account when designing network intrusion detection systems. Developing such resilience approach should play an integral part in designing, classification multi-type attacks and detection schemes using an intelligent combination techniques based on IDPC-ANN. The research objectives involve the following major tasks:

1. Suggesting a proactive feature selection model using BAT swarm optimization algorithm to select relevant features and increase the homogeneity of the distribution the training data in clustering technique.
2. Suggesting a cluster analysis that can solve the imbalance problem of network environments and increase the detection rate of low rate attacks using one of the data mining techniques such as the DPC algorithm. In addition, it should be efficient to be implemented with high dimensional data.
3. Designing a combination model to deal with multi classification attacks aiming to improve the detection rate of low rate attacks such as (R2l, U2r, Analysis, Backdoors, Worms, and Shellcode) and analyzing their performance against security breaches using

performance evaluation such as Accuracy, Detection Rate (recall) and others.

4. Suggesting a fractal mechanism to minimize the time and computational complexity of the testing phase.
5. Validating the proposed system with other works in terms of multi-classification attacks, accuracy, recall, precision, F1.score and detection rate of low rate attack to show the effectiveness of the proposed system.

1.4 Research Contribution

The main contributions of this research are:

1- A proactive feature selection model is implemented to reduce the dimensionality of the given data attributes while improving the system's accuracy.

2- Using a data mining cluster analysis (DPC) algorithm to solve the imbalance problem of multiple classes and increase the detection rate of low rate attacks.

3- Complex dataset from real network environment is analyzed by improving (DPC) algorithm in case of its distance metric by using Gaussian kernel distance instead of Euclidian distance.

4- Time consumed and computational complexity are reduced by proposed a new fuzzy membership matrix FFM based on the fractal factor.

5- Defining performance evaluation of the proposed scheme in terms of multi-classification attacks types and Detection rate of low rate attacks. The obtained results are validated with other works to show the effectiveness of the proposed system.

1.5 Related Works

In this section, several researchers' works that have related intrusion detection system techniques are discussed. Many researchers adopted data mining clustering techniques as point assignment and neural or deep network classifiers as hybrid techniques for building IDS. In addition, present the hybrid techniques, performance evaluation, and dataset name.

In 2016, Rana et al. [14] proposed a new fuzziness based semi-supervised learning strategy to improve IDS classifier performance by combining unlabeled data with a supervised learning algorithm. To construct a fuzzy membership vector, a single hidden layer feed-forward neural network (SLFN) is trained. Then used to classify unlabeled input into low, medium, and high fuzziness categories using the fuzzy quantity. After adding each category to the original training set separately, the classifier is retrained. Experiments on the NSL_KDDTest+ and NSL_KDDTest-21 testing data with accuracy 84.12% and 68.82% respectively. In their studies, the authors did not discuss the detection rate of low rate class examples like R2l or U2r.

In 2016, Tao Ma et al.[15], SCDNN is a new technique that combines spectral clustering and deep neural network algorithms. The dataset is initially partitioned into k subgroups using cluster centers, as in SC, based on sample similarity; following that, the distance between data points in a testing set and the training set is calculated and input into the deep neural network method for intrusion detection based on similarity features. Their study results, which used 20% randomly selected training sets and the NSL KDDTest-21, with accuracy 44.55% and shows a low detection rate of low rate attacks such as R2l and U2r.

In 2016, Nour et al.[16], the authors show how the UNSW_NB15 dataset can be used to evaluate Network Anomaly Detection Systems in three ways. The statistical analysis of the observations and attributes will come first. The

second step involves examining feature correlations. Finally, five machine-learning classifiers are evaluated for accuracy and false alarm rates, and the results are compared to the KDD99 data set. According to the results of the experiments, of UNSW-NB15 accuracy is 78.47%.

In 2017, Chuanlong et al.[17], present a recurrent neural network-based deep learning approach for intrusion detection (RNN-IDS). They also examine how the proposed model performs in binary and multiclass classification and how the size of the learning rate and the number of neurons affect its performance. The experiments showed that RNN-IDS is ideally suited to building a high-accuracy classification model and that its performance accuracy in multiclass classification using the NSL_KDDTest+ and NSL _KDDTest-21 are 81.29% and 64.67% respectively, which are better than standard machine learning classification approaches.

In 2018, Hebatallah et al. [18], proposed a framework that uses filter and wrapper methods to apply several strategies for feature selection. Based on the UNSW NB15 dataset, the J48 and Nave Bayes algorithms are employed as classifiers. The best methodology, according to the experimental data, is to combine 18 features from the GR ranking method with J48 as a classifier, which results in an accuracy of 88.3%.

In 2019, Yanqing et al. [19], combined the modified density peak clustering method with deep belief networks to build a hybrid intrusion detection technique. MDPCA is a technique for detecting similarities in complex and large-scale network data. MDPCA is used to divide the training set into multiple subsets with similar sets of attributes in order to reduce the size of the training set and eliminate the imbalance of training samples. To train, each subgroup is given its own sub-DBNs classifier. Fuzzy membership weights are being used to aggregate the output of all sub-DBNs classifiers. The MDPCA- DBN's performance is evaluated using the NSL_KDDTest+,

NSL _KDDTest-21, and UNSW_NB15 datasets with accuracy 80.82%, 66.18% and 90.21% respectively.

In 2020, Chaofei et al. [20], proposed SAAE-DNN, an intrusion detection system based on the stacked auto-encoder (SAE), attention mechanism, and deep neural network (DNN). The attention mechanism helps the network to gain strong framework of intrusion detection because the SAE represents data with a latent layer. The SAAE encoder can extract features automatically, but it can also improve DNN detection accuracy by initializing the weights of DNN potential layers. The NSL KDD dataset was used to assess the performance of SAAE-DNN in binary and multi-classification. In multi-classification, the SAAE-DNN model outperforms machine-learning approaches such as random forest and decision tree, identifying regularly and attacking symmetrically. On the KDDTest-21 test set, the results of binary-classifications and multi-classifications are 82.14% and 77.57%, respectively.

In 2020, Sydney et al. [21], The UNSW-NB15 intrusion detection dataset, which used to train and test the models, was analyzed. They apply a filter-based feature reduction technique using the XGBoost algorithm. They investigated both binary and multi-classification scheme in the experiment. The authors found that using the XGBoost-based feature selection strategy allow classifiers to improve their test accuracy to 77.51%.

In 2021, CHAO et al. [22], developed a intrusion detection system based on distributed k-means, RF, and deep learning. To overcome IDS challenges that were time consuming and inefficient. In the case of categorizing regular events and attack events, deployed distributed computing on the Spark platform. Then the abnormal occurrences are classified into different attack types using convolutional neural networks (CNN), long short-term memory (LSTM), and other deep learning methods. The performance of multi-target classification in the NSL_KDD dataset, in particular, can reach 85.24%.

In 2021, Isra et al [23], build two models for network intrusion detection based on deep learning and two data preprocessing approaches, a simplistic preprocessing approach and a hybrid two-step preprocessing strategy, to generate significant features. The CNNs algorithm is used in these models. They create binary and multiclass classification models. Deep feature synthesis is used to combines dimensionality reduction with feature engineering in the suggested method. Two benchmark datasets, the NSL_KDD dataset and the UNSW_NB15 dataset, are used to evaluate the models' performance accuracy that reach 81.44% and 80.51% respectively.

Table (1.1) summarizes of all the related works and illustrates the techniques, datasets, and measurement of evaluation.

Table (1.1): Summary of the Mentioned Related Works

Ref, Year	Technique	Dataset	measurement of evaluation
[20] (2016)	Fuzziness based semi-supervised with single hidden layer feed-forward neural network (SLFN)	NSL_KDDTest-21, NSL_KDDTest+	Accuracy
[21] (2016)	a Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm (SCDNN)	NSL_KDDTest-21	Accuracy and detection rate of U2r and R2l
[22] (2016)	Expectation-Maximization (EM) clustering	UNSW_NB15	Accuracy
[23] (2017)	Recurrent Neural Networks (RNN-IDS)	NSL_KDDTest-21, NSL_KDDTest+	Accuracy and detection rate of U2r and R2l

[24] (2018)	GR ranking method and J48 algorithm	UNSW_NB15	Accuracy
[25] (2019)	Modified Density Peak Clustering and Deep Belief Networks(MDPCA-DBN) , DBN algorithm	NSL_KDDTest-21, NSL_KDDTest+	Accuracy and detection rate of U2r and R2l
		UNSW_NB15	Accuracy and detection rate of Analysis, Backdoors, Worms, and Shellcode
[26] (2020)	SAAE encoder with DNN	NSL_KDDTest-21, NSL_KDDTest+	Accuracy
[27] (2020)	ANN-XGBoost	UNSW_NB15	Accuracy
[28] (2021)	Kmeans +RF +CNN +LSM	NSL_KDDTest+	Accuracy and detection rate of U2r and R2l
[29](2021)	Multi-classification Convulsion neural network With deep feature synthesis(MCNN-DFS)	NSL_KDDTest+	Accuracy

In this dissertation, the first phase is preprocessing, which includes one hot encoding method and a Min-Max normalization method for data discretization. In the second phase, implement a proactive BAT algorithm for feature selection for two datasets (NSL_KDD and UNSW_NB15). The feature selection aims to eliminate redundant and irrelevant attributes, which helps increase the homogeneity of training data distribution of the clustering process and reduce their impact on the prediction method.

A combination system was applied in the third phase, including the improved density peak clustering method (IDPC) and an artificial neural network (ANN). IDPC divides the training set into two subsets with similar features based on Gaussian kernel distance, lowering training set size and breaking the imbalance. Each subgroup is assigned to a different sub-ANN

classifier to train. These sub-ANN classifiers are capable of doing well in classification. The output aggregation method will be employed based on fuzzy membership weights to combine the output of all sub-ANN classifiers.

1.6 Dissertation Outline

This chapter introduces the general concepts of the dissertation landscape. It presents the main problem of network attacks and how the proposed system questions and objectives are introduced. The contribution of the proposed system is illustrated to address the main problems of network attack detection using data mining techniques. In addition, the related works are discussed to illustrate the recent network attacks detecting methods with data mining techniques. The rest of this dissertation is organized as follows:-

Chapter Two: reviews the network threats, network security background, the security techniques related to networks attacks in NIDS, and the data mining techniques for anomaly detection.

Chapter Three: provides the details of the proposed system, including the block diagram and the algorithms to implement the system.

Chapter Four: discusses the experimental results of the proposed system and compares them with other works.

Chapter Five: explains the conclusions of the dissertation and suggestions for future work.

Chapter Two

Theoretical Background

Chapter Two

Theoretical Background

2.1 Introduction

Confidentiality, integrity, and availability (CIA trinity) are three key concepts in information security. A cyber-attack is defined as any illegal activity involving one, two, or three components of an information system. Figure (2.1) show the Classical CIA triad of Information Security[24].

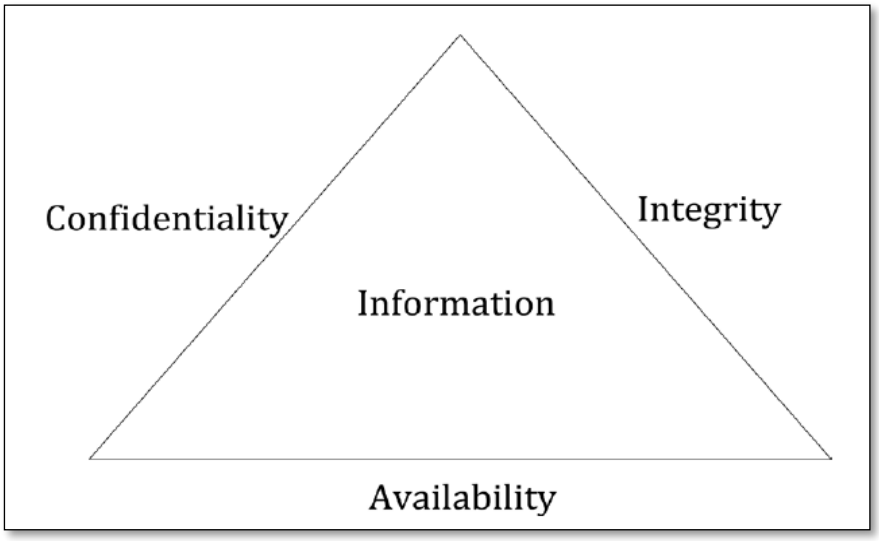


Figure (2.1): Classical CIA Triad Of Information Security [25]**.**

1. Confidentiality is defined, as an unauthorized person is able to read and take advantage of information stored in the computer. This category of concern sometimes extends to "traffic analysis," in which the intruder only observes the patterns of information use. From those patterns, the intruder can infer some information content. This category also includes the unauthorized use of a proprietary program [24] [26].

2. Integrity is the property of accuracy and completeness. Integrity ensures that unauthorized parties do not change information[27] .
3. Availability is the property of being available and accessible by an authorized entity on demand. [33]. Service denial (DoS) attacks are a common type of attack on availability [28].

However, a network incursion is a collection of actions that seeks to compromise the integrity, confidentiality, or availability of a resource. As a result, intrusion detection is necessary as an effective wall to protect network systems. Intrusion detection is valuable for identifying successful intrusions and providing critical information for timely countermeasures[29].

In the following two sections, the header information in Transmission Control Protocol (TCP) and network intrusion detection technologies are presented to correctly identify certain packets features that differentiate them from the malicious packets using the detector method.

2.2 Transmission Control Protocol (TCP)

The identify TCP/IP information header is necessary to understand the detection of an attack. TCP/IP is a set of rules for communication between source and destination devices at different TCP/IP suite levels. The TCP protocol is a connection-oriented protocol that enables reliable and ordered transmission on Internet traffic [30]. Unlike the UDP protocol, which is a connectionless protocol [31]. The Transmission Control Protocol/Internet Protocol (TCP/IP) suite layers have been presented in Figure (2.2) [32].

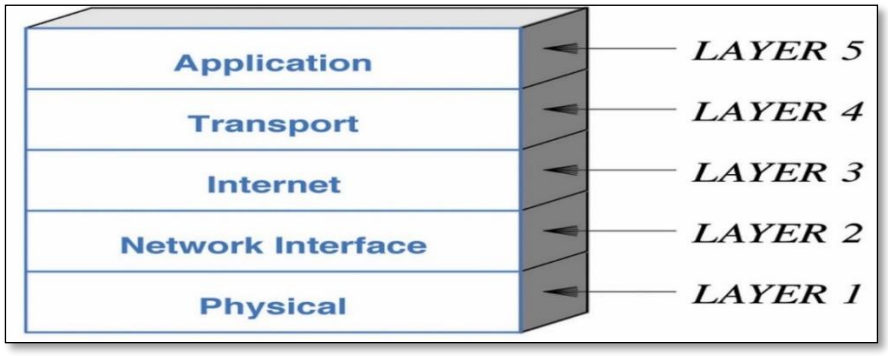


Figure (2.2): TCP/IP Protocol Suite

Two mechanisms occur during the data transmission between the source and destination devices: encapsulation and de-encapsulation, respectively. When the source device starts to send data, it encapsulates through the application layer, transport layer, network layer, and finally to the data link layer. Each layer adds a piece of data on each layer, such as (data, packet, segment, frame and bits). When the host at the destination site receives these bits, it goes in reverse order from the physical layer to reach the application layer. Hyper-Text Transfer Protocol (HTTP), File Transfer Protocol (FTP), Telnet, ping, Address Resolution Protocol (ARP) and Simple Mail Transfer Protocol (SMTP) are application protocols that utilize TCP/IP suite [32]. The IPv4 header contains information that is important to establish a connection. It consists of *thirteen* obligated fields and an optional field. Figure (2.3) illustrates the IPv4 header segment [33].

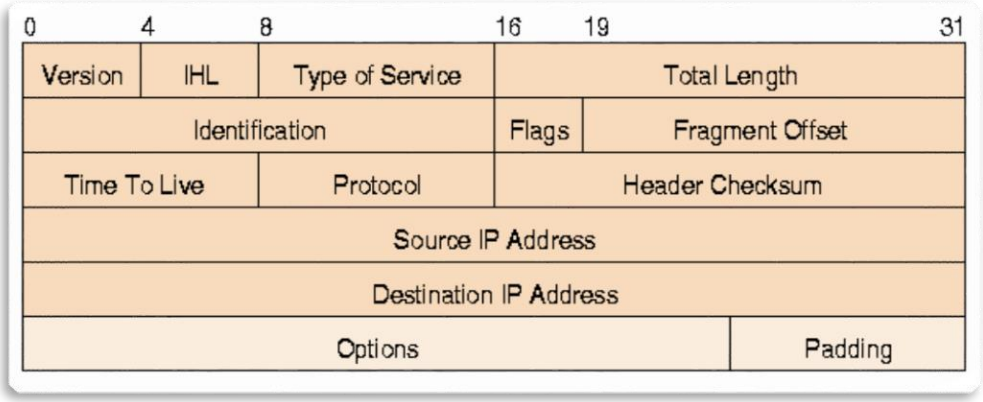


Figure (2.3): IPv4 Header Segment

The IPv4 header features describe the information in each connection between the source and destination devices. Figure (2.4) presents the information header for both TCP and UDP segments, respectively [34].

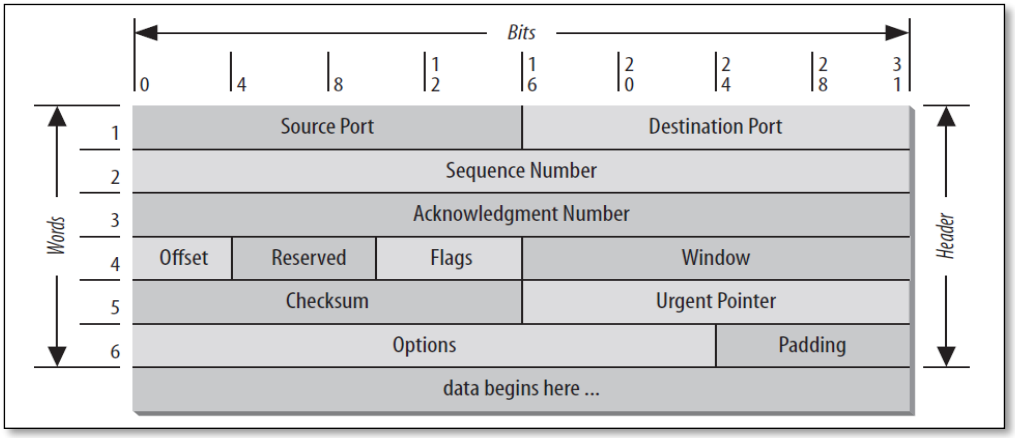


Figure (2.4a): TCP Header

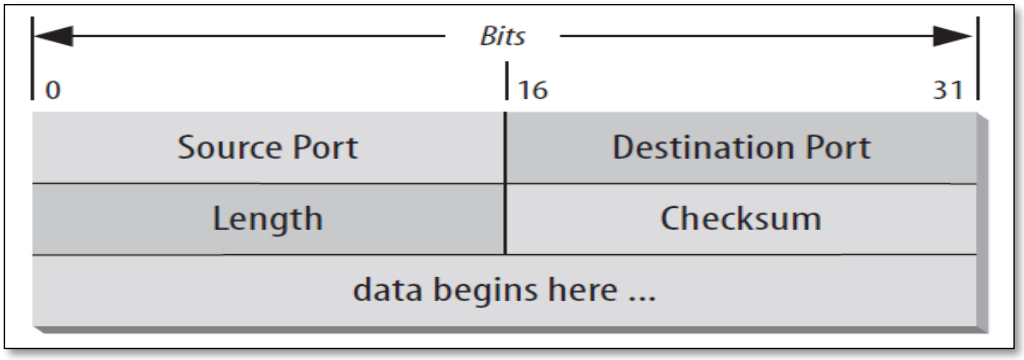


Figure (2.4b): UDP Header

Figure (2.4): TCP and UDP headers in IPv4

2.3 Network Intrusion Detection System

Network attacks are becoming more common and severe as network technologies and applications develop. Intrusion is a key to compromising the CIA of a computer resource[35]. Intrusion detection is the network traffic monitoring and computer event technique to detect harmful or unauthorized activities. All network intrusion detection software or devices are considered network intrusion detection systems (NIDS) [36]. The use of NIDS expertise to convert recorded activities to alert. These alerts are reported to a manager (administrator) or collected via the SIEM

system centrally (Security Information and Event Management). A SIEM system analyzes the outputs of several sources in real-time to correlate the different alarms and provide a comprehensive view of IT security [37].

2.3.1 IDS Technology

The Intrusion Detection Technology is presented in a taxonomical manner here. Several varieties of IDSs available today, each with its own monitoring and analysis techniques. Each technique has its own set of benefits and drawbacks. These techniques can be expressed using a generic IDS process model [38]. Many IDSs can be divided into the following components:

2.3.1.1 IDSs Information Sources and Location

The initial step in classifying IDSs is to locate the information source. IDS monitors and evaluates packets for suspicious activity. May can configure smaller systems to monitor traffic specific to a particular server, switch or router or place a large IDS server on the network to monitor traffic**.** Another version of IDS can be run from a centralized server, which will scan system files for illegal behavior and maintain data integrity. Based on the type of work being analyzed, IDSs can be divided into two major groups:

- A Host-based intrusion detection system (IDS) is a software agent that monitors and analyzes certain host activities such as files, processes, and system logs. Host-based IDS can use a various tools, which look for unauthorized changes within the individual hosts with great accuracy and precision [39].
- A network IDS (NIDS) is a Network-based IDSs capture and analyze network flows to detect anomalies by listening and inspecting the packets information from network segments such as

a switch. As NIDS are more scalable and cross-platform than HIDSs, they are used to defend the IT infrastructure of a company. Because it catches packets flowing over communication channels, NIDS is sometimes known as "intrusion packet-sniffers". The sensor and the management station are the two logical components of a NIDS. The sensor sits on a network section, keeping an eye on it for suspicious activity. The management station receives and displays alarms from the sensor(s) to an operator [40].

2.3.1.2 Detection and Analysis Methods

Detection methods are considered the core engine of the IDSs in detecting attack, which alarms the administrator system or security analysts. There are many types of detection methods regarding the monitoring data. The main two categories are *Signature-based and Anomaly-based methods*.

- Signature-based (sometimes known as 'misuse detection') detection consists of a database of known attack signatures. It compares the monitored data with the database for the signatures. IDS scans the input stream for an attack pattern, like a traditional antivirus. This signature may be a byte-string, but complex patterns are often expressed as a tree-branch diagram. The database of this type must be updated regularly to be effective. The main disadvantage of this approach is that it cannot detect unknown (new) attacks [41].
- Anomaly detection tries to learn the behavior of the "normal" or "expected" system. This approach does not require updates to the database or even one. It can effectively detect the unknown and new attacks that are not previously known, like zero-day attacks, but it also produces many false positives that are difficult to solve. It's

also harder to collect information on the attack because it has no clear signature [38].

In this dissertation, an anomaly detector has been designed and implemented to construct a long term profile and then generate information that can analyze as identifiers to detect anomalies. BAT swarm optimization and data mining clustering with artificial neural network algorithms are used in this dissertation as anomaly detectors.

2.3.1.3 IDS Behavior of Response

IDS Response is a set of actions that the detector system takes when the malicious packets are presented. Active and passive responses are the most two common types of the IDS behavior [42].

Active **R**esponses: The IDS takes immediate and automatic active responses to prevent an attack. The detector in this type takes many steps, such as suspending the attack by blocking the IP address, port numbers and terminating the connection [38].

Passive **R**esponses: The system administrator takes passive responses to suspend the attack. The detector in this type raises alarms and notifications to notify the system administrator or security analysts. Figure (2.5) illustrates the response action to the system administrator when the attack occurs [6].

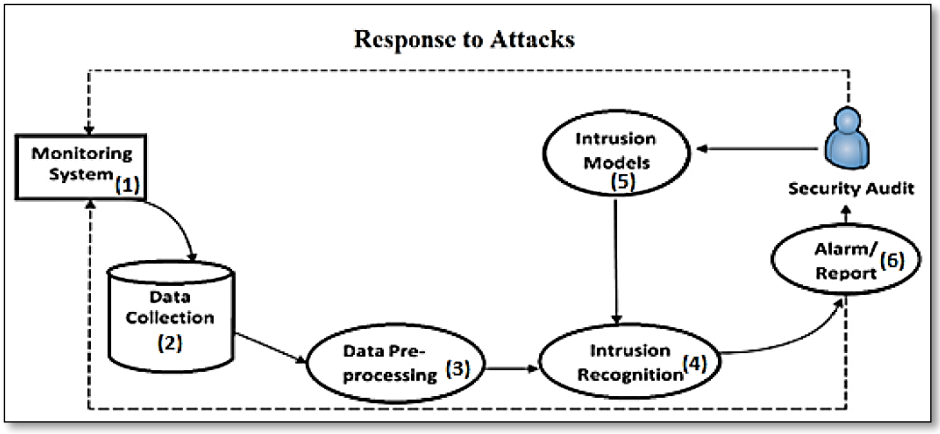


Figure (2.5): Intrusion Detection System with Response to Attack

2.3.1.4 IDS Time aspects

IDS time aspects refer to the elapsed time between the monitored events and the detector analysis of these events. It is primarily classified into two types: post-time analysis and real-time analysis.

In the first type, the detector process does not analyze network flows. It is likely the “store and forwards” fashion in communications systems. The main disadvantage of this type is that it takes time to respond to the attacks.

In a real-time analysis, the detector process continues analyzing the network flows. The detector in this type monitors the traffic and analyzes it in real-time. In addition, it takes an active response to prevent the forwarding of attack by inspecting the packet header information and contents. It takes less time to avoid the malicious. It is inefficient in a high speed of data and traffic volume [6].

2.4 Network Attacks

In a computer network, Intrusion attacks are defined as attacks, that enter the system to read, destroy, expose, alter, disable, and steal your data. Security attacks may be divided into two main classes: active and passive attacks.

- **Passive Attack** The attacker in passive attacks is only trying to study or make use of information from the system without affecting the system's resources. Figure (2.6) shows this type of attacks [12][43].

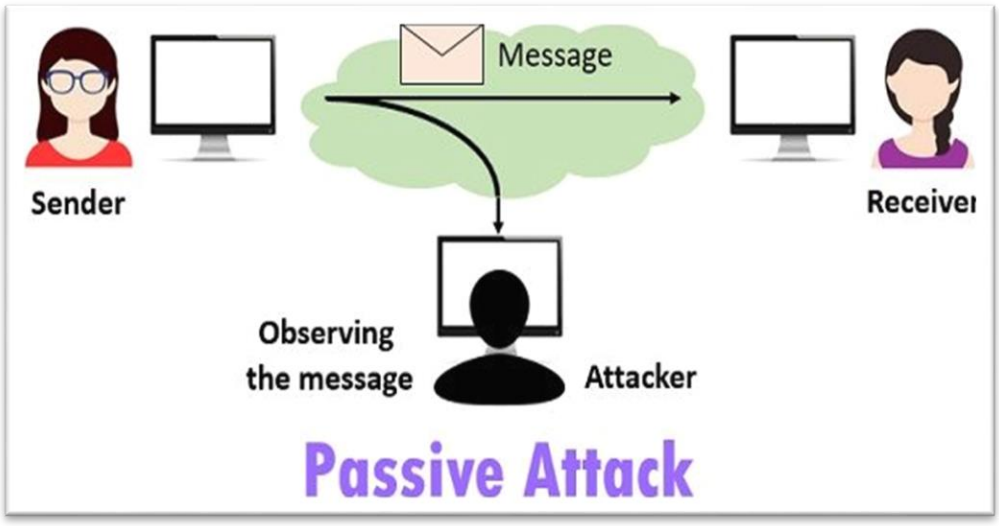


Figure (2.6): Passive Attack

- **Active Attack** The attacker in this type of attack attempts to modify the system's resources or affect its operation. This type of attack is one where the opponent tries to remove, add, or in some other way modify the communication on the channel. Figure (2.7) presents this type of attacks [12][44].

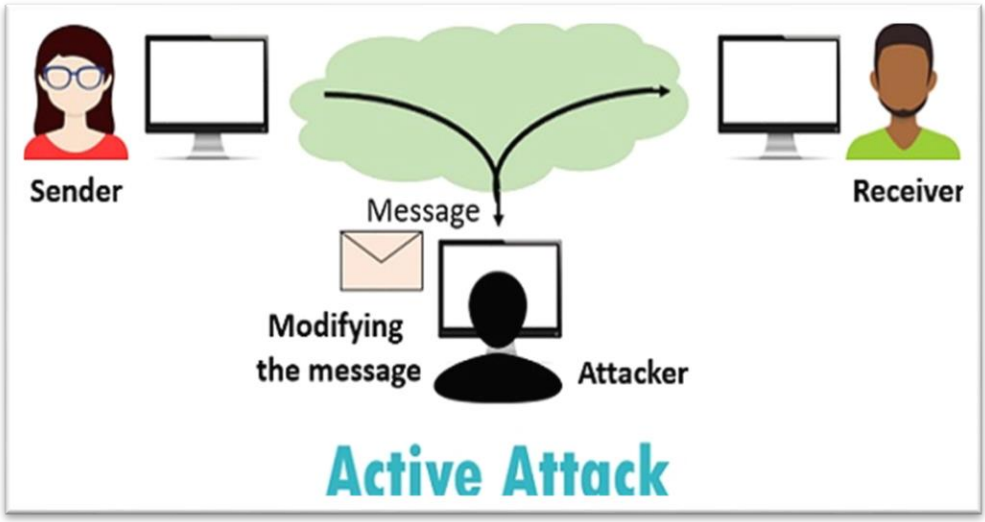


Figure (2.7): Active Attack

2.4.1 Flooding Attacks

It is also called a brute force attack. Multiple packets are sent in this attack to block the computing resources on the target victim, preventing it from serving its legitimate users. This attack consumes many resources,

including network bandwidth, storage space, CPU time, data structures, and network connections. The following subsection explains flooding attacks types [45].

2.4.1.1 TCP SYN Flooding Attack

This attack uses a weakness in the TCP connection setup's three-way handshake. When a server receives an initial SYN (synchronize/start) request from a client, it responds with an SYN/ACK (synchronize/acknowledge) packet and then waits for the client to send the final ACK (Acknowledge). An attacker launches an SYN flooding attack by sending massive SYN packets but never acknowledging any of the responses, leaving the server waiting for non-existent ACKs. Because the server has only a limited buffer queue for new connections, it will not process more incoming connections as the queue becomes overburdened [46]. TCP SYN flood attack is shown in Figure (2.8).

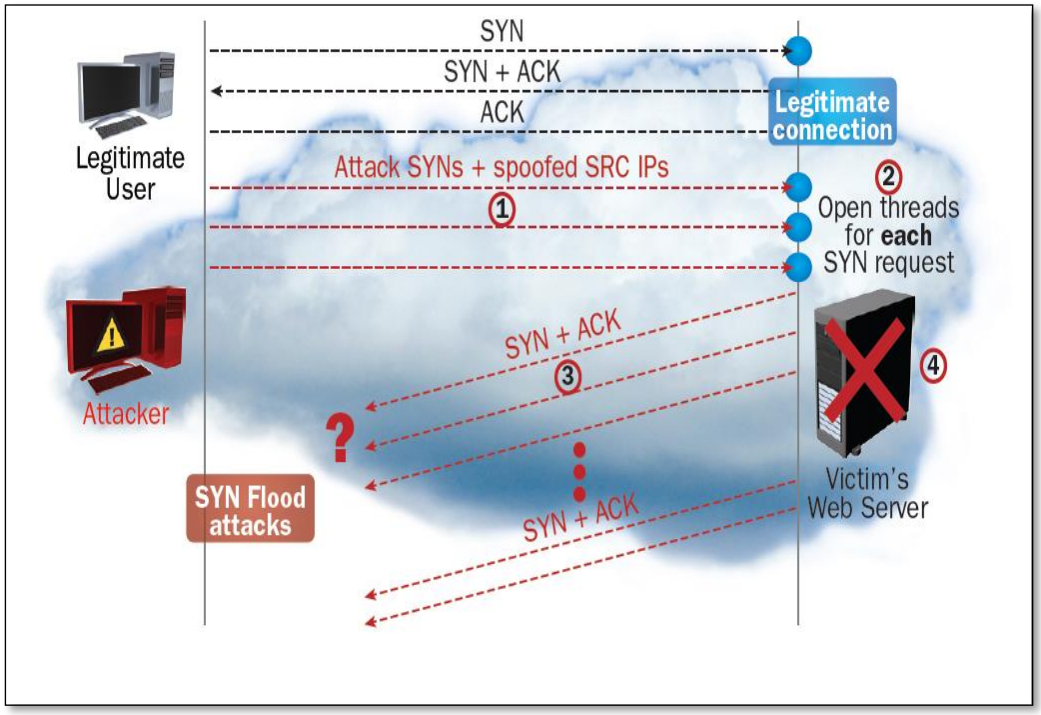


Figure (2.8): TCP SYN Flooding Attack [12]**.**

2.4.1.2 UDP Flooding Attack

In this attack, a massive number of UDP packets are sent to the victim system at randomly or specified ports[47]. When a UDP packet is received by the victim system, it attempts to determine which applications have requested data. Suppose the victim system is not running any applications on the destination port, in that case, it will generate an Internet Control Message Protocol (ICMP) packet to the sending system indicating a "destination port unreachable" message. If the victim ports received enough numbers of UDP packets, the system would shut down, as shown in Figure (2.9) [48].

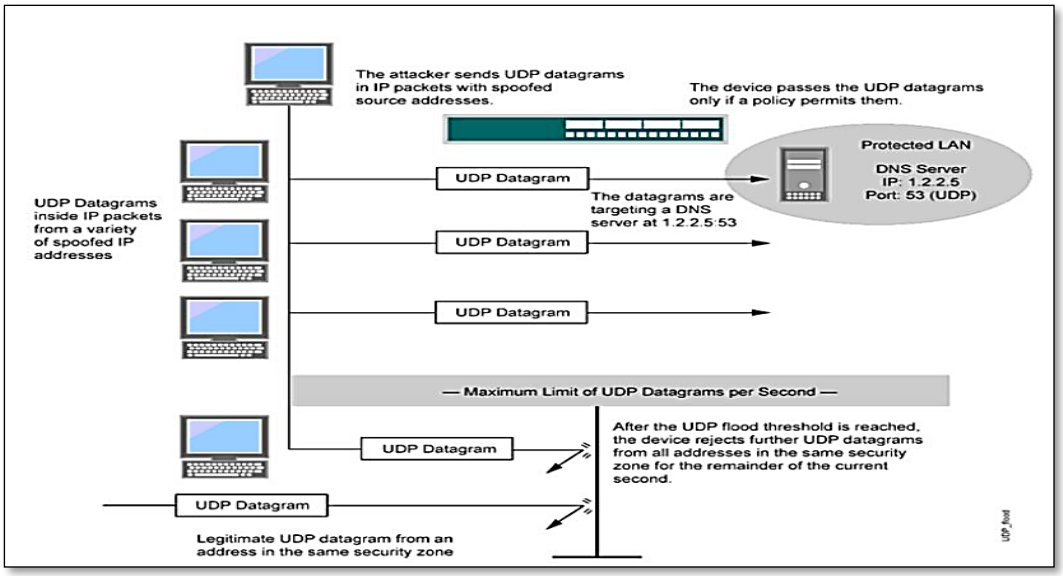


Figure (2.9): UDP Flooding Attack [48].

2.4.1.3 ICMP Flooding Attack

It is also known as (Ping Flood). This attack happens when a large number of ICMP_ECHO_REQUEST packets are sent to the target system [48]. These packets request a response from the victim system, and the resulting traffic overloads the target connection's bandwidth. The source IP address of the ICMP packet can be faked used by an attacker. Figure (2.10) illustrates ICMP flooding attacks [6].

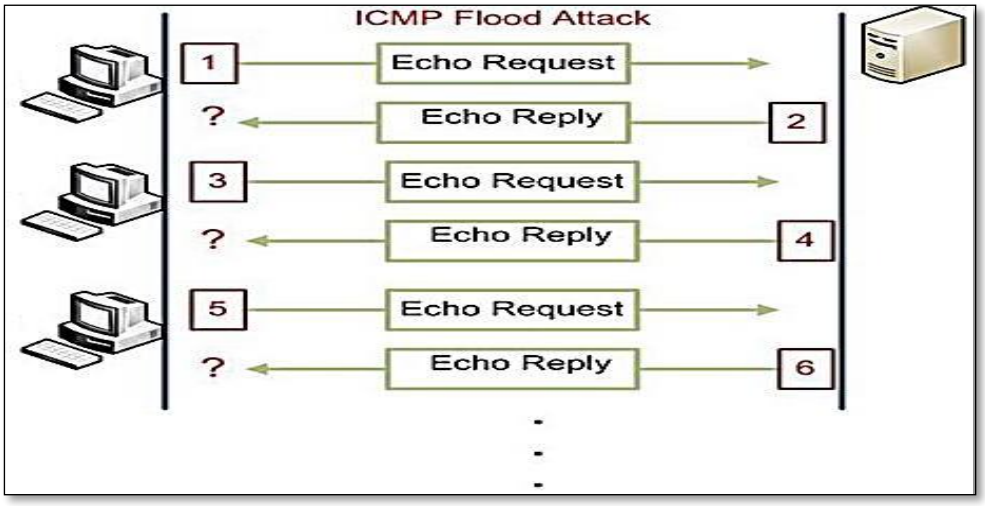


Figure (2.10): ICMP Flooding Attack.

2.4.2 Common Attacks Detected

Protocol attacks, such as TCP SYN attacks, UDP attacks, and ICMP attacks, use a specific feature or implementation weakness in a protocol installed on the victim to consume excessive quantities of computational resources. It floods the server with fake authentication requests, damaging its resources [48].

A port scan is a common technique hackers use to discover open doors or weak points in a network. A port scan attack helps cyber criminals find open ports and figure out whether they are receiving or sending data. It can also reveal whether active security devices like firewalls are being used by an organization [49].

Multiple types of these attacks are detected in the proposed system using two datasets (NSL_KDD, UNSW-NB15) from the active attack type. These are explained in the following subsections:

- **User to Root Attacks (U2R):** An attacker prepares this type of attack to acquire unauthorized access to a victim system across the entire network [50].

- **Remote to local Attacks (r2l):** This attack works to send a packet to connect the victim machine via the network and takes advantage of bugs or weaknesses in the system to take illegal local access to resources on that existing network [51].
- **Probe Attacks:** The attacker scans the network to search for vulnerabilities and gather any information regarding the victim machine to access the victim's resources that exist on the network [50].
- **Denial of Service Attacks (DOS):** The most common DoS attack is the TCP SYN flooding attack. The attacker tries to prevent the user of a specific service or resource. This kind of cyber-attack aims at making a computer or network resource unavailable to its intended users by temporarily or permanently disrupting the services of a host connected to the network [50].
- **Worm Attacks:** Unlike viruses, this malware may clone itself across the network without infecting host data. It has the potential to cause a DDoS attack by repeatedly reproducing itself and overloading the servers. Many worms are designed to spread and have no intention of altering the systems they pass through [52].
- **Fuzzers Attacks:** Fuzzing is a technique for stressing an application and causing unexpected behavior, resource leaks, and crashes by feeding it the randomly generated data. This attack aims to give abnormal data to a system, causing it to crash and reveal reliability issues [12].
- **Backdoors Attacks:** is a kind of malware beyond standard authentication protocols for system access. Distant access to resources, including databases and file servers, within an

application, is therefore allowed so that criminals can send system commands to update malware from a distance [52].

- **Exploits Attacks:** The attacker is aware of a security flaw in an operating system or piece of software and can affect that information by exploiting the vulnerability. It is referred to as "the source of evil" [12].
- **Generic Attacks:** This attack can be used against any block cipher (with a given block and key size), regardless of the structure of the block cipher [53].
- **Reconnaissance Attacks:** Unauthorized data collection of system resources, vulnerabilities, or services is known as reconnaissance. It is also known as information gathering. They allow attackers to know which connections are accessible to the system and supply them with enough information to exploit in denial of service (DOS) attacks [49].
- **Shellcode Attacks:** It is a specific type of code injected remotely and used by hackers to exploit a range of software flaws (vulnerabilities). It gets its name because it usually launches a command shell from which attackers can take control of the system [52].
- **Analysis Attacks:** Based on what the attacker observes over the network, traffic analysis attacks are launched. The attacker can scan network data for significant node locations, routing topologies, and even application activity patterns [52].

2.5 Benchmarks Datasets

A large amount of data is required to train any machine-learning algorithm. The quality and quantity of the dataset are critical. Two datasets are very popular in the network intrusion detection field:

NSL_KDD and UNSW_NB15; we have chosen a variety of realistic attacks.

2.5.1 NSL_KDD Data Set

The NSL_KDD dataset contains solutions to most of these issues that show in the KDD_Cup99 dataset**.** The number of connections in train and test sets is also reduced: from 805050 in KDD_Cup 99 to 148517 in NSL-KDD. Connections are also categorized into groups based on their difficulty degree. This category can aid classifiers in recognizing the most difficult-to-detect attacks during their training phase [54] [55]. In the case of distribution, the classes in the training and testing sets of the NSL_KDD are imbalanced, as shown in Figure (2.11) [56].

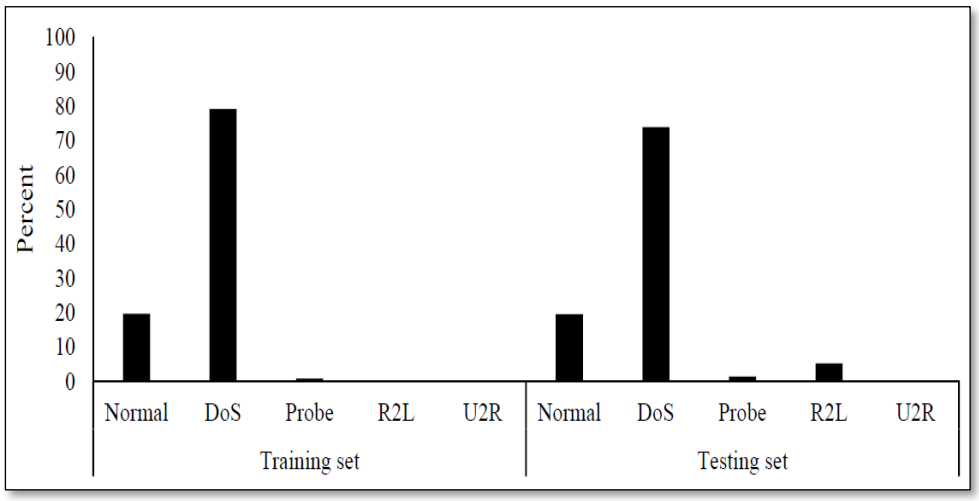


Figure (2.11): The Percentage of Class Distribution in the NSL_KDD Data Set

2.5.2 UNSW_NB15 Data Set

The UNSW-NB15 was created using the IXIA Perfect Storm tool in the Australian Centre for Cyber Security's (ACCS) Cyber Range Lab and released in 2015. It solved the problems of older benchmarks data sets like KDD_CUP1999. The UNSW_NB2015 dataset contains a combination of real-world and synthetic attack activities of the network traffic. In comparison to the NSL_KDD dataset, the UNSW_NB15 has more forms

of attacks. This dataset contains 2,540,044 records with 49 characteristics, including packet-based and flow-based features, that provide realistic and modern normal and abnormal network activity. UNSW_ NB15's complexity originates from its structure, making it more comprehensive in evaluating of existing network intrusion detection systems [22][57]. In the case of distribution, the classes in the training and testing sets of the UNSW_ NB15's are imbalanced, as shown in Figure (2.12) [56].

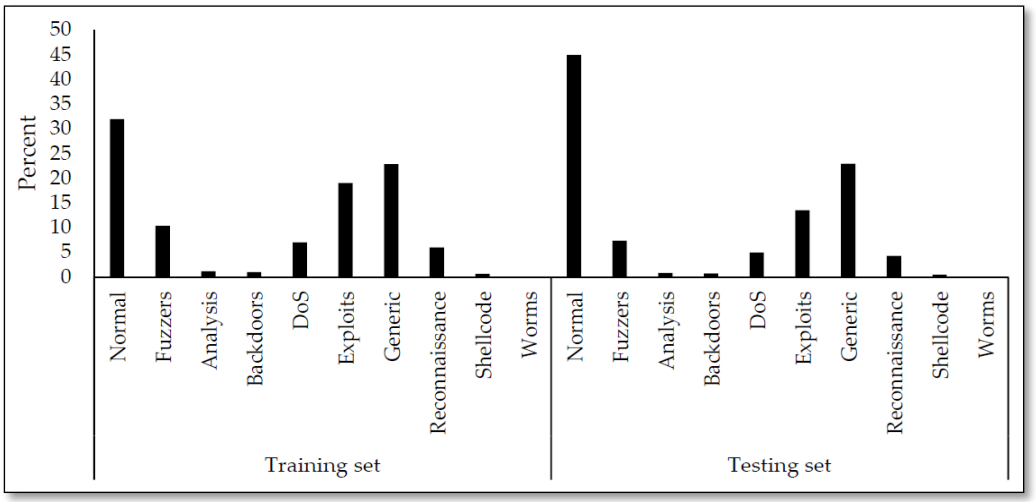


Figure (2.12): The Percentage Of Class Distribution In The UNSW_ NB15 Data Set

2.6 Network Flow based on Knowledge Discovery Techniques

The Computer networks' increasing processing and storage capabilities make it possible to record and store vast amounts of data at a low cost. Even though more data may contain more information, understanding a huge volume of collected data and extracting new and interesting knowledge is often difficult. The phrase "data mining" refers to methods and algorithms for analyzing data to find patterns that define data attributes [58]. Data mining is simply one stage of the knowledge discovery process in databases, but it is the most significant (KDD).

It's worth noting that KDD is characterized as a critical method for identifying accurate, novel, possibly helpful, and ultimately intelligible patterns in massive datasets and extracting relevant data from databases. This process consists of several phases and incorporates database techniques, machine learning (ML), artificial intelligence, and decision-making systems [59]. While both ML and DM use the same techniques, their processes and applications differ. The definitions of ML and DM are so similar that they have a lot of similarities[60]. Knowledge discovery in databases (KDD) has gotten a lot of press in the IT business and society. The KDD methodology is an iterative and reactive process that combines the subject's experience with a various analysis methods, such as DM and ML algorithms. Figure (2.13) shows the five steps of KDD [59].

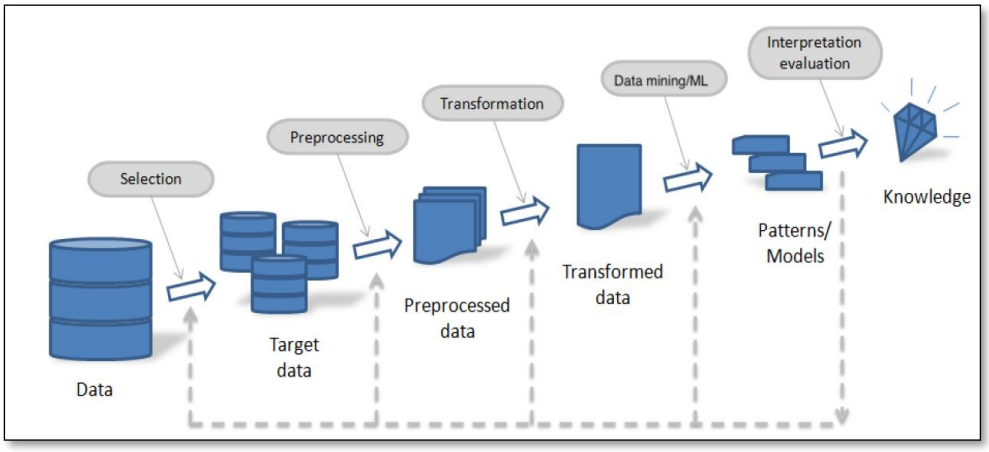


Figure (2.13): KDD Flow for IDS, Showing the Results at Each Step and Its Iterative Nature.

As a result, DM approaches that are viewed as complementary to ML techniques in terms of relevance and creativity can be used with NIDS to detect new, imperviously seen intrusions accurately and automatically. Data mining based on NIDS may efficiently identify the user's critical data and predict the results, which can be used to detect many forms of attacks [61].

2.7 Network Flow

This model has been applied to a large amount of network traffic. These data are collected from different network flows and then saved in a database to retrieve the desired features. In network-based detection against malicious attacks, these databases include various types of information such as protocol type, Network service, and flag.

2.7.1 Data cleaning and preprocessing

The first step of preprocessing phase involves filling the missing values. The most common category that might use to fill in the blanks in this scenario is that if there are a lot of missing values, they can be replaced with a new category [14] [15]. The preprocessing step is necessary for network traffic either manually or programmatically. Missing data, noise and outliers consider data disturbances and need to be preprocessed [62].

2.7.2 Data Transformation

A significant step in network traffic is data transformation. This step includes many tasks such as smoothing to remove noise from data, aggregation, summarization and normalization. in this subsection propose two methods are One-hot encoding and min-max normalization as flow:

2.7.2.1 One-hot encoding

A structured dataset often includes numerous columns of numerical or category data. Machine Learning algorithms are only capable of recognizing numbers, not text. As a result, before using the textual/categorical input to train a model, we must transform it to numbers. This difficulty can be solved using the One-Hot Encoding approach [63].

2.7.2.2 Min-Max Normalization

A linear transformation is performed on the original data by the min-max normalization. It is one of the most widely used scaling methods, and the basic principle 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 and maximum value [64][65]. The following equation (2.1) shows the computation:

$$\boldsymbol{Min\_Max(xi') = \frac{xi - min(xi)}{max(xi) - min(xi)}} \quad \textbf{… (2.1)}$$

where $Min\_Max(xi')$ represents the shrinkage of the original attribute values to the interval (0 to 1) when the values are positive or to the interval (-1 to 1) for negative data; xi is the entire set of values in a column; $min(xi)$ the minimum value in a column, and $max(xi)$ is the maximum value in a column.

2.7.3 Feature Selection

It is crucial to select the necessary attributes from a large amount of data. Typically, the network traffic includes more features that are irrelevant to the data model[66]. The primary goal of feature selection is to identify the underlying features among the data. It work for dataset dimensionality reduction to improve the performance of data mining techniques[67].

There are two types of feature selection: a wrapper model and a filter model. A filter model uses general characteristics of the training dataset without any learning algorithm, and it works fast to select the feature[68]. In contrast, the wrapper model uses a predictor algorithm such as a decision tree, and it performs to choose the most relevant features. However, the wrapper model can usually produce the best subsets of features for a specific learning algorithm and consequently improve

performance compared to filter methods [69]. For example, the modern Wrapper model, such as SI Algorithms, is frequently inspired by nature, and the resulting algorithm can be of many different variations. Nonetheless, for calculating the major modernized formula, all of the algorithms are used with some of the fundamental qualities.

The Wrapper model is one of the robust feature selection (FS) techniques to select the optimal feature's based on testing different groups of subsets. Metaheuristic algorithms are used to optimize the subset features that feed into the wrapper model. These models has been used the randomness to find an optimal solution. The figure (2.14) shows the steps of Wrapper feature selection [70].

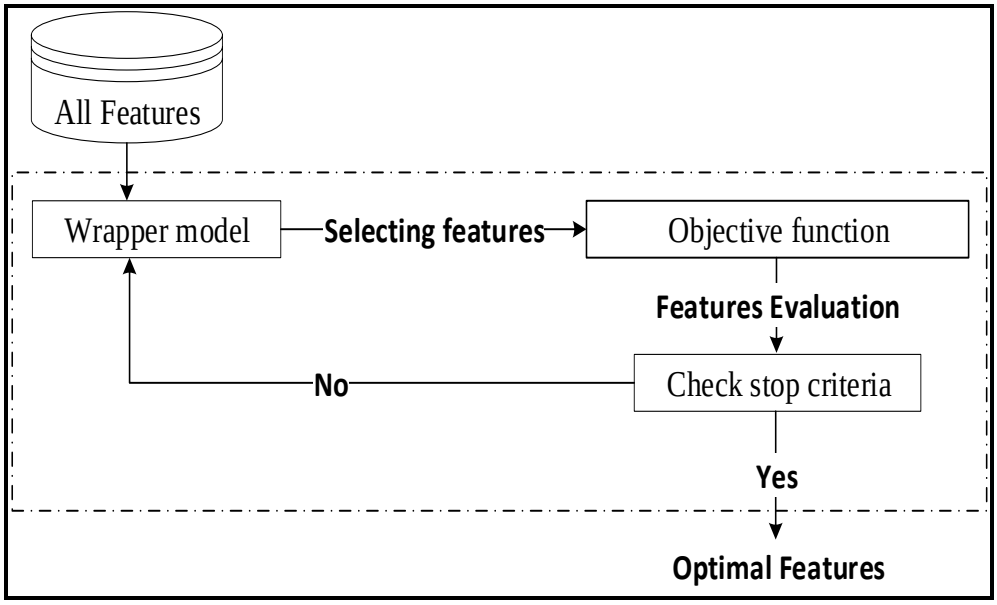


Fig (2.14): Wrapper Feature Selection Model

- **Bat Algorithm**

Xin-She Yang developed the Bat Algorithm in 2010, a SI algorithm inspired by the nature of bats [71]. The essential properties of this algorithm are established with the help of micro-bats. The bat will detect the object using reflected sound from the echolocation process[70]. The

algorithm generally begins by adding bat velocity (v_i) to the current position x_i, resulting in a new candidate solution (x_i'). This process must be dynamically updated after each iteration. The bat velocity was modified based on the best current solution (x_*). However, this will support exploration search by adding frequency to a new bat velocity (v_i'). The following equation(2.2) can be used to calculate modifying the frequency of the bat [72][73]:

$$\boldsymbol{f_i = f_{min} + (f_{max} - f_{min}) * \beta} \quad \textbf{...(2.2)}$$

Where: f_{max} and f_{min} are maximum and minimum bat frequencies should be manually specified by the administrator, β is a random vector that belongs in this range [0, 1].

$$\boldsymbol{v_i' = v_i + (x_i - x_*) f_i} \quad \textbf{...(2.3)}$$

$$\boldsymbol{x_i' = x_i + v_i'} \quad \textbf{...(2.4)}$$

The exploration in the search process of this algorithm based on bat loudness (A_i) and emission pulse rate (r_i). The equation (2.5) calculate A_i and equation (2.6) calculates the r_i such as blow:

$$\boldsymbol{A_i' = \alpha A_i} \quad \textbf{...(2.5)}$$

Where: A_i' is a new value for bat loudness, and $\boldsymbol{\alpha}$ constant value[74].

$$\boldsymbol{r_i' = r_i[1 - e^{-\gamma}]} \quad \textbf{...(2.6)}$$

Where: r_i' is a new emission pulse rate value and γ is constant value. The bat will search in local space use regular random move to make an efficient position in this local space as:

$$\boldsymbol{Xnext = Xold + c * A_*} \quad \textbf{...(2.7)}$$

Where $\boldsymbol{c}$ is a random number and A_*^t is the average loudness of all bats at iteration t.

2.7.4 Data Mining and Machine Learning

In this step, data mining and machine learning algorithms are applied. This step enables the researcher to select suitable techniques such as clustering, classification, prediction, and time series analysis [59].

2.7.5 Model Evaluation

Many performance criteria can be used for model evaluation. In terms of attack detection, performance measures like accuracy, precision, recall, F1 score and Silhouette Index measure are used to evaluate the proposed system. It can be applied to training and testing phases [75][76].

2.8 Data Mining Techniques

Many of the problems can be formulated using the following data mining tasks, which will be illustrated in the following subsections:

2.8.1 Association

It is the process of establishing the relationship between items, which exist together in data records. For example, in the market basket, it determines which items are frequently purchased together within the same transactions, which helps to design attractive packages or groupings of products and services [77].

2.8.2 Classification

It is a data mining technique that aims to form a model that can be used to classify unlabeled data. For example, build a model for predicting future client behaviors by categorizing the records of the database into some predefined classes based on certain standards [77].

- **K Nearest Neighbor (KNN):** is a non-parametric technique used in supervised machine learning to classify new object

based on the high-density class of the nearest available cases. Generally, the distance between objects is calculated by one of Minkowski Distances (Manhattan, Euclidean Distance, and Distance)[70].

- **Support Vector Machine (SVM):** is a discriminative supervised machine learning introduced by as both regression and classifier model. Technically, it classifies a new object based on Hyperplane and Support Vectors. The hyperplane is multiple lines detected boundaries of classes that help to determine class data objects easily. SVM model sets the diminution of hyperplane based on present features in the dataset.
- **Decision Tree (DT):** is a non-parametric supervised machine learning model that predicts the class of query data based on a sequences series of decision rules[78]. The root of each decision rule is a feature of data has the highest information gained than others.
- **Naïve Bayes (NB):** is a probabilistic supervised machine learning model predicts the category of query data based on core concepts Bayes' theorem. Technically, it calculates the probability $P(y/x)$ of query data x with all training classes y. The high $P(y/x)$ is determined by the new class of x [79].
- **Random Forest (RF):** is a supervised learning algorithm which is used for both classifications as well as regression. However, it is mainly used for classification problems. Technically, it builds own model based on four major steps: select random sample from data, construct a decision tree for every sample, voting will be performed for every predicted

result, at last select the most voted prediction result as the final prediction result.

2.8.3 Regression

Regression is also named estimation, it is a type of numerical estimation method, which is used to replace each data object with the actual value represented by the prediction value. It deals with continuous value results and arises with some unknown variable. Regression's uses include prediction, modelling of underlying relationships, and testing theories about relations between variables [77].

2.8.4 Clustering

It is a technique for extracting new knowledge through unsupervised data mining. The technique of arranging a set of items into clusters based on a distance measure is known as clustering. Clustering produces a collection of clusters, each of which has a group of points, which are close to one another but far apart from other clusters. The distance measure *d(X, Y)* takes two variables in the space and returns a numeric distance between these two arguments [77].

2.9 Cluster Analysis

Cluster analysis is an unsupervised data mining methodology that groups object into clusters similar to each other but different from data points in other clusters in order to extract new knowledge from a massive amount of data. In addition, as shown in figure (2.15), clustering methods are divided into five groups [77].

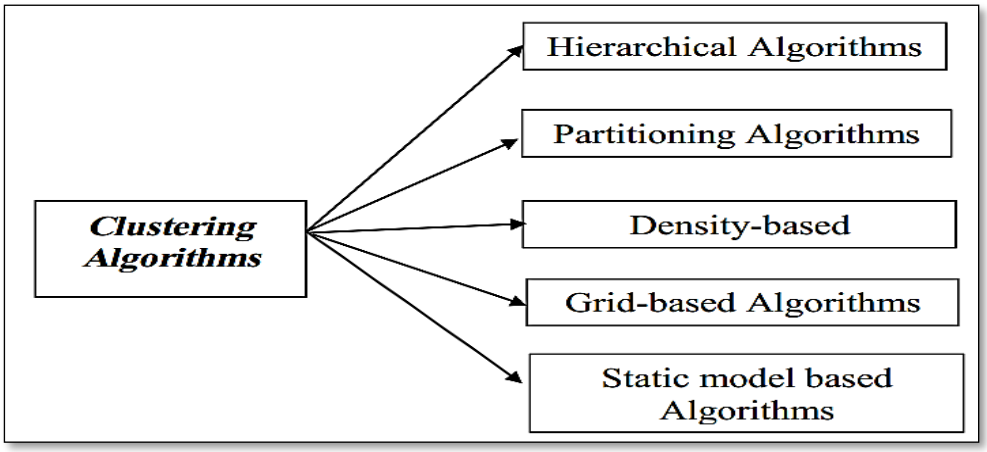


Figure (2.15): Common Types of Data Mining Clustering Techniques [77].

- **Hierarchical Algorithms**: Each data point is considered as a separate cluster in the hierarchical cluster analysis, which then merges them using one of several different definitions of "near," such as Euclidean Distance. Hierarchical approaches are divided into two categories: agglomerative and divisive. In agglomerative approaches, it start with single-data-point clusters and merge sub-clusters until only one cluster remains. Divisive approaches divide superclusters in the opposite direction until all points are in their clusters [80]. A, B, C, D, and E are five labelled points shown in Figure (2.16) as an example of hierarchical clustering [81].

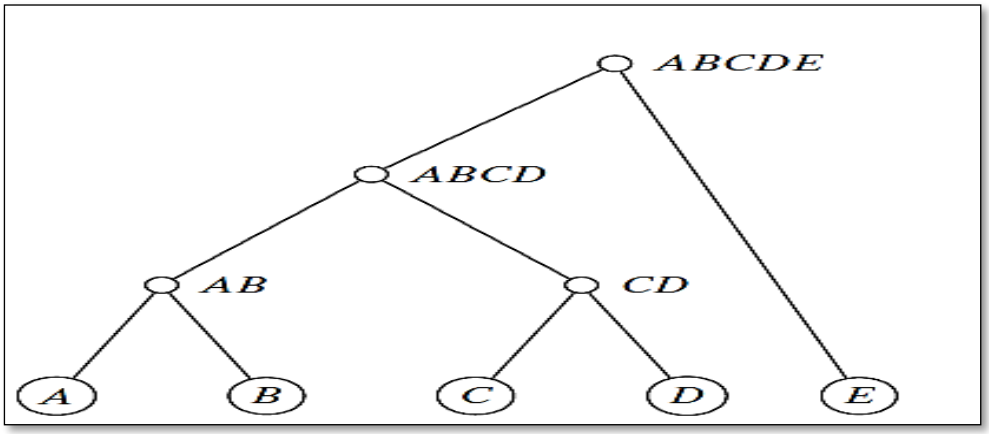


Figure (2.16): Shows An Example Of Hierarchical Clustering Of Five Labeled Points: A, B, C, D, And E.

- **Partitioning Algorithms**: Partitioning, also known as relocation methods, is the most fundamental and basic form of cluster analysis. It divides a set of objects into multiple distinct groups or clusters. It's also referred to as (Centroid-based clustering). The objects are divided into k partitions via the partitioning algorithm, with each partition representing a cluster [77]. Figure (2.17) show Partitioning Clusters [82].

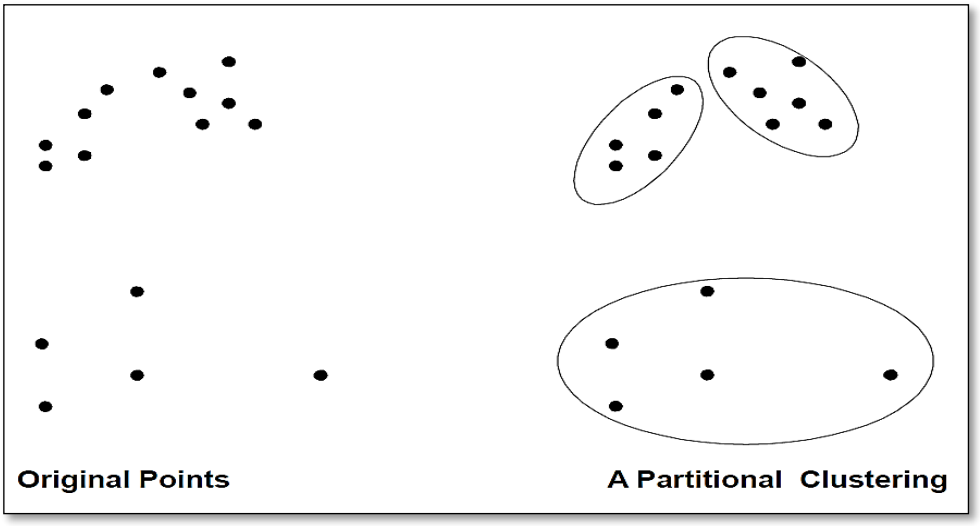


Figure (2.17): Partitioning Clusters

- **Grid-based Clustering**: The grid-based clustering approach considers cells rather than data points. Grid-based methods reduce the processing time by separating the object space into a finite number of grid cells. Grid-based clustering involves quantizing the data space into a limited number of cells that form the grid structure and performing clustering on the grids [83]. Figure (2.18) show Grid-based Clusters.

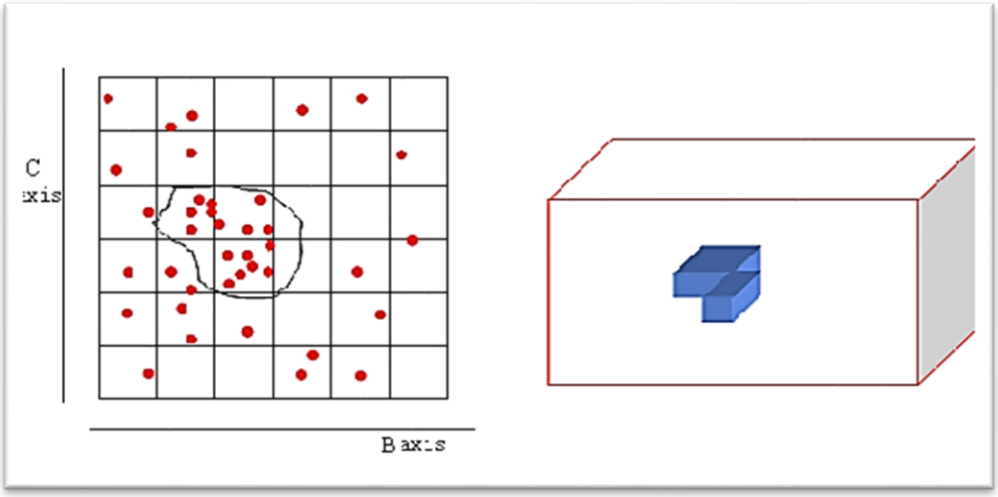


Figure (2.18): Grid-based Clusters

Density-based Clustering: in the case of locating spherical-shaped clusters, partitioning and hierarchical approaches are used. They have trouble finding clusters of any shape, such as the "S" and oval clusters shown in Figure (2.19). However, it can characterize clusters as dense regions in the data space separated by sparse regions to locate clusters of any shape[84]. Density-based clustering algorithms can discover clusters with non-spherical shapes use this strategy as their primary strategy. The cluster increases incrementally with this clustering technique as long as the density (number of data points) in the "neighborhood" passes a certain threshold [77].

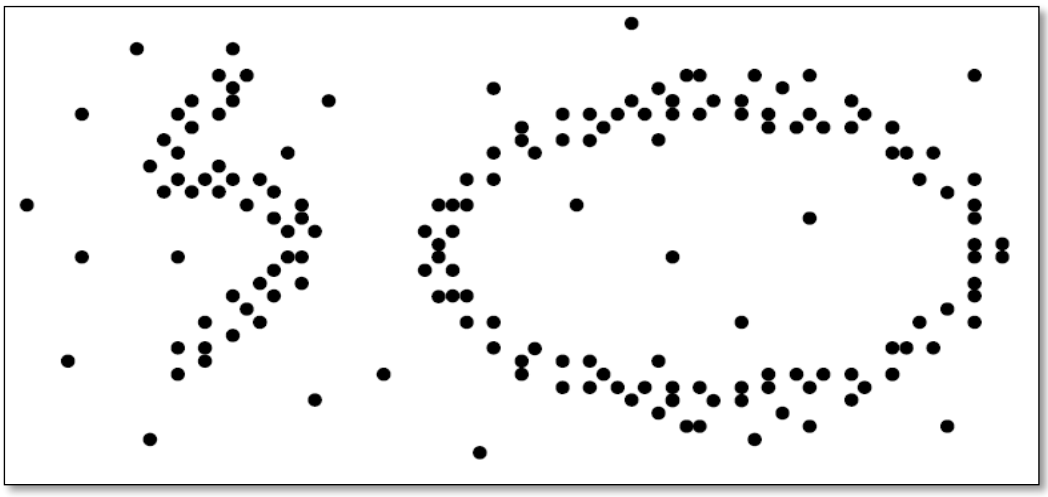


Figure (2.19): Clusters of Arbitrary Shape Based Density Cluster

2.10 Distance and Similarity Measure

The distance between two items is calculated by a function called dis(x, y). It returns a collection of real numbers in coordinate space. Four criteria must be met when measuring distance[85]:-

A. $\mathrm{dis}(x, y) \geq 0$.

B. $\mathrm{dis}(x, y)$=0 if and only if x = y.

C. It has a symmetric property such as $\mathrm{dis}(x, y) = \mathrm{dis}(y, x)$

D. It has a short path using the triangle inequality $\mathrm{dis}(x, y) \leq \mathrm{dis}(x, z) + \mathrm{dis}(y, z)$

- **Euclidean distance measure:** The most common distance is Euclidean distance measure, which is defined in equation (2.8)

$$\boldsymbol{dis}\ (\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{xi} - \boldsymbol{yi})^{\boldsymbol{2}}} \qquad \textbf{...(2.8)}$$

  Where $\boldsymbol{x}$i and $\boldsymbol{y}$i are two real vectors with (n) dimensions, this distance is a normal distance, which sums the square in each dimension under the positive square root. It is used in most data mining clustering algorithms to calculate the distance between each pair of data points when the data lies on a low dimensional manifold [85]. Because Euclidian distance does not account for the quality of clustering, it is ineffective for maximizing homogeneity inside each cluster and heterogeneity across clusters [86].

- **Gaussian kernel distance measure:** Kernel methods are frequently employed in the analysis of massive, high-dimensional datasets. These methods use an affinity kernel to express similarities, distances, or correlations between data points, which is a non-parametric approach to data representation [87]. The Gaussian kernel function is widely employed for cluster analysis as

a similarity measure [88]. The formula of Gaussian kernel function is defined in equation (2.9) [89]:

$$\boldsymbol{dis}\ (\boldsymbol{x},\boldsymbol{y}) = \mathbf{exp}(-\frac{||\vec{x}-\vec{y}||^{2}}{2(\sigma^{2})})\ \ \boldsymbol{where}\ \boldsymbol{\sigma} > \mathbf{0} \qquad \textbf{...(2.9)}$$

Where $\boldsymbol{\sigma} = scaling\ parameter$, and $\| xi - xj \|$ is the Euclidean metric when($xi \neq xj$), where (xi and xj) are points in the dataset.

- **Fractal measure:** Natural groups of data have a high degree of self-similarity, which can be measured by calculating the fractal similarity of the original dataset and the fractal similarity of each of the instances. It is possible to group self-similar elements of data that are likely to belong to the same clusters by calculating the fractal similarity of the whole data and the fractal similarity of each of the instances. Self-similarity, which suggests a scaling connection when little bits of an objects are exact smaller copies of the whole objects, has become one of the characteristics of fractal [90]. The fractal factor is calculated using Equation (2.10) below [91]:

$$\boldsymbol{f_i} = \frac{(\sum_{i=1}^{n}(\boldsymbol{x}-\boldsymbol{R'})^{2})^{2}}{\sum_{i=1}^{n}(\boldsymbol{x}-\boldsymbol{R'})^{4}} \qquad \textbf{...(2.10)}$$

Where x a point in the dataset, R' is the mean of x or any summarization metrics. The R' is calculated by equation (2.11)

$$\boldsymbol{R'} = \frac{1}{N}\sum_{j=1}^{N}\boldsymbol{x_j} \qquad \textbf{...(2.11)}$$

Where n is dimensional of pointx_j. Equation (2.12) calculate the ΔF between f_{x_j} and each f_{xi} in cluster:

$$\Delta\boldsymbol{F_j} = (|\boldsymbol{f_{x_j}} - \boldsymbol{f_{x[i]}}|)/\ \boldsymbol{f_{x_j}} \qquad \textbf{...(2.12)}$$

2.11 Density Peak Clustering (DPC) algorithm

The Density Peak Clustering (DPC) algorithm is one of the Density-based methods. This algorithm discover clusters based on a high local density of data points [92]. The DPC deal with continuous regions. The original Density Peak Clustering (DPC) algorithm start by defining the distance between two data objects to calculate the density and separation distances from other data objects with higher density [93]. Let X = {x1, x2 . . . xn} be a dataset with n data points. Each xi has M attributes. Therefore, xij is the *jth* attribute of data point xi.The Euclidean distance between the data point's xi and xj can be expressed as follows [94]:

$$\boldsymbol{dis}\ (\boldsymbol{xi}, \boldsymbol{xj}) = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{xi} - \boldsymbol{xj})^{\boldsymbol{2}}} \qquad \textbf{…(2.13)}$$

Equations (2.14) express the calculation of local density for each point in the given dataset.

$$\boldsymbol{Pi} = \sum_{\boldsymbol{i,j} \in \boldsymbol{S}} \boldsymbol{exp}\left(-\frac{\boldsymbol{di,j^2}}{\boldsymbol{cd^2}}\right) \qquad \textbf{…(2.14)}$$

Where di, j is the distance between point i and j and cd is initial specified parameter as cutoff distance to determine the neighborhood radius to include 1% to 2% of the data in the neighborhood. In the second step is to calculate separation distance δi, which work to find the minimum distance between the point xi and any other point of higher density. Equation (2.15) show the calculation of the separation distance δi [95]:

$$\boldsymbol{\delta i} = \begin{cases} \boldsymbol{min}(\mathrm{d}\boldsymbol{ij}), & \boldsymbol{if\ \ density_{pi} < density\_pj} \\ & \boldsymbol{or} \\ \boldsymbol{max}(\mathrm{d}\boldsymbol{ij}), & \end{cases} \qquad \textbf{…(2.15)}$$

This method identifies the cluster centers by anomalously high $\boldsymbol{Pi}$ and $\boldsymbol{\delta i}$ after the local density and separation distance for each point have been computed. The Peak points (center points) are selected using a two-

dimensional decision graph with a local density as the horizontal axis and separation distance as the vertical axis [96]. Figure (2.20) is illustrated by the simple example, which shows 28 points representing the decision graph that identifies points 1 and 10, as cluster centers [97].

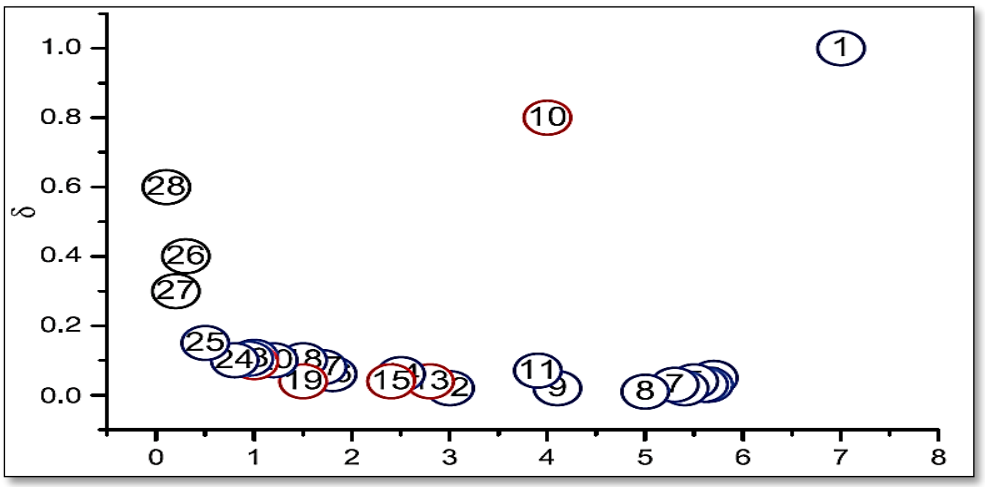


Figure (2.20): Illustrated By the Simple Example to Represent The Decision Graph

After determining the center points, the remaining points in the dataset will be assigned to the nearest center.

2.12 Artificial Neural Networks

Artificial Neural Networks (ANNs) are machine learning models inspired by human brain processes [98]. They are members of the parametric classifier family, with parameters such as a number of neurons/layers, weights, and biases, and a collection of hyper-parameters such as several epochs, learning rate, and batch size. A typical Neural Network (NN) can be represented as a directed, usually acyclic graph, with neurons as nodes and edges as synapsis and neurons organized into layers. The network architecture is formed intuitively by the way neurons are connected. It means that ANN could solve various problems, such as detecting intrusion in networks. As a result, ANN can be divided into two groups depending on their connection patterns:

- No-loop feed-forward networks. This network of neurons is organized into layers with unidirectional connections. Furthermore, topologies in this category produce only one set of outputs from a given input. The reaction to input is independent of the network state that came before it (also known as static networks).
- Loop-based recurrent (feedback) networks. A dynamic network, on the other hand, falls under this category. As a result of the feedback channels, it is bidirectional. Figure (2.21) shows the known ANN architectures [98].

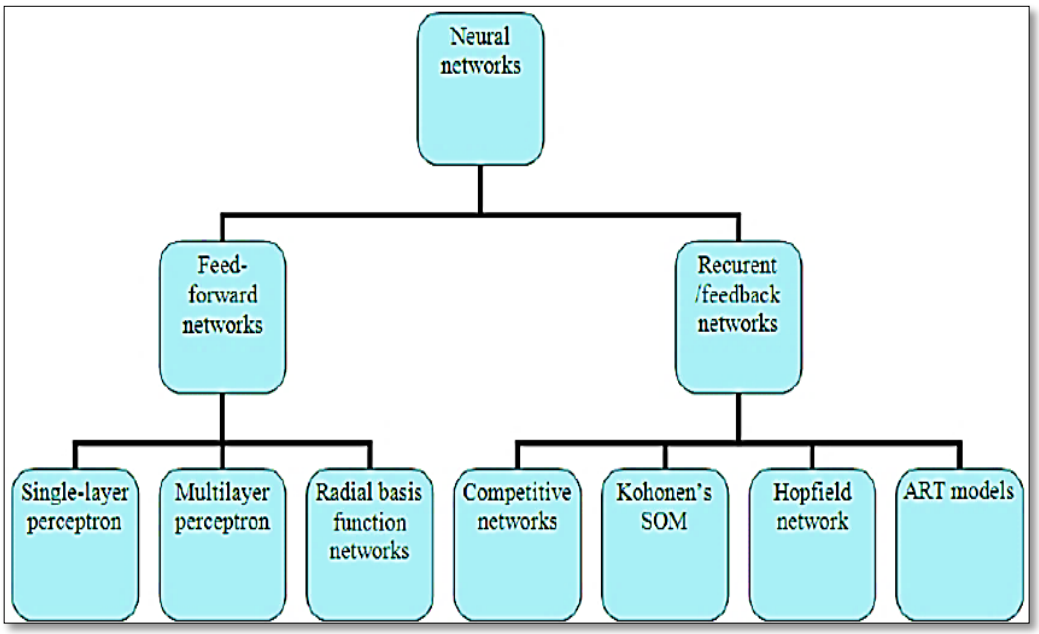


Figure (2.21): Artificial Neural Network Architectures

ANN is typically made up of an input layer, some hidden layers, and an output layer. Lastly, learning models must be tested once the training is completed to determine their effectiveness. This evaluation should require new data not include in the training set. Otherwise, the performance evaluation would be biased because the model has already seen that data. A validation set can also be used to compare the values of various parameters (for example, a learning rate or several neurons).

Following training, the value with the best validation set results is used. The entire network is then put to the test on the test set.

2.12.1 Multi-Layer Perceptron

A Multi-Layer Perceptron is a simple feed-forward neural network that differs from a linear perceptron in terms of activation function and layer count. It is made up of three layers: an input layer, a collection of hidden layers, and an output layer [99][100]. Its neurons use nonlinear activation functions and are trained using backpropagation. The Multi-Layer Perceptron algorithm is technically based on two operations: First, the Forward Propagation method is used to make a prediction, which is calculated using Equations (2.16) and (2.17) as shown below [101]:

$$\boldsymbol{Net_i = \sum_{j=1}^{n} w_{ij}\, x_i + b_i \quad , j = 1, 2, 3, \ldots n} \qquad \textbf{...(2.16)}$$

$$\boldsymbol{Y_i' = \varphi\,(Net_i)} \qquad \textbf{...(2.17)}$$

Where Net_i is show the result of net inputs *xi* multiplied by interconnection weight w_{ij}, $b_i$is bias, φ is the activation function, n number of outputs, and Y_i' Means actual output of the net.

The next operation is to train data based on the Back Propagation (BP) method, which is responsible to updates weights of MLP. After each training epoch, the BP method will update the weights based on product error. The error is basically calculated as Equation (2.18):

$$\boldsymbol{E = \frac{1}{n} \sum_{i=1}^{n} (y_{Actual\ i} - y_{desired\ i})^2} \qquad \textbf{...(2.18)}$$

2.12.2 Activation Functions

In the cause of the training process in the neural network may be suffering from saturation problem of the activation function. The activation function plays a perfect role in the process of network learning. However,

the activation function can classify into two types are: The first is the saturated activation function, which is commonly used for artificial neural network decision boundaries, such as the sigmoid activation function, which vanishines gradient problems and produces non-zero centered output value. However, unsaturated activation functions, such as the rectified linear unit, fall into the second category (ReLU). The ReLU activation function effectively solves the saturation problem and is less computationally expensive because it does not use exponential terms. Figure (2.22) show ReLU Activation Function plot[102] [103].

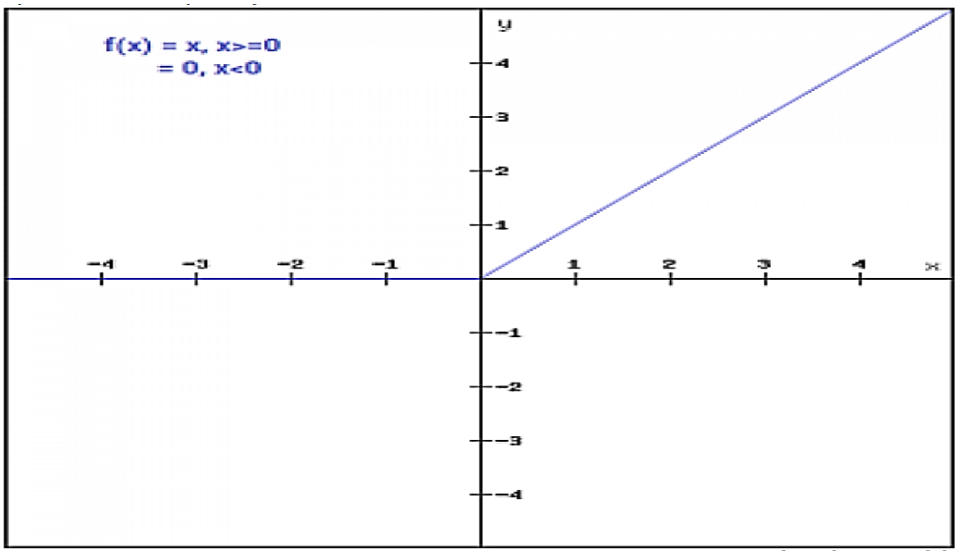


Figure (2.22): Relu Activation Function Plot

2.12.3 Optimization Methods of Neural Network

Neural Network algorithms often rely on optimizing some objective function, so the choice of the optimization algorithm is a crucial part of the learning process [104]. By minimizing the function, optimizers are used to solve optimization problems. Optimization Methods that only use the gradient or first derivative to search for the optimum are called first-order methods. The most common first-order optimization methods are: (Stochastic Gradient Descent (SGD), Adaptive Gradient (AdaGrad), AdaDelta, Momentum and Nesterov Accelerated Gradient (NAG),

Adam). Of all the methods listed above, Adam is thought to be the best. For each parameter, Adam calculates adaptive learning rates.

2.13 Performance Evaluation Metrics

A confusion matrix can calculate the performance measures for intrusion detection. The Confusion Matrix is a tool for summarizing a classifier's predicted performance on test data. The confusion matrix is used to determine all standard measures, such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). True Negative (TN) is the number of cases successfully forecasted as Normal class in cyber security, while True Positive (TP) is the number of instances correctly predicted as attacks, False Positive (FP) is the number of normal instances that are classified as an attack, and False Negative (FN) is the number of attack instances that are were classified as normal. For two or more classes, a confusion matrix can be utilized. Figure (2.23) illustrates the two binary class's confusion matrix [75].

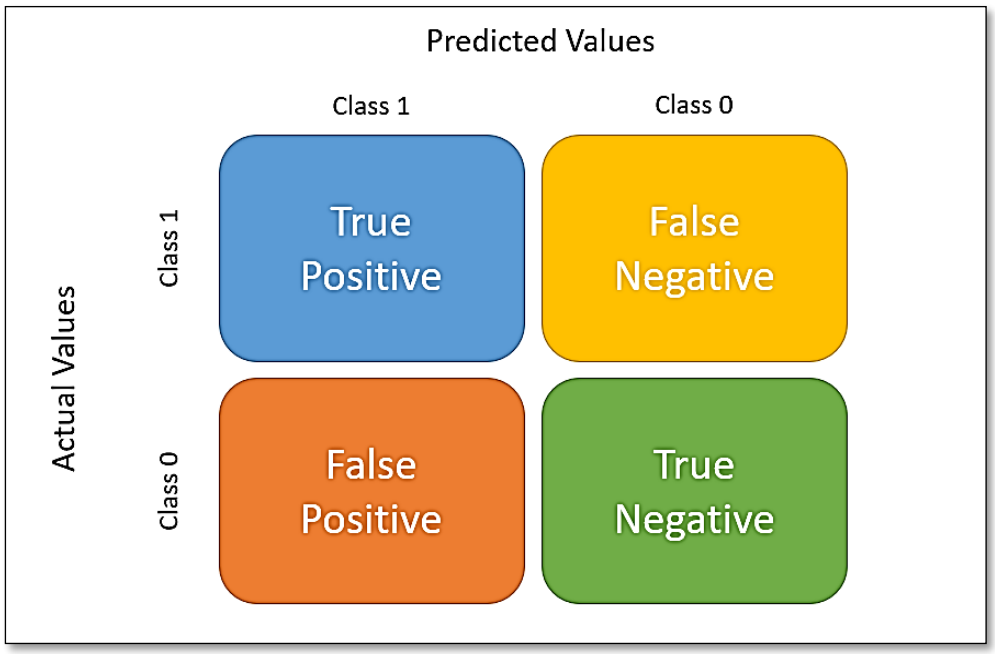


Figure (2.23): Confusion Matrix

To evaluate the results of the proposed system, standard metrics are used, such as measures like accuracy, precision, recall, and F1 score, while the Silhouette Index measure is presented to evaluate the quality of clusters [76][105].

- **Accuracy:** also called **Classification rate (CR),** is the percentage of True Positive and True Negative to the total of True and False Positive, True and False Negative for all classes. It can be calculated using the equation (2.19) below:

$$\boldsymbol{Accuracy = \frac{TP+TN}{TP+TN+FP+FN}} \quad \textbf{…(2.19)}$$

- **Recall*:*** also called **Detection Rate (DR), *True Positive Rate (TPR),* or Sensitivity in some fields.** It's the percentage of attacks that are successfully classified to the total number of attacks measured such as equation (2.20).

$$\boldsymbol{Recall = \frac{TP}{TP+FN}} \quad \textbf{…(2.20)}$$

- **Precision (PR):** is the fraction of data instances predicted as positive that is actually positive. Precision may be computed using equation (2.21) with the values of TP and FP obtained from the confusion matrix.

$$\boldsymbol{Precision = \frac{TP}{TP+FP}} \quad \textbf{…(2.21)}$$

- **F1.score:** also called **F-value**, is the harmonic mean of precision and recall. F1.score may be computed using equation (2.22).

$$\boldsymbol{F1\ score = \frac{2\times Precision\times Recall}{Precision+Recall}} \quad \textbf{…(2.22)}$$

- **Silhouette Index***:* The average silhouette width of a cluster's points is used to characterize its silhouette. The silhouette width for a given point x ranges from -1 to 1. If the value is close to -1, the point is closer on average to another cluster than the one to which it belongs. If the value is near to 1, the average distance between it and its own cluster is much less than the average distance between any other cluster. As a result, the clusters become more compact and divided as the silhouette increases. Mathematically, the silhouette width for each point x is defined by the equation (2.23) below [76]:

$$\textbf{Silhoute index} = \frac{\mathbf{x(i)-y(i)}}{\mathbf{max\{(x(i),y(i)\}}} \qquad \textbf{...(2.23)}$$

Where x(i) is a lower distance among cluster and object and *y(i)* is the higher distance between object and clusters.

Chapter Three

The Proposed System

Chapter Three
The Proposed System

3.1 Introduction

This chapter details the parts of the proposed system of a network intrusion detection approach using a combination model based on data mining and artificial neural network techniques.

The proposed system follows the methodology of the criteria used to develop and implement the NIDS systems. The main criterion required for the NIDS system can be divided into four major phases used in developing the NIDS system: (1)The preprocessing phase, where the problem is defined and the data is prepared for input to the next manipulation system. (2) The feature selection phase. (3) The development phase, the training system structures are worked out. (4)The performance measurement in the evaluation phase identifies the results and the problem objective.

3.2 General Proposed System Overview

The majority of network cyberattacks can be carried out using the TCP/IP suite's layers. For example, UDP/ICMP flooding attacks send UDP/ICMP packets to the victim, limiting the network connectivity and producing network congestion.

This proposed system using combination techniques based on data mining and artificial neural network approaches, which can used to detect network intrusions. Because network traffic contains linear and massive characteristics, modelling and analyzing network traffic in intrusion detection situations is difficult.

A general proposed system model consists of four phases, implemented to train system on detection and determine the type of attacks, started in preprocessing data, feature selection, clustering, and artificial neural network, ended in aggregation to predict the attack types. The general architecture of the proposed system for network intrusion detection is shown in Figure (3.1).

The first phase is preprocessing; a one-hot-encoding approach is used to transfer a single nominal attribute into n- numeric attributes, where n is the number of unique attributes in the same column. This technique has the advantage of producing binary vector rather than ordinal results and keeping everything in an orthogonal vector space. The next operation is to normalize data using the Min-Max normalization scaler.

The second phase is the feature selection phase, which is based on A proactive Bat optimization algorithm to eliminate redundant and irrelevant attributes; it also helps increase the homogeneity of training data distribution of the clustering process and enhance the prediction process.

The selected features are used as input to the Improved DPC clustering algorithm in the training phase. Using a robust data mining cluster analysis (IDPC) algorithm divides the complex network dataset into two distinct clusters, each cluster as far away from the other as possible. Network traffic that has to low rate attacks has similar characteristics and is therefore aggregated into the same cluster. This case will solve the imbalance problem of multiple classes and increasing the detection rate of these classes compared to the original dataset. The output of each cluster data point is used as input to separate ANN classifier. In contrast, the predictions of these ANN classifiers are aggregated into the

final output of the intrusion detection, which is aggregated with the help of the fractal fuzzy membership degree.

Finally, the performance measures for system evaluation are applied to the outputs such as Accuracy, Recall, Precision and F1 score to evaluate the results.

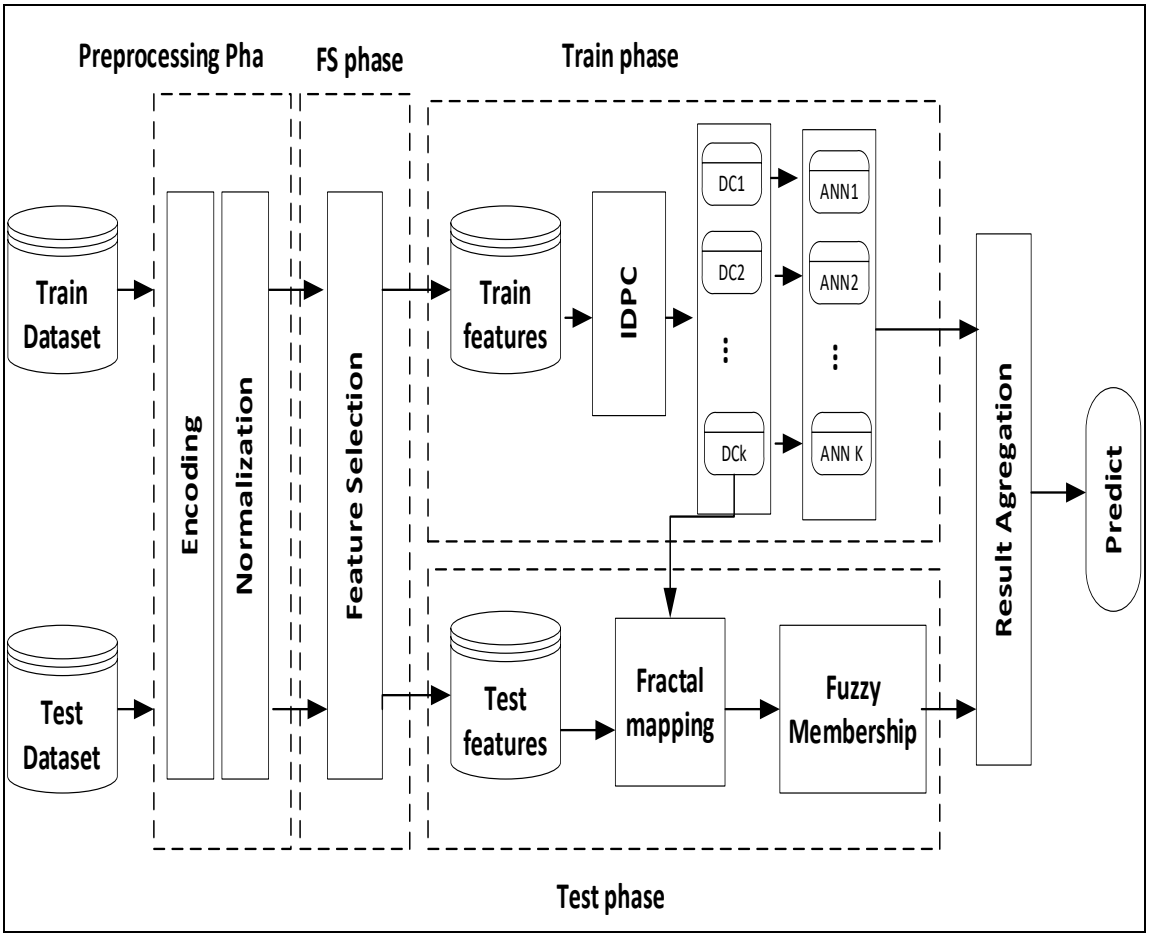


Figure (3.1): The Proposed System Architecture

The following subsection shows the parts of the proposed work in detail.

3.2.1 Network Datasets

The proposed system used two datasets with extensive network flows, NSL_KDD and UNSW_NB15 datasets, as inputs to a NIDS. Each dataset consists of set network packets, and each packet includes several attributes with different data types (e.g., binary, float, nominal and integer).

More details about the description of two datasets (NSL_KDD, UNSW_NB15) are shows in Appendix A.

3.3 Preprocessing Phase

In this stage, two datasets (NSL_KDD, UNSW_NB15) are used as inputs to the preprocessing phase of the proposed NIDS. The following subsections explain the preprocessing steps.

3.3.1 Data Transformation using One-Hot encoding

In this step, **one hot encoding** is proposed to convert the nominal attributes to numeric attributes for two datasets and analyze the network characteristics for normal and attack traffic.

The non-numeric attributes have converted to numeric attributes using a one-hot encoding method. In the case of the NSL_KDD dataset, which has 38 numeric and three nominal attributes. In the case of the UNSW_NB15 dataset, which has 39 numeric and three nominal attributes.

3.3.2 Data Normalization using Min-Max Method

Normalizing the dataset for specific scaling values of each attribute into a particular range is a necessary step of the preprocessing phase. The advantage of this method is that it removes bias from the datasets without changing their statistical properties. The Min-max normalization section is defined based on equation (2.1) and the algorithm (3.1).

One of the most popular approaches to normalize data in this step is to use Min-max method. Each attribute has a minimum value will sit to zero, and the attribute with maximum value will sit to one, and the rest of the other values are set to a decimal between 0 and 1. The Min-max scaling is visualized in Figure (3.2).

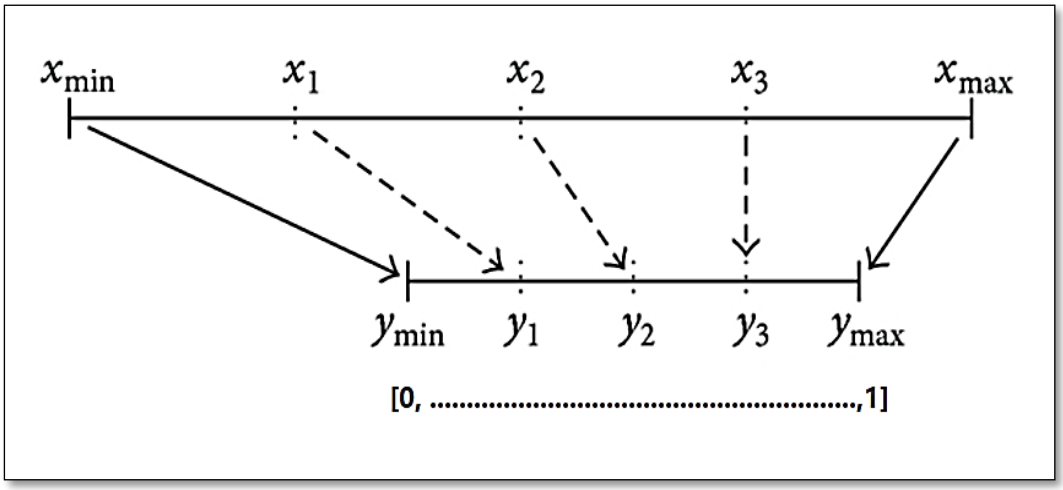


Figure (3.2): Min-max scaling.

Algorithm (3.1) : Min-max Normalization

Input : *D_List* **(List of attributes values)**

Output : $norm_x$ **(Normalized List of attributes values)**

Begin {

1. Repeat **{**
2. foreach **attribute value** x **in** *D_List* do **{**
3. $norm(xi') = \frac{xi - min(xi)}{max(xi) - min(xi)}$
4. **}// End foreach**
5. until **(End of** *D_List***)**
6. Return $norm_x$ **(normalized values with n features)**
7. } // End Algorithm

In the algorithm (3.1), xi is input data,$max(xi), min(xi)$ is maximum and minimum value in data respectively, $norm(xi')$ the output data after normalization.

3.4 Feature Selection Phase

In this phase, A Proactive Bat Search Optimization Algorithm (ABA) selects the most relevant features before applying the cluster analysis. ABA is an intelligent behavior of self-organized that inspired by the survival of the fittest theory. Technically, the life of a bat search

restarts when having no enhanced search progress. This model is developed to regenerate the population (replace the current population with a new population); when the model has been fumbling, the search process goes forward the stagnation. The process of regenerating a new population increases the probability to enhance the exploration of the swarm model. Figure (3.3) shows a proactive swarm optimization search process.

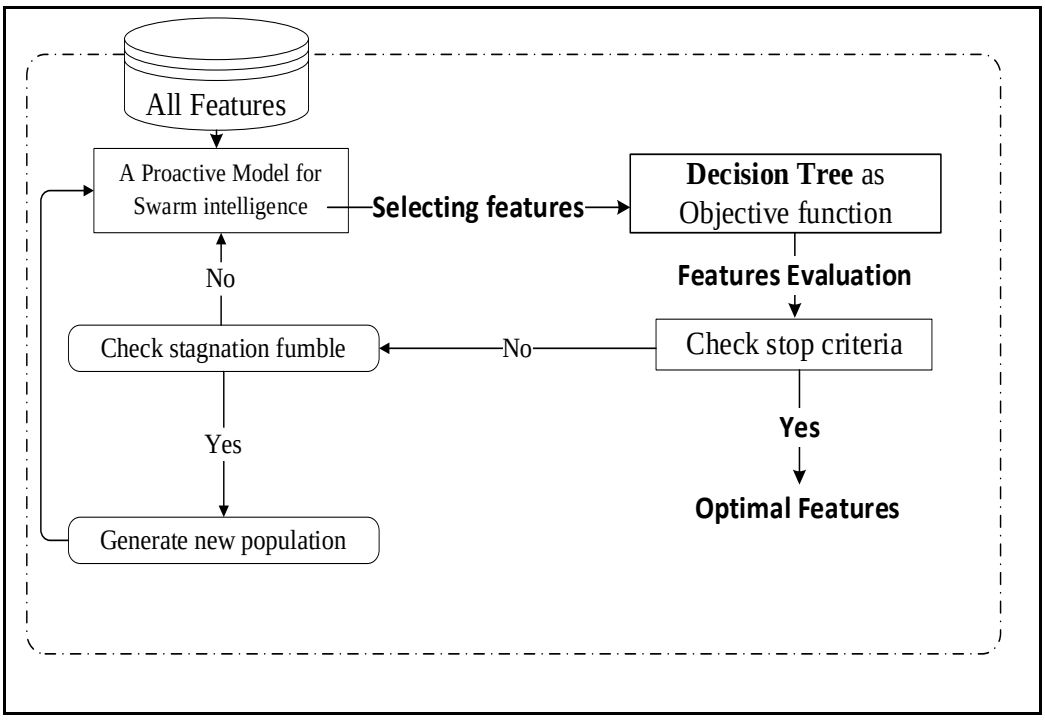


Figure (3.3): A Proactive Swarm Optimization Searching Process

The bat stagnation on local optimum often happens because of weakness in exploration or exploitation strategy of a searching process. In the case of the exploitation, the process focuses on using the best solution to enhance the search process, while the exploration will seek the swarm's ability to explore possible solutions within the searching space. Therefore, a high diversity will increase by exploring new search spaces. The feature selection method uses chapter two equations from the subsection (2.7.3.1). The algorithm (3.2), (3.3), (3.4) illustrates the steps of proposed feature selection algorithm.

Algorithm (3.2) : A proactive Bat Algorithm for Features Selection

Input : Bat parameters:

Population size (P_z), Problem dimensions (m), upper limit (U) ,

lower limit (L), F_{max}, F_{min}

D← *Normalize* **Dataset (NSL_KDD dataset OR UNSW_NB15 dataset)**

Feature selection parameter:

$\theta \leftarrow threshold$

Output: $Best_{cost}$, $Best_{featur}$

1. Begin **{**
2. $\sigma \leftarrow 0$ **//set zero to σ because no a stagnation in begging**
3. **P ← Population initialization (P_z, U, L, m)** // algorithm(3.3)
4. **$V_i \leftarrow 0$, $F_i \leftarrow 0$, //set initialize A, r**
5. While $iter < max\_iteration$ **and not satisfy the termination, criteria do** {
6. For each $X \in P$ do**{**
7. $F_i \leftarrow F_{min} + (F_{max} - F_{min})\beta$
8. $V_i^{iter} \leftarrow V_i^{iter-1} + (X_i^{iter-1} - G_{best}) * F_i$
9. $X_i^{iter} \leftarrow X_i^{iter-1} + V_i^{iter}$
10. If **random** > r_i, then {
11. $X_i^{iter} = X_i^{iter} + \varepsilon * A$
12. }//EndFor
13. **New_D, F_{pos}**← Object Function for Select Features **(D, X_i^{iter}, θ)** // ---------- Algorithm (3.4)
14. **$cost_{pos}$← accuracy of (New_D)**
15. If **$cost_{pos}$** > $Best_{cost}$ then {
1. $Best_{cost} \leftarrow cost_{pos}$
16. $G_{best} \leftarrow X_i^{iter}$
17. $Best_{featuer} \leftarrow F_{pos}$
18. $A_i^{iter} \leftarrow \alpha * A_i^{iter-1}$
19. $r_i^{iter} \leftarrow r_i^{iter-1} * (1 - e^{-\gamma})$
20. $\sigma \leftarrow 0$
21. Else

```
22.                    σ ← σ + 1
23.                }//EndIf
2.             }// EndForEach
24.              If  σ > 2 ∗ P_z  then {
25.                    P ← Regenerate Population initialization //
   algorithm(3.3)
26.              }//EndIf
27.           }// EndWhile
28.      Return Best_cost, Best_featuer
29.   //End Algorithm
```

Firstly, the initial Population size ($\boldsymbol{P_z}$), Problem dimensions ($\boldsymbol{m}$), upper limit ($\boldsymbol{U}$), lower limit ($\boldsymbol{L}$), $\boldsymbol{F_{max}, F_{min}}$ are initialized for each bat as Algorithm (3.3).

Algorithm (3.3) : Population initialization

Input: **upper limit (U) , lower limit (L) , vector size (V) , population size (P_z)**

Output: ***X*** **(initial population)**

```
1.  Begin {
2.          Population ← [] # population stored as a list
3.          For i in range(P_z) do
4.                pos = [uniform(U, L) for _ in range(V)] // new candidate
   solution
5.                X [i] ← pos
6.          }// EndFor
7.         Return X
8.         } //End Algorithm
```

Algorithm (3.4): Object Function for Select optimal Features

Input: **D← dataset, optimization vector (op), θ ← threshold**

Output: **data after features selection (New_D), optimal Features (F)**

```
1.    Begin {
```

```
2.    For each x[i] ∈ op and x[i] > θ{
3.        F ← F ∪ i  // store the index of optimal feature
4.    }// EndFor
5.    New_D ← [] // new dataset after selected features
6.    Foreach d_j ∈ D do{
7.        d′ ← []
8.        For i in range (size(d_j) and i ∈ F do{
9.            d′ ← d′ ∪ d_j[i] // store the features that has index in F
10.       }//EndFor
11.       New_D ← append (d′)
12.   }//EndFor
13.   Performance ← obj(New_D) // obj is DT algorithm
14.   Return New_D , F , Performance
15.   }//End Algorithm
```

3.5 Forming of Clustering Analysis

Cluster analysis is a type of unsupervised data mining and machine learning that does not require the presence of a class (target) in the data. It clusters similar objects together, with minimized intra-cluster distances and inter-cluster distances that are maximized. The cluster analysis technique is depicted in Figure (3.4).

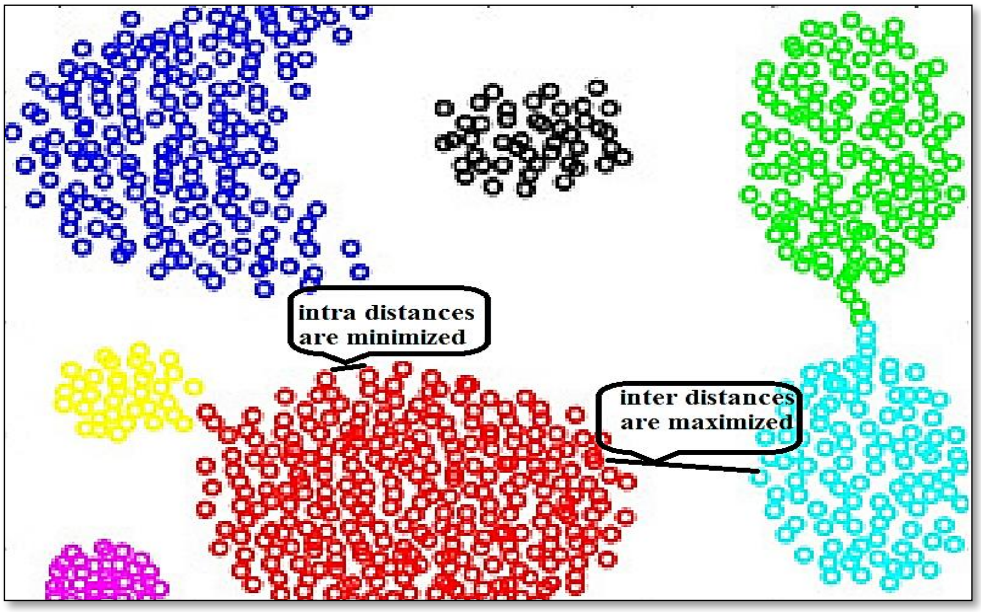


Figure (3.4): Forming Of Clustering Process

There is a need to modify the cluster parameters and distance measures to achieve the best-desired clusters output in the cluster analysis. Forming clusters is based on Density-based methods, which is the Improved Density Peak Clustering algorithm. The novel modification of this algorithm is presented in the following sub-sections:

3.5.1 Improved Density Peak Clustering Algorithm

It is clearly shown in section (2.11), the core idea of DPC to calculate cluster centers depend on measuring both the local Density and the separation distance. Both those measures depend on calculating the distance between any two points with the given dataset. The original DPC uses the Euclidean distance measure to find the distance between two data points. However, the Euclidean distance causes misclassifications when the complex dataset and has high dimensional features. Therefore, the Gaussian kernel distance is used to calculate the distance between two points in high dimensional data. This is the most crucial stage in handling

the geometry, increasing accuracy, and lowering the false alarm rate. Algorithm (3.5) illustrates the Improved DPC steps for the training stage.

Algorithm (3.5) : Improved DPC steps for the training stage

Input: **data← traindata,**

cd **← threshold for defining the neighborhood,**

Output: **the K sub train sets DC1, DC2… DCK.**

1-Begin **{**

2- For i in range (len (self.data) **):{**

3- ***np_distance*** **=** $\exp\left(-\frac{||\text{self.data[i]}-\text{self.data[j]}||^2}{(2(\sigma^2))}\right)$

4- ***cd*** **= int(cd * len(data))** **// *cd* = 2% of all data size**

5- ***density_pi*** $= \sum_{i,j\in S} exp\left(-\frac{np\_distance^2}{cd^2}\right)$

6- }//EndFor

7- For i2 in range(len(*density_pi*)):{

8- $\wp(\text{data[i]}) = \begin{cases} min(np\_distance), & if\ density_{pi} < density\_pj \\ max(np\_distance) & or \end{cases}$

9- ***delta*** **=** $\wp$**(data[i])**

10- }//EndFor

11- **For i3 in range(len(*density_pi*)):{** **// Compute ϒi (gama_i) parameter**

12- ***density_pi*** **= listed(range(len(*density_pi*))**

13- ***delta*** **= listed (range(len(*delta*))**

14- ***gama_i = density_pi[i] * delta[i]***

//--- select highest *gama_i* to be the peak point (centroid)

15- **plot.scter1(*density_pi*,dalta)**

16- **Return (*peakpoints*)**

17- }//EndFor

18- **For j in range(len(self.data)): { // distributed train points on nearest peak points**

19- If j2 not in ind_center: **// ind_center is index of *peakpoints***

20- **calculate *Gaussian kernal distance*** // Algorithm (3.6)

21- }//EndFor

22- return **the K sub train sets DC1, DC2, _ _ _ , DCK.**

23- }//End Algorithm

In the Algorithm (3.5), Where $\boldsymbol{\sigma} = scaling\ parameter$, and || *xi* – *xj* || is the Euclidean metric when($xi \neq xj$), where (xi and xj) are points in the dataset. Where *cd* refers to the cutoff distance, which is the initial specified parameter that works as the threshold to control the weight degradation rate. The suitable suggested parameter to determine the neighborhood radius *cd* is 2% of the training data in the neighborhood.

Algorithm (3.6) illustrates a process of calculate Gaussian kernel distance between all points to distribute nonpeak points in the training dataset.

Algorithm (3.6) : Gaussian Kernel distance

Input : ***peakpoints , self.data, ind_center,*** **//list of centroid points , training data points, index of centroid points,**

Output**:** ***self.clusterdata*** // **array list of train data clusters**

1. Begin **{**
2. *foreach (***peakpoints***) do {*
3. ***for (i = 1 to self.data){***
4. ***if self.data not in ind_center:***
5. ***dis*** $\left(\overrightarrow{\mathbf{data[i]}}, \overrightarrow{\mathbf{data[j]}} \right)$ **= exp**$\left(-\frac{||\mathbf{self.data[i]-self.data[j]}||^{\mathbf{2}}}{\left(\mathbf{2(\sigma^2)}\right)}\right)$
6. *end for*
7. ***assign a label to point based on a cluster with minimum*** *Gaussian Kernel* ***distance and save point in self.clusterdata***
8. *end foreach*
9. End Algorithm

In algorithm (3.5), the clustering procedure extracts only two points, which are the center (peak) of train data. In algorithm (3.6), the rest of the non-peak data points will assign to the same cluster as its nearest data point.

3.5.2 Fractal Fuzzy Membership function (FFM) on Test sample

After assigning each point in the train data to the most appropriate cluster, calculate the Fuzzy Membership function between the test sample and data point in each cluster. The fuzzy membership finds the relevant query item (*test sample*) ratio to each cluster ($DC_1, DC_2, \ldots . DC_N$) in the clustering pool. It depends on the distance between the *test sample* and the nearest point in each cluster. This process will create a Fuzzy Membership degree of test sample related to each cluster. In the last step, the Fuzzy Membership degree will multiply with the output prediction of the ANN classifier to reduce the bias of the low rate class and increase the accuracy of the result. Figure (3.5) declares the test's Fuzzy Membership degree related to the cluster pool.

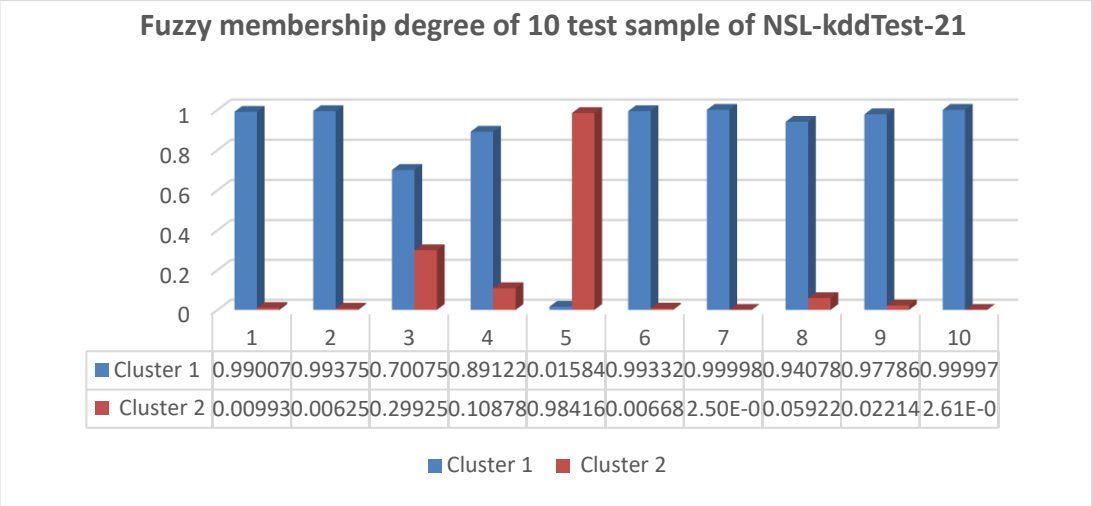
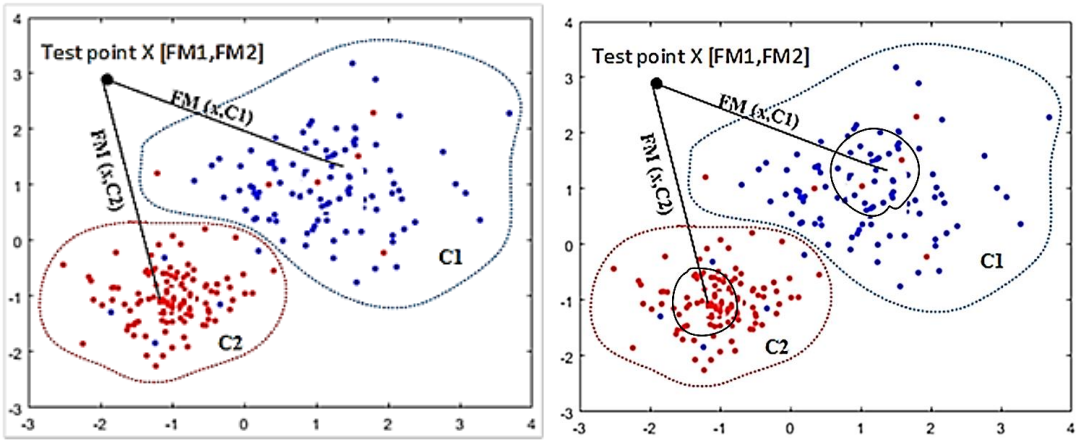


Figure (3.5): Declare The Degree Of The Test Relation To Cluster Pool

To find the FM needs to visit all points in each cluster. This process potentially requires some processing time and computational complexity, therefore, a novel FM based on fractal factor is proposed to overcome the time consuming and computational complexity. This process will extract subcultures ($FDC'_1, FDC'_2, \ldots . FDC'_n$) from the main clusters ($DC_1, DC_2, \ldots . DC_N$). The new subcultures have high similarity in behavior

to the test item (*test sample*). Figure (3.6) illustrates how fractal metrics reduced the search space.



a-Without fractal b-with fractal

Figure (3.6): Illustrates How Fractal Metric Reduced The Search Space.

The FFM has inversely proportional with the distance (d) between the nearest point in sub cluster (FDC'_n) and the test point. i.e. the point in a cluster with a small distance value to *test sample*, the FFM degree will be greater than the point in a cluster with a high distance value. The FFM function uses equations from section (2.10) in chapter two. Figure (3.7) illustrates the Fractal Fuzzy steps. Algorithm (3.7) illustrates the Fractal Fuzzy Membership function.

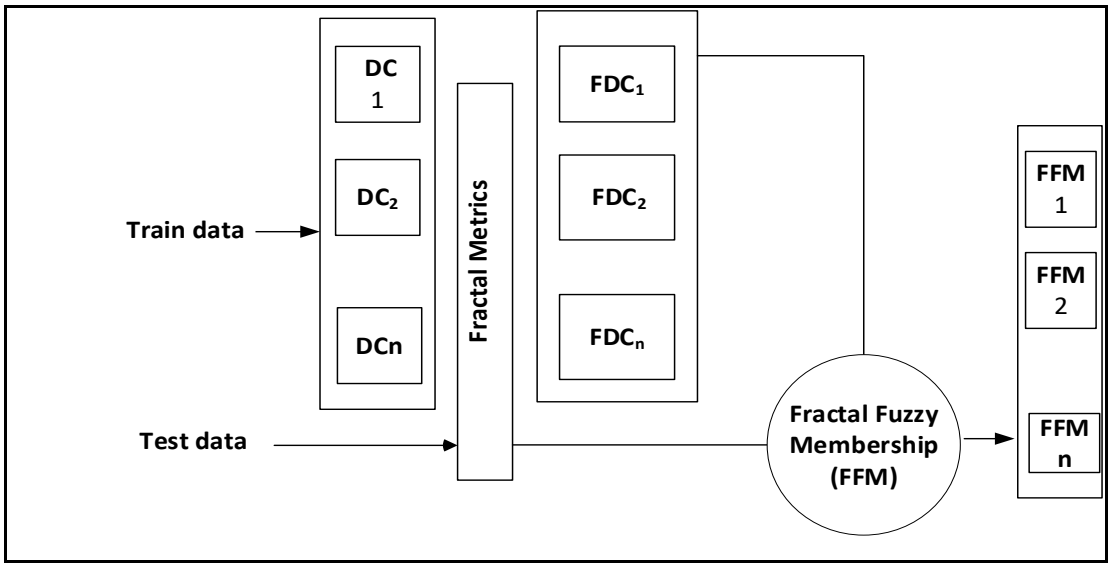


Figure (3.7): Illustrates The Fractal Fuzzy Steps.

Algorithm (3.7) : Fractal Fuzzy Membership function

Input: **TD←test dataset ,** $\boldsymbol{\beta} \leftarrow \boldsymbol{threshold}$

$\mathbf{DC_1, DC_2, \ldots DC_N}$ **←train clusters,**

Output: **Fractal Fuzzy Membership (*FFM*)**

1. Begin **{**

2. For each **i** ∈ **DC** do {

3. $\mathbf{FDC_i}$**=[]**

$\boldsymbol{R' = \frac{1}{N}\sum_{i=1}^{N} DC_i}$ **//R' is the mean of DC,**

n is dimensional of point DC_i

4. $\mathbf{FDC_i} = \frac{(\sum_{i=1}^{n}(DC-R')^2)^2}{\sum_{i=1}^{n}(DC-R')^4}$ **// fractal of train clusters**

5. **}**// EndFor

6. **FFM←** **[]**

7. For each **t** ∈ **TD** do {

8. $\mathbf{FFM_i} \leftarrow$ **[]**

9. $\mathbf{FDT}_t = \frac{(\sum_{t=1}^{n}(TD-R')^2)^2}{\sum_{t=1}^{n}(TD-R')^4}$ **// fractal of test dataset**

10. $\mathbf{DC'_1, DC'_2, \ldots DC'_n} \leftarrow \{\mathbf{subcluster_t} \mid \frac{|\mathbf{FDT}_t - \mathbf{FDC_N}|}{\mathbf{FDT}_t} \leq \boldsymbol{\beta}\}$

11. $\boldsymbol{d_t} \leftarrow$ **[]**

12. For each $\mathbf{DC'}$ **in** $\mathbf{subcluster_t}$ do **{**

13. $\boldsymbol{d_t}$**[j] ← the nearest distance (DC′, t)**

14. **}**// EndFor

15. For each **d** ∈ $\boldsymbol{d_t}$ **and k ∈ cluster no.**

16. $\mathbf{FFM_i[k]} \leftarrow \frac{1}{\sum_{k=1}^{n}(\frac{d}{d_t[k]})^2}$ **// n is the number of clusters**

17. **}**// EndFor

18. **Append the** $\boldsymbol{FFM_i}$ **to** $\boldsymbol{FFM}$

19. **}**// EndFor

20. Return $\boldsymbol{FFM}$

21. }}//End Algorithm

3.6 ANN Classification Model

The result of the previous step is two sub-clusters DC_1, DC_2 from the training dataset. Each sub-cluster is used to train the corresponding different ANN classifier. These ANNs differ from each other because they are trained on different subsets. Each ANN has an input layer, hidden layers and an output layer. Algorithm (3.8) illustrates Neural Network ANN classification.

Algorithm (3.8) : Neural Network ANN

Input**: cluster data (DCi) , number of epoch (*n_epoch*=) ,**

Output**: Predict of each ANN(Predict_ANNi)**

1. Begin{
2. For **i= 1 to *n_epoch*** do
3. $\boldsymbol{Net_i = \sum_{j=1}^{n} w_{ij} x_i + bi}$ **, j = 1, 2, 3, … … … … n**
4. $\boldsymbol{Y_i^{out} = \varphi\ (Net_i)}$**// φ is ReLU Activation function**
   // Y_i^{out} is the output of units in the output layer.
5. For each **network output unit Y_i^{out}{**
6. $\boldsymbol{E = \frac{1}{n} \sum_{i=1}^{n} (Y_i^{out} - Y_i^{desired})^2}$**// calculate its error**
7. **Update each network weight**
8. }//EndFor
9. }//EndFor
10. }//EndWhile
11. For each **test sample (x_j), ϵ test** do
12. **Pridact$_{xj}$ ← Forward (weight, x_j)**
13. **Predict_ANNi[j] ← Pridact$_{xj}$**
14. }//EndForEach
15. **Predict_ANNi**
16. }//End Algorithm

3.7 Aggregation Model

Aggregation is the process of connecting modules (by linking outputs of one module to output of another). In this proposed system, the production of each ANN is multiplied with corresponding fuzzy membership, then another production of ANN. The predictions result of the test sample *xj* in each sub- *ANN i* classifier is defined as *ANNi (xj).* Figure (3.8) declare the aggregation method predict the final output. The algorithm (3.9) illustrates the step of the aggregation model.

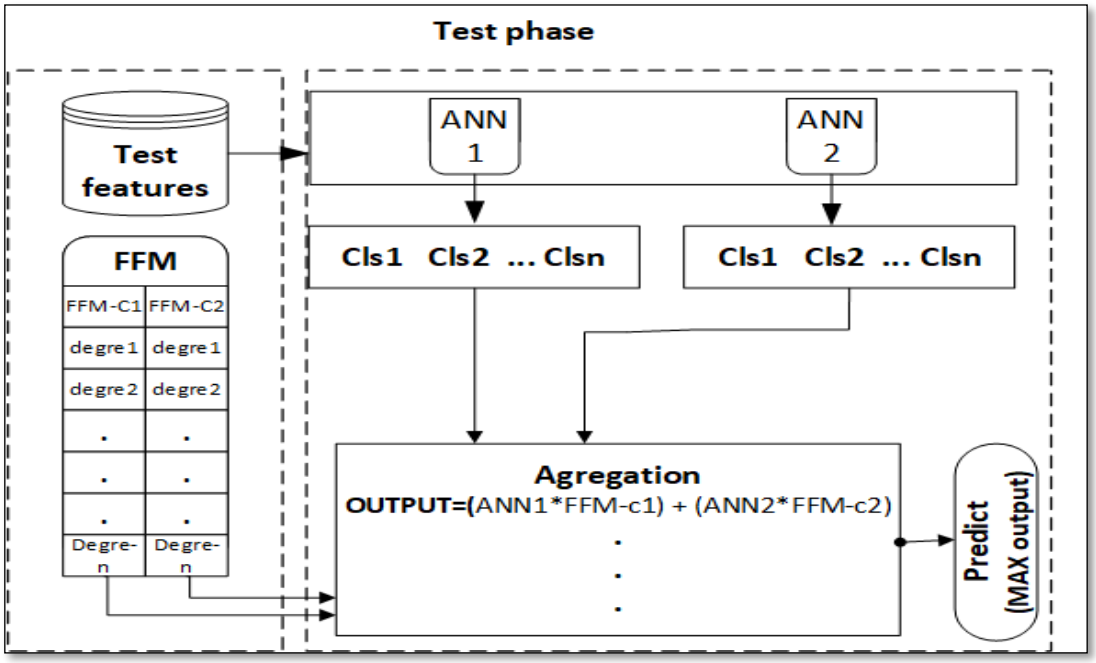


Figure (3.8): Illustrates The Aggregation Method Predict Final Output.

Algorithm (3.9) : Aggregation Model

Input: **TD←test dataset , FM←Fuzzy Model ,**

ANN_1 , ANN_2, … ANN_n, ← **Neural Network model (ANN)**

Output: **Prediction (Pre)**

1. Begin **{**
2. **Pre ← []**
3. For each **t** ∈ **TD** do {
4. For $\mathbf{ANN_i} \in$ **ANN k** ∈ **cluster no.** do {
5. $\mathbf{T_{out}} \leftarrow \mathbf{t_{out}} + MAX\left(\mathbf{FFM_{t,k}} \cdot \mathbf{ANN_i(t)}\right)$

$//out = \sum_{i=1}^{k} FFM\ t, k * MLPi(t)$

6. **}**//EndFor

7. **Pre ← Map to predict (t_{out}) // store the predict class of (t)**

8. **}//End** For each

9. Return **Pre**

10. }//End Algorithm

3.8 System Evaluation

To evaluate the proposed IDS model's effectiveness, the unsupervised evaluation metrics Silhouette Index is calculated to evaluate the performance of training clusters and confusion matrix is used for ANN classification performance to compute the achievable testing data accuracy, precision, recall, f1-score which, has mentioned in section (2.13).

3.8.1 Unsupervised evaluation metrics

Unlike supervised learning methods, measuring the performance of a clustering algorithm is more complicated than just counting the number of errors, precision, and recall. The training data clusters are evaluated using the Silhouette Index.

3.8.2 Supervise evaluation metrics

The metrics for evaluating the ANN classifier's performance that has been carried out in our proposed model include (accuracy, precision, Recall, f1scor). The calculation of these metrics depends on four types of basic computation metrics of the confusion matrix called True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

The best performance of the classification algorithm should be with high accuracy, high detection rate, and high F1score. Algorithm (3.10) is illustrated to calculate the confusion matrix mentioned in section (2.13).

Algorithm (3.10) : Confusion Matrix

Input **:** Test_ Pre ← **(Array list of the test points)**

Output : CM ← (Binary Confusion Matrix)

```
1.  Begin
2.    Set   TN←0, TP←0, FP←0, FN←0
3.    for i=0 to Test_ Pre.count
4.        foreach (packet t in Test_ Pre [i] )do {
5.          if (t.actual=="normal") && (t.predict =="normal")
6.                TN:=TN+1;
7.          else if (t.actual=="normal") && (t.predict =="attack")
8.                FP:=FP+1
9.          else if ((t.actual=="attack") && (t.predict =="normal")
10.               FN:=FN+1;
11.         else if ((t.actual=="attack") && (t.predict =="attack")
12.               TP:=TP+1;
13.       }//EndFor
14.   }//EndFor
15.   Add TN, FP, FN, TP to CM
16. }//End Algorithm
```

Chapter Four

Implementation and Results

Chapter Four

Implementation and Results

4.1 Introduction

This chapter explains the implementation and the results of the proposed system. The proposed system is tested with different malicious data sets such as NSL_KDD test-21, NSL_KDD test+ and UNSW_NB15. These datasets are network intrusion datasets launched against a victim by real attackers. The following subsections present the descriptions of these datasets.

4.2 Data Sets Description

NSL_KDD, UNSW_NB15 datasets consist of a large number of packets; the first one contains four major attacks, while the second data set contains nine major attacks. The features of these data sets include TCP/IP header information of TCP/IP suite, which has mentioned in section (2.2). The following subsections present the details of these datasets.

4.2.1 System Requirement

Hardware: Processor Intel ® Core (™)i7-8750H CPU, Ram 32 GB, Storage 500 GB, Freq. 2.20 GHz.

Operating System windows 10 64bit.

Programming Language PyCharm community 2019.3.

4.2.2 NSL_KDD Data Set

The NSL_KDD data set is not the most recent, but it is a new version that addresses the shortcomings of the KDDCup1999 data set. The NSL_KDD data set consists of four files are: the first file is called

NSL_KDDTrain+ for training, which is the complete training data; the second file is called 20 Percent Training Set, which is also for training data. Moreover, two files for testing are called NSL_KDDTest+ and NSL_KDDTest-21. Table (4.1) below show a summary of these files**.**

Table (4.1): Description of NSL_KDD Dataset Files.

File name	Description	Records	Attacks %
NSL_KDDTrain+	Full NSL_KDD training set	125,973	46.5%
NSL_KDD20 Percent Training	A 20% of the full NSL_KDDTrin+ of training data set	25,192	46.6%
NSL_KDDTest+	Full NSL_KDD testing set	22,544	56.9%
NSL_KDDTest-21	Is a subset of NSL_KDD testing set (more complex)	11,850	81.8%

The NSL_KDDTrain+ dataset has 22 different sub-types of attacks aggregated to four major types. In comparison, the NSL_KDDTest-21 and NSL_KDDTest+ datasets have 39 sub-types of attacks, which means the testing dataset has an additional 17 novel, unknown sub-attacks not available in the training datasets. Figure (4.1) shows the NSL_KDD sub-attacks types of training set. Figure (4.2) shows the NSL_KDD additional sub-attacks types of testing set.

The total description of NSL_KDD features has explained in appendix A.

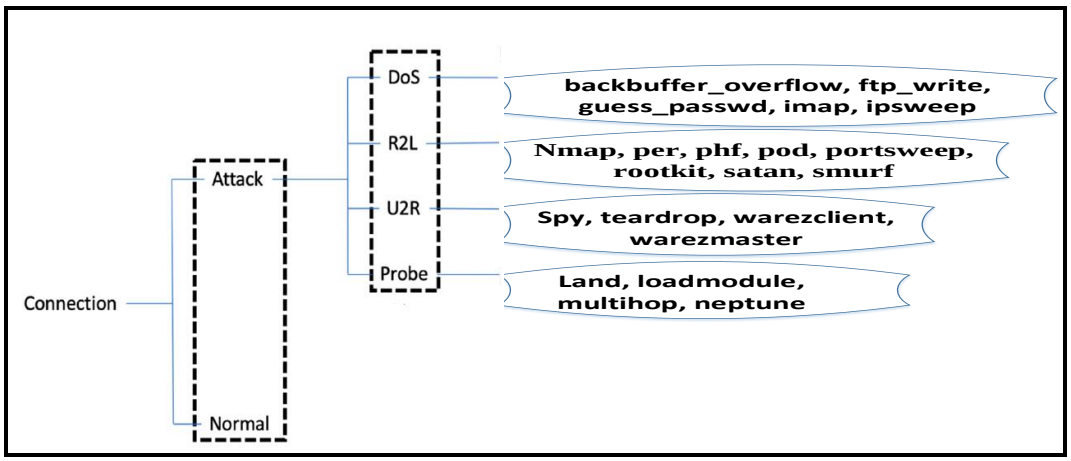


Figure (4.1): NSL_KDD Sub-Attacks Types of Training set.

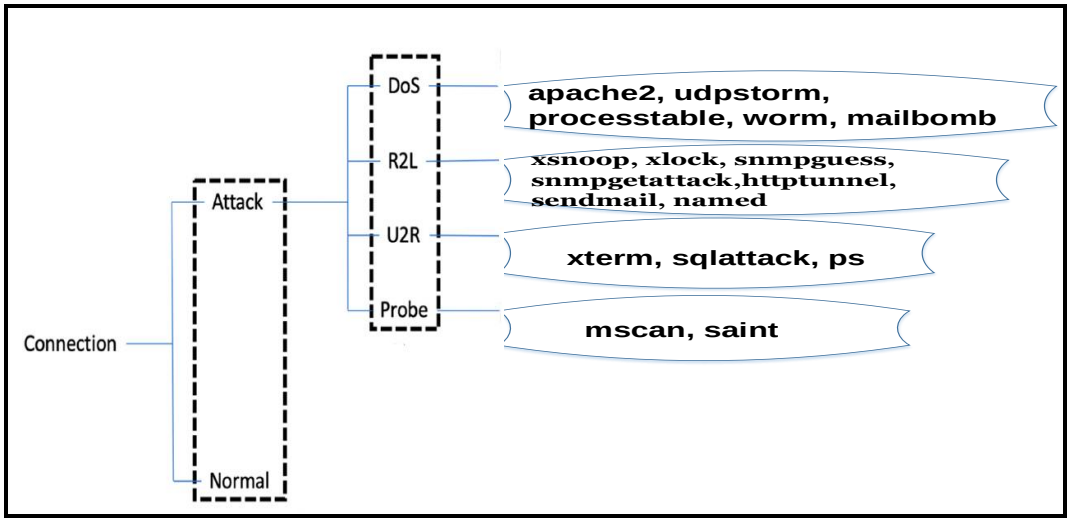


Figure (4.2): NSL_KDD Additional Sub-Attacks Types of Testing set.

In the proposed system, the training process is done by selecting a randomly 20% of the full training (NSL_KDDTrain+) dataset based on random method. In contrast, the testing process is evaluated based on full NSL_KDDTest-21 and NSL_KDDTest+ datasets. The description of NSL_KDD training and testing of the proposed system is shown in table (4.2).

Table (4.2): Description Of 20% NSL_KDD Training And Full Testing Data Set Files.

	Training Dataset	Testing Dataset	
Category	20 % from full NSL_KDDTrain+	NSL_KDDTest-21	NSL_KDDTest+
Dos	9234	4342	7458
Normal	13,449	2152	9711
Probe	2289	2402	2421
U2R	12	200	200
R2L	197	2754	2754
Total	25,192	11,850	22544

Figure (4.3) shows the class description of 20% from full NSL_KDD training data with the imbalance of attacks classes such as R2l and U2r with very few records in the training dataset.

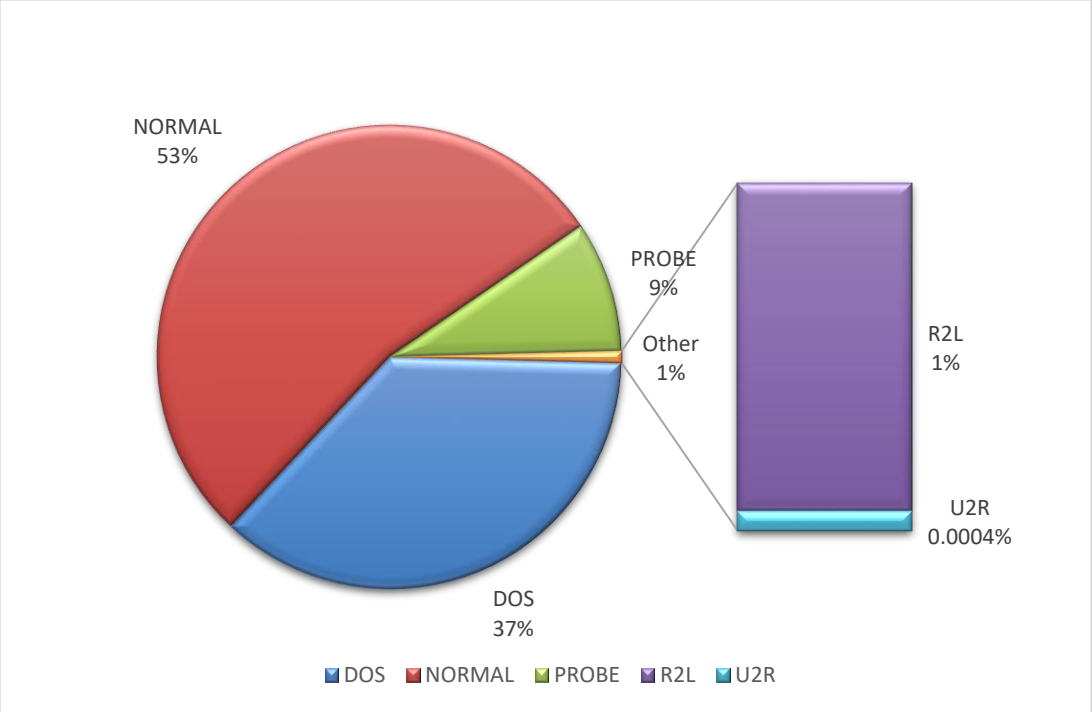


Figure (4.3): Class Distribution of 20% from NSL_KDDTrain+ Data Set

4.2.3 UNSW_NB15 Data Set

The UNSW_NB15 consists of two million packets; each packet has 49 attributes with the class label in the whole dataset, while the partitions UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv is configured as partitions of the UNSW_NB15 dataset with only 42 attributes pulse label of class. The partition data set has six attributes (srcip , sport , dstip , dsport , Stime and Ltime) removed from the total data set.

In the proposed system, utilize the random state method to choose 20% of train data from a partition UNSW_NB15_training set as training data and 20% of test data from a partition UNSW_NB15_testing set as testing data in our proposed system such as table (4.3).

Table (4.3): The Category Descriptions of UNSW_NB15 Training and Testing Set Files.

Category	20% UNSW NB15-train set	20% UNSW NB15-test set
Normal	11190	7349
Fuzzers	3599	1205
Analysis	406	155
Backdoors	336	114
DoS	2491	833
Exploits	6675	2223
Generic	8044	3780
Reconnaissance	2097	730
Shellcode	206	69
Worms	25	8
Total	**35069**	**16466**

The records have been divided into two categories: regular and attacks. According to the nature of the attacks, the attacks are further divided into nine families.

The description of UNSW_NB15 features explained in appendix A.

Figure (4.4) below shows the class description of 20% of UNSW_NB15 training data with an imbalance of attacks classes such as Analysis, Backdoors, Shellcode and Worms, which has very low records in the training dataset.

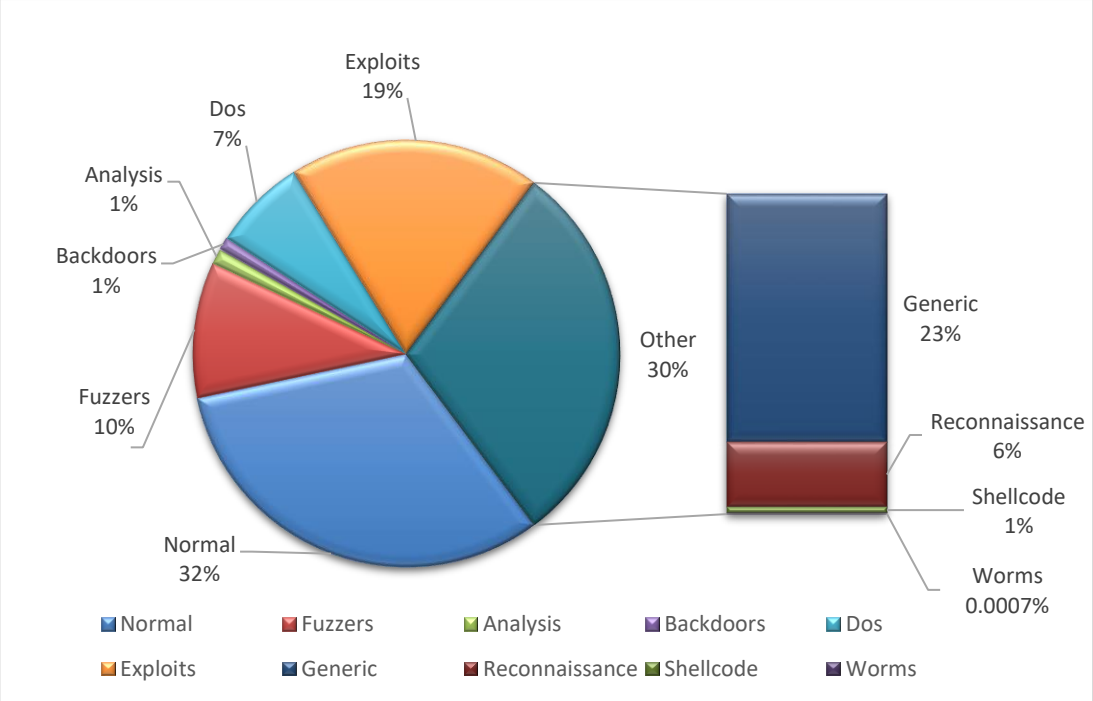


Figure (4.4): Class Distribution Of 20% UNSW_NB15_Training Data Set.

4.3 Data Transformation

This section describes the preprocessing phase, which converts data from nominal to numeric of the NSL_KDD and UNSW_NB15 data sets. Finally, data normalization have been implemented to remove bias from the data without changing their statistical properties.

4.3.1 Data Transformation based on One-Hot Encoding

The One-Hot Encoding method is applied to NSL_KDD and UNSW_NB15 data sets to transform nominal data into numeric data. In the case of the NSL_KDD data set, which has 38 numeric attributes and three nominal attributes. Table (4.4) below shows the nominal data of the NSL_KDD data set.

Table (4.4): The Nominal Data Of NSL_KDD Data Set.

No	Attributes name	description	type
2	protocol type	The type of protocol which used for connection	nominal

		(icmp, udp, tcp)	
3	Service	Network service on the destination (other, link, netbios_ssn, smtp, netstat, ctf, ntp_u, harvest, efs, klogin, systat, exec, nntp, pop_3, printer, vmnet, netbios_ns, urh_i, ssh, http_8001, iso_tsap, aol, sql_net, shell, supdup, auth, whois, discard, sunrpc, urp_i, Rje, ftp, daytime, domain_u, pm_dump,time, hostnames, name, ecr_i, bgp, telnet, domain, ftp_data, nnsp, courier, finger, uucp_path, X11, imap4, mtp, login, tftp_u, kshell, private, http_2784,echo, http, idap, tim_i, netbios_dgm, uucp, eco_i, Remote_job, IRC, http_443, red_i, Z39_50, Pop_2, gopher, Csnet_ns)	nominal
4	"flag"	normal or error status of the connection (OTH, S1, S2, RSTO, RSTRs, RSTOS0, SF, HS, REJ, S0, S3)	nominal

The "protocol type" attributes have three discrete values, the "service" attributes have 70 discrete values, and the "flag" attributes have 11 discrete values. In short, the one-hot encoding produces a vector with a length equal to the number of categories in the data attributes. However, the 41 attributes in the NSL_KDD data set will be converted to 122 numeric attributes.

The UNSW_NB15 data sets contain 39 numeric attributes and three nominal attributes. The "protocol type" attributes have 133 discrete values, the "service" attributes have 13 discrete values, and the "states" attributes have 11 discrete values. Table (4.5) shows the nominal data of the UNSW_NB15 data set.

Table (4.5): The Nominal Data of UNSW_NB15 Data Set.

No	Attributes name	description	Type
2	"protocol type"	Transaction protocol (3pc, a/n, aes-sp3-d, …)	nominal

3	"service"	Network service on the destination (dhcp, dns, ftp, …)	nominal
4	"states"	Indicates to the state and its dependent protocol (ACC, CLO, CON, …)	nominal

The UNSW_NB15 data set has 42 attributes encoded to 196 attributes. This technique overcame the difficulty of ranking because a binary vector represents each category. Both datasets have three categorical attributes, as mentioned earlier.

For example, the NSL_KDD data set has a nominal column called (protocol _type) with three attributes encoded using the one-hot-encoding method. Figure (4.5) illustrates the One-hot-encoding process of the (protocol_ type) attribute in the NSL_KDD data set. That means beyond the one-hot encoding simplicity, and its strength is to represent the discrete nature of categories.

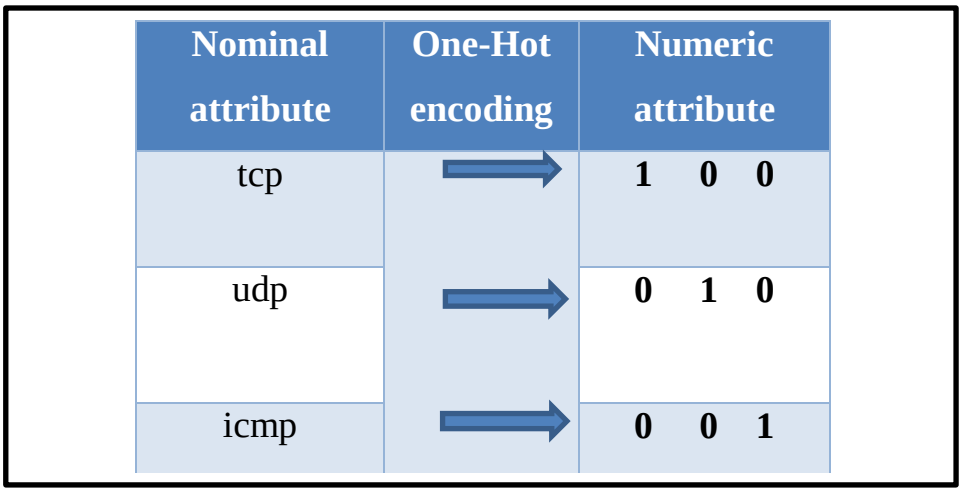


Figure (4.5): One-Hot-Encoding of (protocol_ type) Attribute in NSL_KDD Data Set

4.3.2 Data Normalization based on MIN-MAX

Normalization is a crucial preprocessing step in data mining that aims to standardize the values of all attributes or features across a wide dynamic range into a specified range. For proximity measures like Distance measure, which are sensitive to changes in the size or scales of the

attributes, normalization before clustering is critical. The idea is to equalize the size or magnitude of these attributes as well as their variability to ensure that all dimensions are treated similarly.

One of the critical reasons to use the Min-Max method because there are some attributes in the NSL-KDD dataset has a very large scope, such as [duration column] has scale [0, 57715], [src_bytes column] [0,1379963888] and [dst_bytes column] has [0,1309937401], Which are the difference between the maximum and minimum numbers are particularly large, the clustering process will be affected.

4.4 Feature Selection Analysis

To prove a precision of a proactive feature selection approach that shows in experimental evaluation of 118 features of (20 Percent Training Set) data file as training data with (NSL_ KDDTest-21) and (NSL_ KDDTest+) as testing data. Four machine learning methods have been used as the objective function (K-NN, SVM, NB, and DT). The compression criteria are given best (high) accuracy, precision, recall, and F1 score in this work. Both table (4.6) and table (4.7) abstract the experimental result of performance BA and PSO and their proactive BA and a proactive PSO.

Table (4.6): The Experiment With Feature Selection (20 Percent Training Set With NSL_KDDTest-21).

ML	Criteria Metrics	Wrapper Model			
		PSO	A -PSO	BA	A-BA
KNN	Accuracy	61.27	**63.02**	58.85	**61.96**
	Precision	68.92	65.51	68.72	72.04
	Recall	50.90	54.69	48.91	51.56
	F1-scor	58.51	58.14	57.15	60.02
SVM	Accuracy	52.34	**55.03**	51.74	**53.43**

	Precision	40.91	41.00	41.63	41.57
	Recall	43.68	45.54	43.27	44.11
	F1-scor	42.24	43.14	42.43	42.80
NB	Accuracy	48.22	**50.29**	45.50	**51.25**
	Precision	50.04	50.68	49.15	50.85
	Recall	43.57	45.51	42.38	45.47
	F1-scor	46.51	47.94	45.87	47.98
DT	Accuracy	63.64	**64.31**	62.22	**64.42**
	Precision	70.75	69.35	67.43	72.13
	Recall	53.69	54.06	52.14	53.66
	F1-scor	60.99	60.64	58.74	61.50

Table (4.7): The Experiment With Feature Selection (20 Percent Training Set With NSL_KDDTest+).

ML	Criteria Metrics	Wrapper Model			
		PSO	A-PSO	BA	A -BA
KNN	Accuracy	79.65	**79.74**	78.88	**79.51**
	Precision	78.65	70.23	77.99	77.56
	Recall	58.07	57.07	54.89	55.89
	F1-scor	66.81	62.89	64.29	64.96
SVM	Accuracy	74.46	**76.52**	74.76	**76.06**
	Precision	46.00	48.26	49.61	50.61
	Recall	47.87	49.13	48.49	47.32
	F1-scor	46.91	48.65	49.05	48.91
NB	Accuracy	51.97	**54.67**	50.50	**52.32**
	Precision	53.16	53.44	53.68	50.31
	Recall	40.95	44.10	40.05	43.84
	F1-scor	46.24	48.32	45.86	46.85
DT	Accuracy	80.65	**81.55**	79.86	**81.75**
	Precision	74.94	82.78	82.96	85.82
	Recall	57.16	60.86	57.73	61.06
	F1-scor	65.44	70.14	68.07	71.35

Based on a previous experimentations of a proactive PSO and BAT algorithms as feature selection, the a proactive BAT algorithm is used in proposed system as feature selection process as in suction (3.4) in chapter three with the help of the DT algorithm as an objective function. A small shift is used for each parameter in the BAT algorithm according to the impact order to give a more specific setting. The best parameter settings can be found such as Frequency is [0,2], max iteration is 10 , population size is 20 , loudness is 0.25 and pulse rate is 0.5.

In the case of the NSL-KDD data set, the experiment has shown the optimal number of selected features is 94 out of 122 for (20% from full NSL-KDD train+) with NSL_KDDTest-21 and NSL_KDDTest+. While in the UNSW_NB15 dataset, the optimal features are 158 out of 196 attributes. Table (4.8) shows the sequence and name in the NSL_KDD dataset. Table (4.9) presents each selected feature's sequence and name in the UNSW_NB15 dataset.

Table (4.8): The Optimal Features That Selected From NSL_KDD Dataset.

No	*Name of feature*	No	*Name of feature*	No	*Name of feature*
0	duration	**42**	Link	**80**	Bgp
2	dst_bytes	**44**	Smtp	**81**	telnet
3	land	**45**	Netstat	**83**	ftp_data
4	wrong_fragment	**46**	Ctf	**84**	Nnsp
5	urgent	**47**	ntp_u	**85**	courier
6	hot	**48**	Harvest	**86**	Finger
9	num_compromised	**49**	Efs	**87**	uucp_path
10	root_shell	**50**	Klogin	**88**	X11
12	num_root	**51**	Systat	**90**	Mtp
14	num_shells	**53**	nntp	**91**	Login
15	num_access_files	**54**	pop_3	**92**	tftp_u
17	is_hot_login	**56**	Vmnet	**93**	Kshell
18	is_guest_login	**57**	netbios_ns	**96**	Echo

19	count	**58**	urh_i	**97**	http
20	srv_count	**59**	Ssh	**98**	Idap
21	serror_rate	**60**	http_8001	**102**	eco_i
23	rerror_rate	**61**	iso_tsap	**103**	Remote_job
24	srv_rerror_rate	**62**	Aol	**105**	http_443
25	same_srv_rate	**65**	Supdup	**109**	gopher
26	diff_srv_rate	**66**	Auth	**110**	Csnet_ns
27	srv_diff_host_rate	**67**	Whois	**111**	OTH
28	dst_host_count	**69**	Sunrpc	**112**	S1
31	dst_host_diff_srv_rate	**70**	urp_i	**113**	S2
32	dst_host_same_src_port_rate	**71**	Rje	**114**	RSTO
34	dst_host_serror_rate	**72**	ftp	**116**	RSTOS0
35	dst_host_srv_serror_rate	**73**	Daytime	**117**	SF
36	dst_host_rerror_rate	**74**	domain_u	**118**	HS
37	dst_host_srv_rerror_rate	**75**	pm_dump	**119**	REJ
38	Icmp	**76**	Time	**120**	S0
39	Udp	**77**	Hostnames	**121**	S3
40	Tcp	**78**	Name		
41	Other	**79**	ecr_i		

Table (4.9): The Optimal Features That Selected From UNSW_NB15 Data Set.

No	*Name of feature*	No	*Name of feature*	No	*Name of feature*	*No*	*Name of feature*
0	dur	52	Mtp	102	Scps	150	bbn-rcc
1	sbytes	53	pri-enc	103	etherip	151	egp
2	dbytes	54	sat-mon	104	Aris	152	emcon
3	sttl	55	Cphb	106	compaq-peer	153	igp
4	dttl	56	sun-nd	107	Vrrp	154	nvp
5	sloss	58	Xtp	108	Iatp	156	xnet
6	dloss	59	Il	109	Stp	157	chaos
7	Sload	60	Unas	111	Srp	158	mux
9	Spkts	61	mfe-nsp	112	Sm	160	hmp
10	Dpkts	62	3pc	113	Isis	161	prm

11	swin	63	ipv6-route	115	Fire	162	trunk-1
12	dwin	64	Idrp	117	Crtp	163	xns-idp
13	stcpb	65	Bna	118	Sps	164	leaf-1
14	dtcpb	66	Swipe	119	merit-inp	165	leaf-2
16	dmeansz	67	kryptolan	120	Idpr	167	irtp
17	trans_depth	68	Cpnx	121	Skip	168	iso-tp4
18	res_bdy_len	69	Rsvp	122	Any	170	trunk-2
19	Sjit	71	Vmtp	123	Larp	171	cbt
22	Dintpkt	75	ax.25	124	Ipip	172	dhcp
23	tcprtt	76	Gmtp	125	Micp	173	dns
24	synack	79	Pgm	127	Ifmp	174	ftp
28	ct_flw_http_mthd	80	idpr-cmtp	128	tp++	175	ftp-data
29	is_ftp_login	81	Zero	129	a/n	176	http
31	ct_srv_src	82	Rvd	131	i-nlsp	177	irc
35	ct_src_dport_ltm	83	Mobile	132	ipx-n-ip	178	NON
36	ct_dst_sport_ltm	84	Narp	133	Sdrp	179	pop3
37	ct_dst_src_ltm	85	Fc	134	Tlsp	180	radius
38	attack_cat	86	Pipe	135	Gre	181	smtp
39	tcp	87	ipcomp	136	Mhrp	182	snmp
40	udp	88	ipv6-no	137	Ddx	184	ssl
42	ospf	89	sat-expak	138	Ippc	186	CLO
43	icmp	90	ipv6-opts	139	Visa	188	ECO
44	igmp	91	Snp	140	secure-vmtp	189	FIN
45	rtp	92	Ipcv	142	Vines	190	INT
46	ddp	93	br-sat-mon	144	Iplt	191	no
47	ipv6-frag	94	Ttp	145	Ggp	192	PAR
48	cftp	96	nsfnet-igp	146	Ip	193	REQ
49	wsn	97	sprite-rpc	147	Ipnip	195	URN
50	pvp	98	aes-sp3-d	148	st2		
51	wb-expak	99	sccopmce	149	argus		

4.5 Combination Techniques of NIDS Analysis

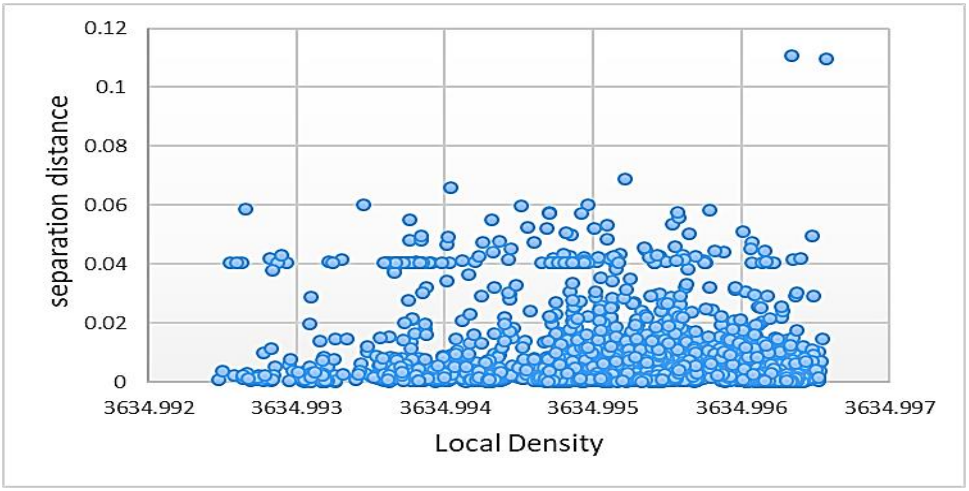
A Network Intrusion Detection System for attacks detection has been implemented based on combination techniques using IDPC clustering and ANN algorithms with the help of fractal fuzzy membership function using two Datasets: the first experiment implements 20% of the total NSL_KDD training file and NSL_KDDTest-21, NSL_KDDTest+ for testing data. The second experiment implements the UNSW_NB15 dataset with 20% of UNSW_NB15 training set and 20% of UNSW_NB15 testing set.

4.5.1 IDPC Clustering Analysis in a Training Stage

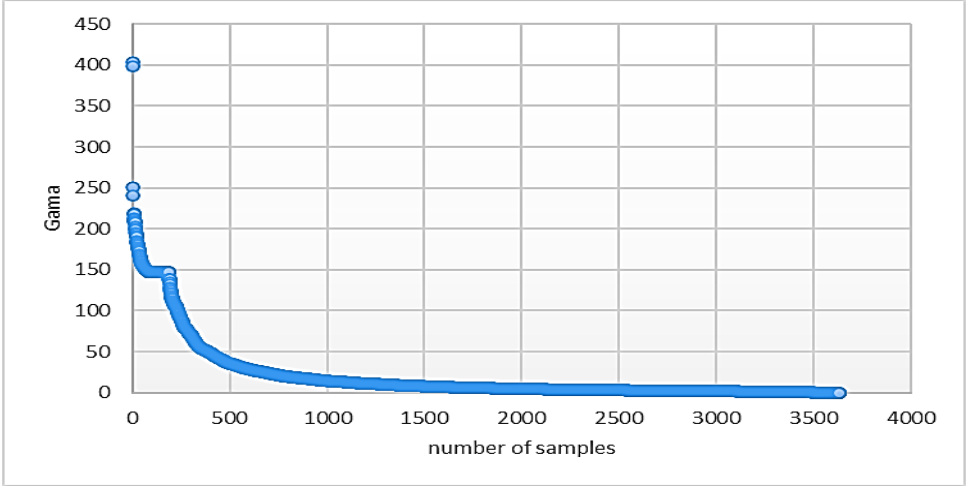
The core idea of the DPC is based on the Euclidean distance metric. This metric is responsible for calculating the distance between all points in the data set. Then after declaring peak points, this metric will distribute the non-peak data (the rest of train data) to the nearest peak and distribute the test data points to the nearest cluster. However, Euclidian distance does not capture the quality of the clustering making it an unsuitable metric to apply for maximizing both the homogeneity within each cluster and the heterogeneity between different clusters in case of complex data.

Since the proposed system is designed to be a proactive model, it must create clusters of forgiven datasets from an arbitrary distribution. However, investigate a sample of the NSL_KDD dataset using DPC clustering based on two distance metrics.

The experiment below used two distance metrics with the DPC algorithm as equations (2.8) and (2.9) in chapter two. Figure (4.6) shows the decision graph and sorting graph for DPC based on Gaussian-kernel distance of highest density peak and highest separation distance of NSL_KDD training data with two peaks points to be the centroid of the clusters. This way declare the peaks of the clusters are very clear and useful.



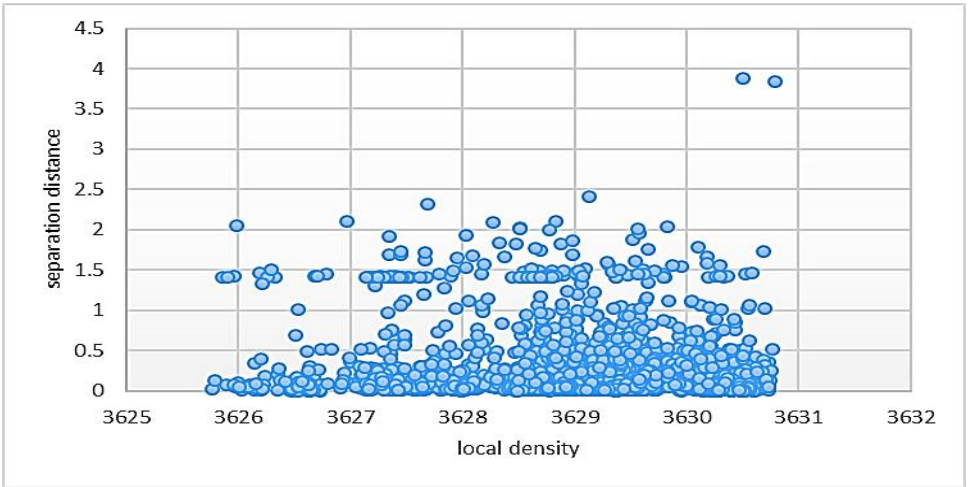
(A) Decision Graph



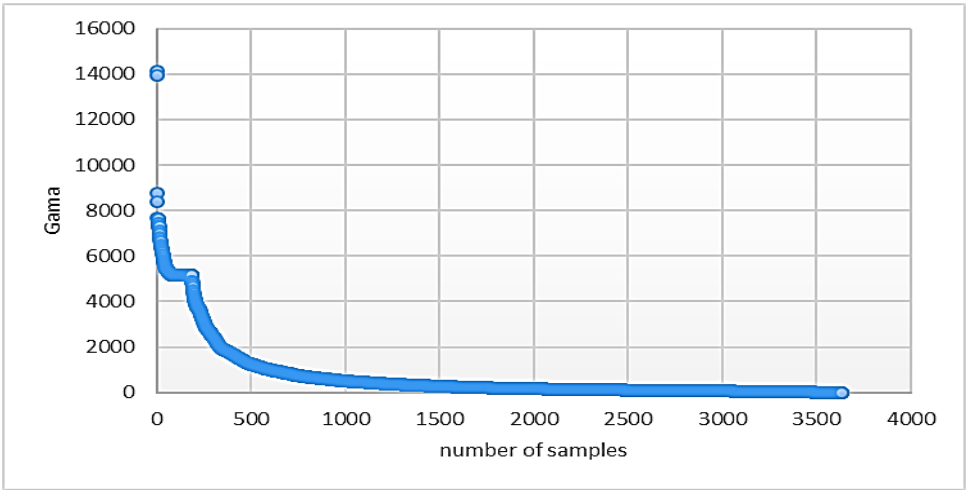
(B)γ Sorting Graph

Figure (4.6): Decision And γ Sorting Graphs For DPC Using Gaussian-Kernel

Figure (4.7) shows the Decision graph and sorting graph for DPC based on Euclidean distance of highest density peak and highest separation distance of NSL_KDD training data, giving two peaks points to be the centroid of the clusters. Euclidean distance declares the peaks of the clusters very clear, but the drawback, at high dimensions, Euclidean distance loses almost all meaning.



(A) Decision Graph



(B) γ Sorting Graph

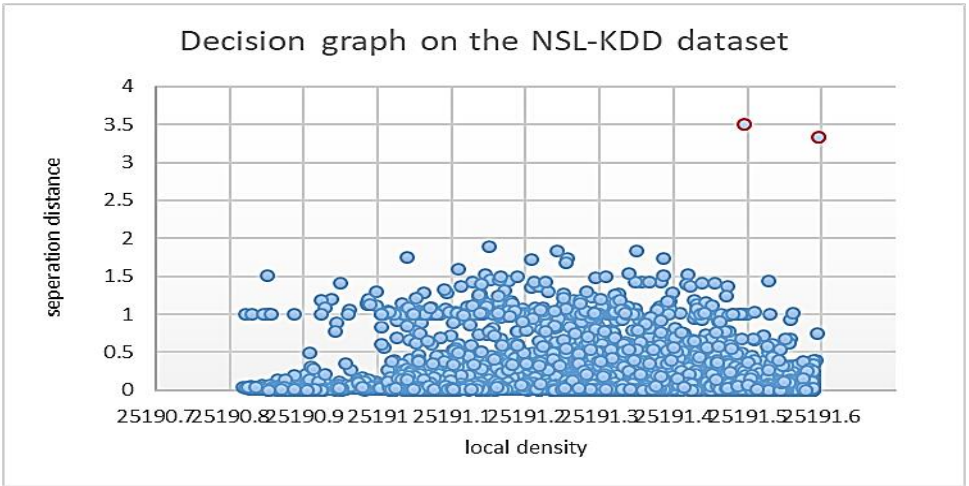
Figure (4.7): Decision And γ Sorting Graphs For DPC Using Euclidean

Table (4.10) shows the evaluation of using the density peak clustering (DPC) algorithm. However, the DPC algorithm using Gaussian kernel distance shows good clustering distribution results based on silhouette metric.

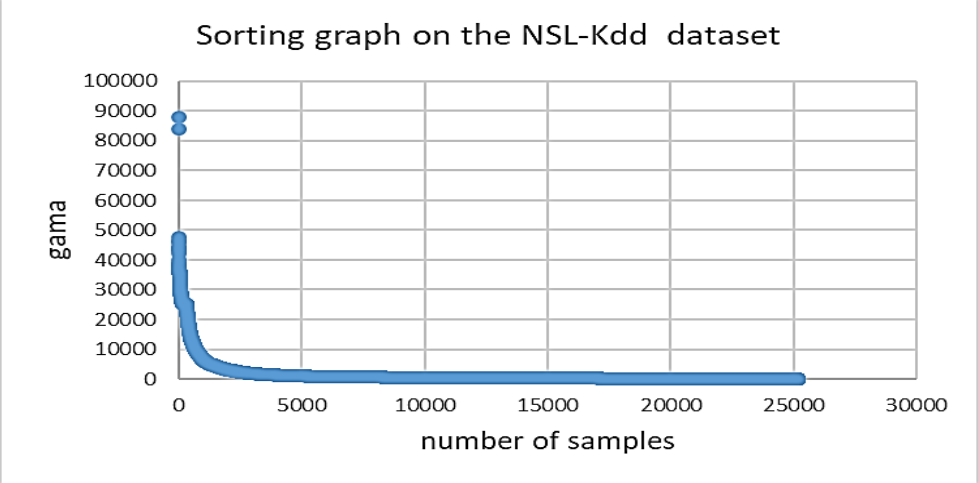
Table (4.10): Clustering Based On Distance Metric Of NSL-KDD Dataset.

Model	Train data	Test data	Distance metric	Peaks No	Silhouette
DPC	3635	1557	Euclidean	2	0.2752
DPC	3635	1557	Gaussian	2	**0.3905**

In the proposed system, the next step after a feature selection phase is using IDPC algorithm based on Gaussian kernel distance to dynamically divide the training dataset into two clusters using a decision and sorting graphs based on feature similarity to break the training dataset's imbalance and improve the detection rate of low rate classes. The process of determining the peak points is based on plots of all data points in a decision graph and sorting graph, as shown in figures (4.8) for the NSL_KDD data set and (4.9) for the UNSW_NB15 data set.

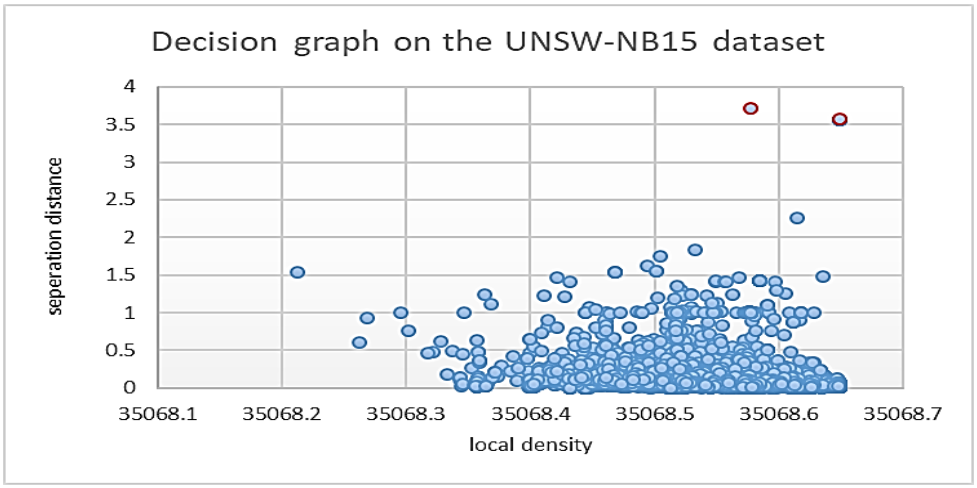


A. Decision Graph For NSL_KDD Train Data.

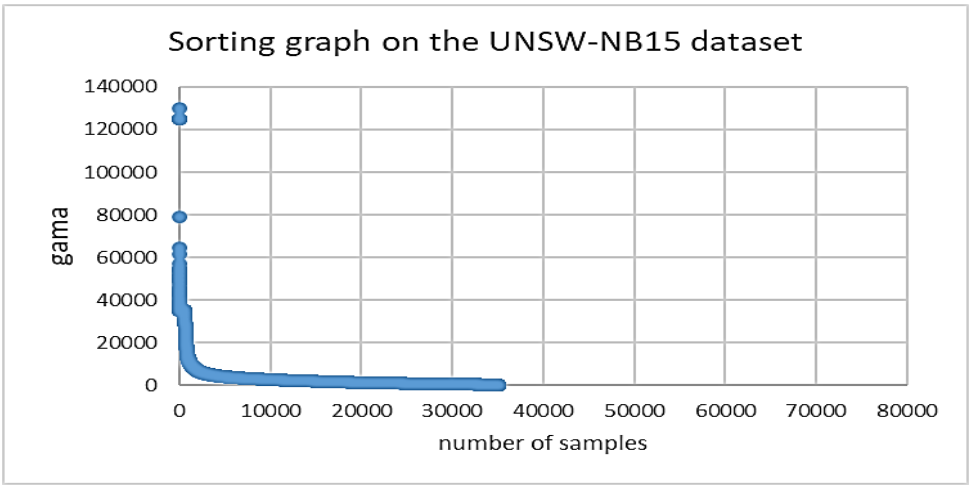


B. Sorting Graph for NSL_KDD Train Data.

Figure (4.8): Decision And Sorting Graphs Based On NSL_KDD Train Data.



A. Decision Graph Based On UNSW_NB15 Data Set With 20% Training



B. Sorting Graph Based On UNSW_NB15 Data Set With 20% Training

Figure (4.9): Decision And Sorting Graph Based On UNSW_NB15 Data Set With 20% Training.

Using a proactive feature selection technique will enhance the clustering process by removing the unreasonable features that seriously affect the detection performance. The training data will distribute for each peak according to the most feature similarity, which breaks the imbalance of train data. Experiments are carried out to evaluate IDPC's performance based on a proactive feature selection for both datasets, which enhance detection performance and helps to increase the homogeneity of training data distribution of the clustering process. Such as table (4.11).

Table (4.11): Evaluate The Performance Of IDPC Based On A Proactive Bat Algorithm.

Dataset	Feature selection	Train			test	silhoute
		Cluster1	Cluster2	total		
20% training NSL_KDD with NSL_KDD Test+	No	7896	17296	25192	22544	0.13840
	Yes	17467	7725		22544	**0.29658**
20% training NSL_KDD with NSL_KDDTest-21	No	7896	17296	25192	11850	0.09869
	Yes	17467	7725		11850	**0.19125**
20% Training UNSW_NB15 with 20% Testing UNSW_NB15	No	15930	19137	35069	16466	0.3549
	Yes	19826	15243		16466	**0.4740**

Figure (4.10) shows two Clusters of the NSL_KDD training dataset with all features. In comparison, Figure (4.11) shows two Clusters of the description of the NSL_ KDD training dataset with 94 features that enhance the distribution of normal, R2l and U2r classes.

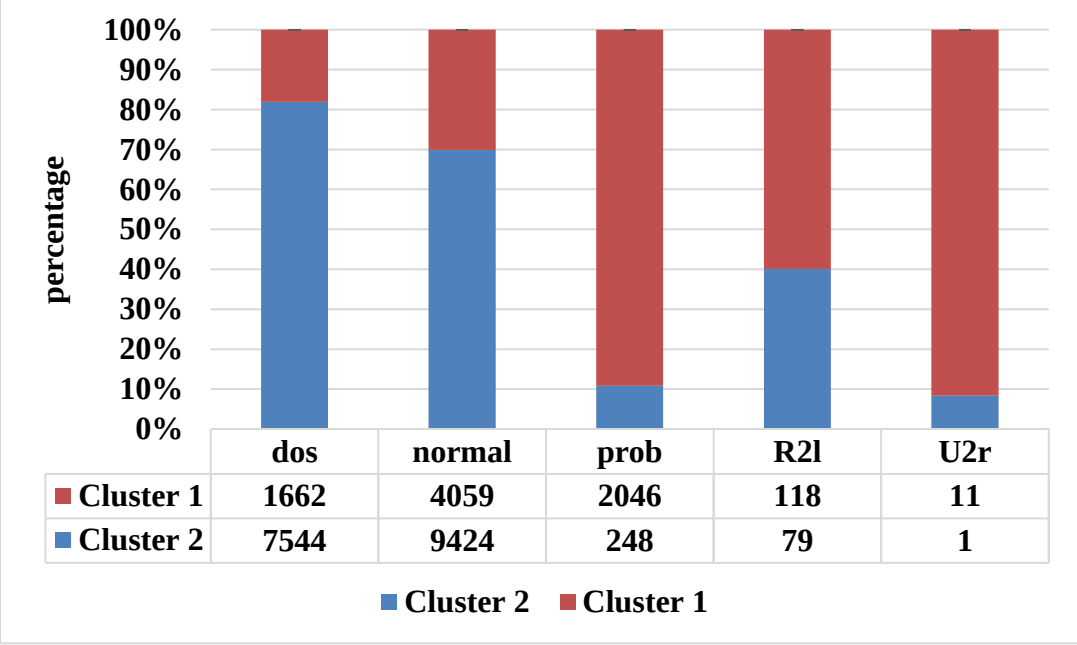


Figure (4.10): Clusters Description of NSL_KDD Data Set With Full Features.

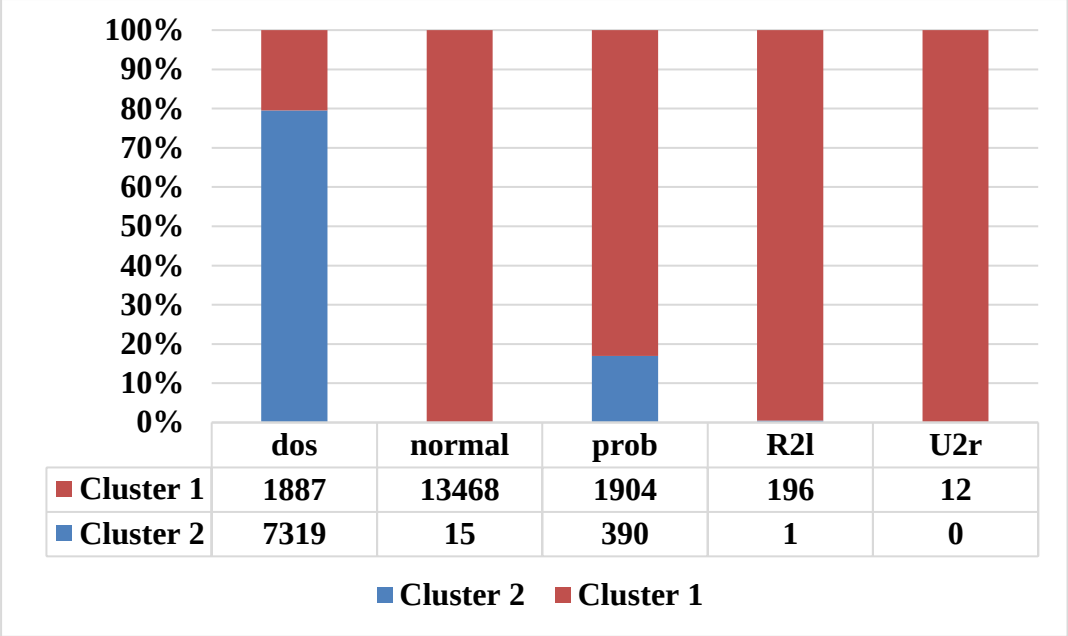


Figure (4.11): Clusters Description of NSL_KDD Data Set With 94 Features.

Figure (4.12) shows two Clusters descriptions of the UNSW_NB15 training dataset with all features. In comparison, Figure (4.13) shows two Clusters descriptions of the UNSW_NB15 training dataset for 158 features enhancing the distribution of training data for most classes, especially the Analysis, Backdoors and Fuzzers classes.

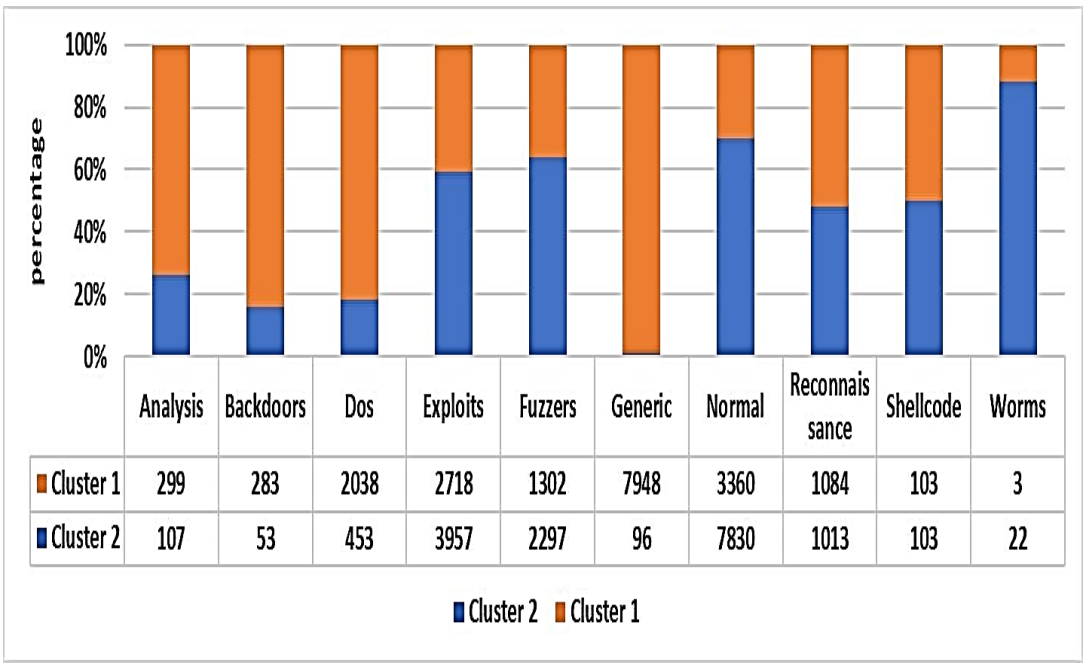


Figure (4.12): Clusters Description of UNSW_NB15 Data Set With Full Features.

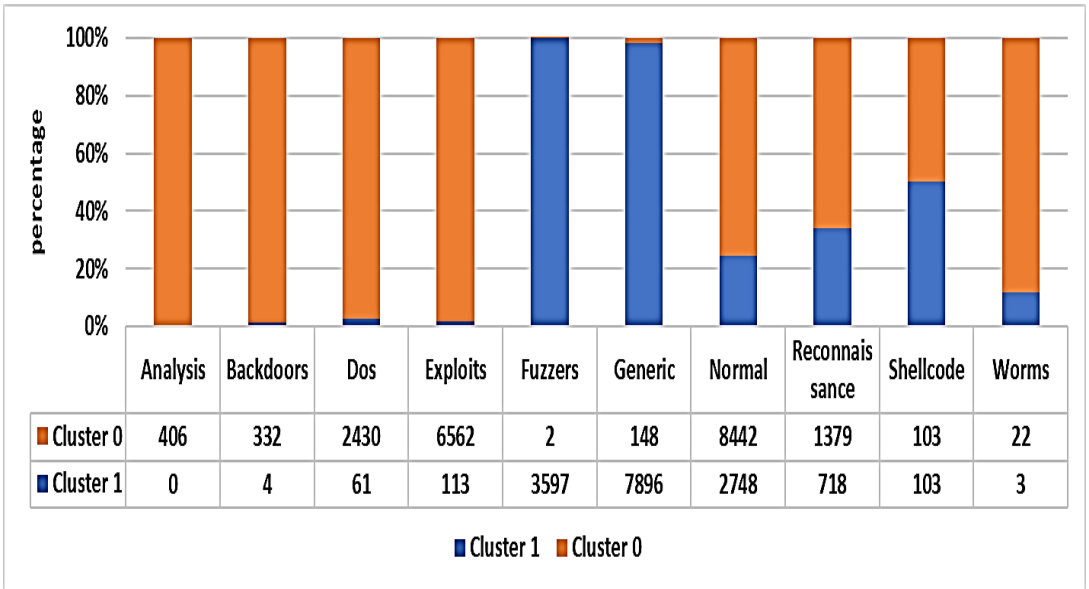


Figure (4.13): Clusters Description of UNSW_NB15 Data Set With 158 Features.

After assigning each point in the training data to the most appropriate cluster chosen in the previous steps, the next step is to calculate the fractal fuzzy membership matrix.

In general, the system searches for the nearest point to test point (x_j) in each cluster (c_j). This operation required to visit each point in the cluster i and examine them to select the nearest point. Therefore, it consumes a large amount of time and computational complexity. So, we proposed gathering data with the same approximation behavior of the test point in one sub-cluster of the cluster i .

The novel fractal fuzzy membership matrix (FFM) is proposed for this purpose. It is technically based on the fractal phenomenon, rather than calculating the distance between each test sample and all the training data points in each cluster, which is computationally complex and time-consuming. To solve this problem, we apply the fractal factor (f) between the test sample x_j and each point in clusters x_i. The delta-F determines the similarity ratio between the fractal of (x_j) test and fractal points in the cluster (c_j). The difference in time required to calculate the fractal fuzzy

membership function declared in the first experimentation that shows in the table (4.12) using (5192) records from the (*20 Percent Training Set of NSL_KDD)* to evaluate the best (delta-F) parameter to reduce search space and time needed. The proposed fractal metric minimizes the time required for building the fuzzy membership matrix.

Table (4.12): The Experimentation Using (20 Percent Training Set Of NSL_KDD) To Evaluate Best (Delta-F) Parameter.

	dataset	Training data	Testing data	Delta F	Silhoute	*FFU* Time(sc)
DPC	NSL_KDD	3635	1557	NaN	0.3905	325.46
DPC-F	NSL_KDD	3635	1557	0.8	0.3905	102.60
DPC-F	NSL_KDD	3635	1557	0.6	0.3905	99.86
DPC-F	NSL_KDD	3635	1557	0.4	0.3905	97.34
DPC-F	NSL_KDD	3635	1557	**0.2**	**0.3908**	**75.46**

The different values of the proposed fractal metric are tested. The weights used to test delta-F are 0.2, 0.4, 0.6, and 0.8. The optimal value is 0.2. Figure (4.14) shows the effect value of delta-F on time required to build the fuzzy membership matrix in the dataset (*20 Percent Training Set of NSL_KDD)*.

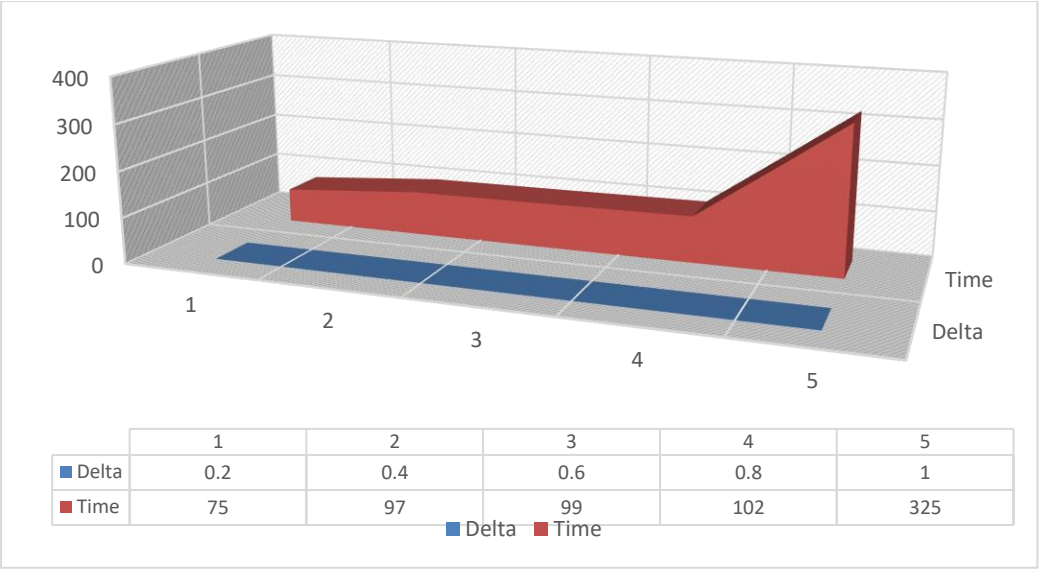


Figure (4.14): Delta-F Based On Time Required To Building The Fuzzy Membership Matrix.

The second experimentation that shows in the table (4.13) using (25192) records as the (20% from full NSL_KDDTrain+) as training data and (NSL_Test-21),(NSL_ Test+)as testing data to approve how the proposed fractal metric minimizes the time required for building the fuzzy membership matrix.

Table (4.13): The Comparison Time Between Building The Fuzzy Membership Matrix With Fractal Metrics And Without Fractal.

	Dataset	Train+Test			Delta F	Silhoute	*FFU*
		Train	Test	total			Time(sc)
ABA-IDPC	20% from full NSL_KDD training with NSL_ Test+	25192	22544	47736	NON	0.2966	5049.35
		25192	22544		0.4	0.2965	4792.45
		25192	22544		0.2	**0.2974**	**3361.42**
	20% from full NSL_KDD training with NSL_Test-21	25192	11850	37042	NON	0.1912	3113.96
		25192	11850		0.4	0.1912	2568.18
		25192	11850		0.2	**0.1926**	**1798.65**

4.5.2 ANN Classifiers Analysis in a Training Stage for both Data Sets

Each cluster will be trained on a separate ANN classifier. The structure for both NSL_KDD and UNSW-NB15 data sets are:

- The number of the hidden layers are two,
- The optimizer of each ANN is Adam,
- ReLU is the activation function that is used of the hidden layer,
- softmax is the activation function that used for the output layer.

The number of neurons for hidden layers in the NSL_KDD data set is [80, 10,], while the number of neurons for hidden layers in the UNSW-N15 data set is [120, 20,]. The learning rate and the number of neurons are two crucial hyper-parameters that must be established. If the learning rate is excessively high during the training phase, the network will

oscillate, resulting in non-convergence. Convergence will be slow if the learning rate is too low. The Adam optimizer's default learning-rate in the sklearn is (0.001); hence, candidate's learning rate is **(0.1, 0.001, 0.002, 0.003, 0.004, 0.005, and 0.0001)** such as figure (4.15) and figure (4.16) below.

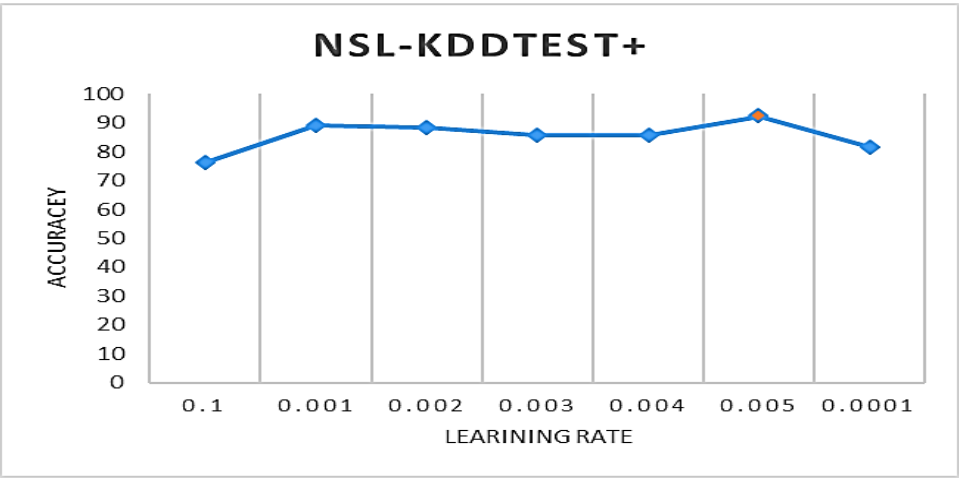


Figure (4.15): Comparison Accuracy With Different Learning Rates Based On NSL_KDD Data Set.

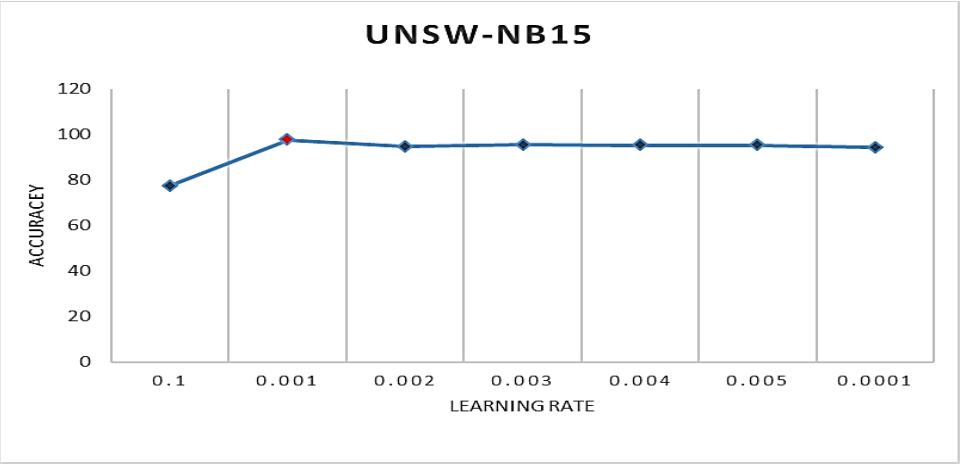


Figure (4.16): Comparison Accuracy With Different Learning Rates Based On UNSW_NB15 Data Set.

The experimentations show that a reasonable learning rate for NSL_KDD is 0.005 while 0.001 for the UNSW_NB15 dataset. The second hyper-parameter is the number of neurons evaluated as figure (4.17) and figure (4.18) for both datasets. Few neurons could produce

under-fitting because the network may not learn properly. Many neurons could make overfitting because the network learns too much from training data and does not generalize. Therefore, there must be an intermediate number of neurons that ensures good training.

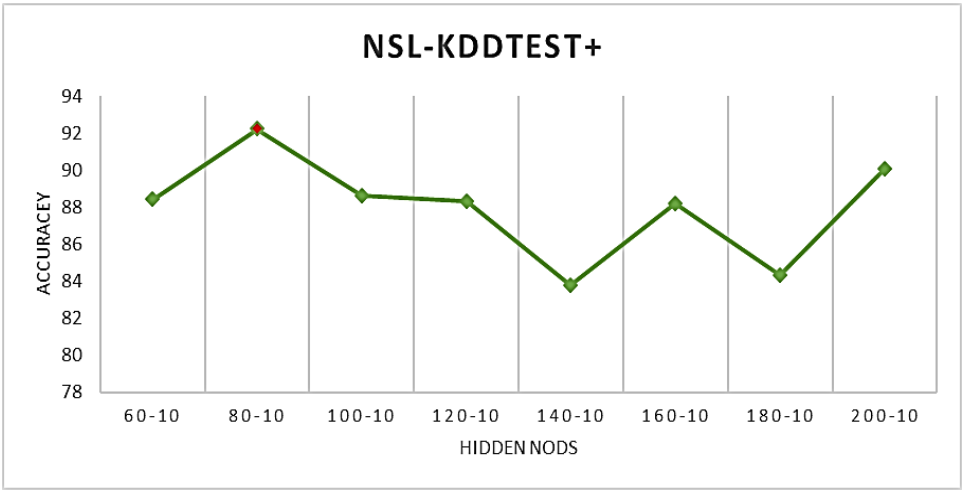


Figure (4.17): Comparison Accuracy With Different Hidden Neurons Based On NSL_KDD Data Set.

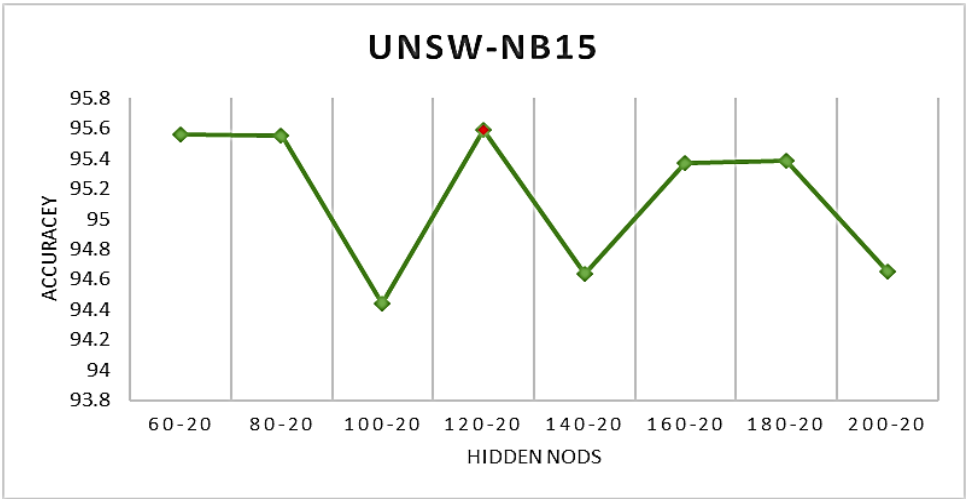


Figure (4.18): Comparison Accuracy With Different Hidden Neurons Based On UNSW_NB15 Data Set.

4.5.3 ANN Classifiers Analysis in a Testing Stage for Both Data Sets

In *a testing stage*, with the help of fractal fuzzy membership degree, the predicted values of each sub-ANNi algorithm will aggregate depending on the aggregation method. The test sample xj in each sub-

ANNi classifier is known as *ANNi(xj)*. However, the predicted values of each test sample will be aggregated using the algorithm (3.9) in chapter three, which is used to detect attack categories. Then, a confusion matrix for testing data points is created using the confusion Matrix algorithm (3.10).

4.5.3.1 ANN Analysis in a Testing Stage for NSL_KDD Data Set

The NSL_KDD Data set was used to test the performance of the proposed system. Table (4.14) show the Confusion Matrix result for NSL_KDDTest-21 Datasets in a Testing Phase.

Table (4.14): Confusion Matrix Results For Five Classes For NSL_Kddtest-21 Data Sets In A Testing Phase.

		Predicted Class				
		Dos	**Normal**	**Probe**	**R2l**	**U2r**
Actual Class	**Dos**	**3468**	213	593	68	0
	Normal	6	**2131**	15	0	0
	Probe	226	60	**2021**	95	0
	R2l	455	144	67	**2088**	0
	U2r	0	131	4	0	**65**

Table (4.15) presents the binary confusion matrix result for NSL_KDDTest-21 from the table (4.14). The accuracy detection was **95.2%,** the detection rate was **94.3%**, and the misclassification rate was **0.05%**.

Table (4.15): Binary Confusion Matrix Results Using NSL_KDDtest-21 Datasets In A Testing Phase

		Normal	Attack
	Normal	TN (**2131**)	FN (548)
	Attack	FP (21)	TP (9150)

Table (4.16) show the Confusion Matrix result for NSL_KDDTest+ Data sets in a Testing Phase. The multi-class confusion matrix for both test data has five classes, one class for the normal transaction and four classes for malicious detection (Dos, Probe, R2l, U2r) attacks.

Table (4.16): Confusion Matrix Results For Five Classes For NSL_KDDTest+ Data Sets In A Testing Phase.

		Predicted Class				
		Dos	**Normal**	**Probe**	**R2l**	**U2r**
Actual Class	**Dos**	**6494**	209	681	74	0
	Normal	6	**9688**	16	1	0
	Probe	166	56	**2104**	95	0
	R2l	118	148	36	**2452**	0
	U2r	0	143	0	0	**57**

Table (4.17) presents the binary confusion matrix result for NSL_KDDTest+ from the table (4.16). The accuracy detection was 97.4%, the detection rate was 95.6%, and the misclassification rate was 0.02%.

Table (4.17): Binary Confusion Matrix Results Using NSL_KDDTest+ Data Sets In A Testing Phase.

		Normal	Attack
	Normal	TN (**9688**)	FN (556)
	Attack	FP (23)	TP (12277)

4.5.3.2 ANN Analysis in a Testing Stage for UNSW_ND15 Data Set

The UNSW_ND15 Data set was used to test the proposed system performance. Table (4.18) show the multi-class Confusion Matrix result for UNSW_NB15 Data set in a Testing Phase using the confusion matrix for ten classes, one class for the regular transaction, and the other nine

classes for malicious detection (Generic , Exploits , Fuzzers , Dos , Reconnaissance , Analysis , Backdoor , Shellcode , Worms).

Table (4.18): Confusion Matrix Results For Ten Classes For UNSW_NB15 Data Set In A Testing Phase.

		Predicted Class									
		Analysis	Backdoor	Dos	Exploits	Fuzzers	Generic	Reconnaissance	Shellcode	Worms	Normal
	Analysis	**136**	0	0	0	0	0	19	0	0	0
	Backdoor	0	**110**	1	0	0	0	3	0	0	0
	Dos	0	6	**826**	0	0	0	1	0	0	0
	Exploits	0	0	0	**2051**	0	0	0	163	0	9
	Fuzzers	0	0	0	485	**710**	0	0	0	0	10
	Generic	0	0	0	0	0	**3780**	0	0	0	7
	Reconnaissance	0	0	0	22	0	0	**708**	0	0	0
	Shellcode	0	0	0	0	0	0	0	**65**	0	4
	Worms	1	0	0	1	0	0	1	0	**5**	0
	Normal	0	0	0	0	0	0	0	0	0	**7349**

Table (4.19) presents the binary confusion matrix result for UNSW_NB15 from table (4.18). The accuracy detection was 99.8%, the detection rate was 99.7%, and the misclassification rate was 0%.

Table (4.19): Binary Confusion Matrix Results Using UNSW_NB15 Data Sets In A Testing Phase.

		Normal	Attack
	Normal	TN (**7349**)	FN (23)
	Attack	FP (0)	TP (9094)

4.6 Results Comparison

It is clearly shown that the result of the proposed system in Table (4.20) presents the overall performance based on accuracy, Precision, Recall and f1scor for the system for three testing data sets.

Table (4.20): The Accuracy (%) For The proposed System Based On Three Data Sets.

Dataset	Accuracy	Precision	Recall	f1scor	Time (sec)
NSL_KDDTest-21	**82.47**	83.44	82.47	82.95	0.1504
NSL_KDDTest+	**92.24**	92.68	92.24	92.46	0.3103
UNSW_NB15	**95.59**	96.68	95.59	96.13	0.4537

The experimented on these data sets shows how well the proposed system performed. The results compared to five established machine-learning classifiers such as K-Nearest Neighbor (KNN), Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT) and Artificial neural network (ANN). The Comparison Results are evaluated based on the four evaluation metrics (Accuracy, Precision, Recall, f1scor). The results are depicted in tables (4.21), (4.22) and (4.23).

Table (4.21): Results Comparison Of Different Algorithms Using (NSL_KDDTest-21) Data Set.

Model	Dos	Normal	Probe	R2l	U2r	Accuracy	Precision	Recall	f1scor
KNN	71.2	67.6	64.2	11.7	4.0	54.21	71.47	54.21	61.66
RF	59.4	88.1	61.9	6.7	0.5	51.94	79.63	51.94	62.87
SVM	57.0	68.1	63.2	0	0	46.10	50.64	46.10	48.27
DT	57.8	68.9	62.1	22.4	9.5	51.69	65.06	51.69	57.61
ANN	71.1	82.9	76.4	13.2	4.5	59.81	75.92	59.81	66.91
Proposed system	**79.8**	**99.0**	**84.1**	**75.8**	**32.5**	**82.47**	**83.44**	**82.47**	**82.95**

As shown in table (4.21), the proposed system for the (NSL_KDDTest-21) dataset shows a high detection rate of low rate classes such as U2R and R2L attack types, which have detection accuracy of 32.5% and 75.8%, respectively. It is produces a higher accuracy of 82.47% than the other compared methods.

Table (4.22): Results Comparison Of Different Algorithms Using NSL_KDDTest+ Data Set.

Model	Dos	Normal	Probe	R2l	U2r	Accuracy	Precision	Recall	f1scor
KNN	82.9	92.4	64.5	11.7	4.0	75.68	78.32	75.68	76.98
RF	75.8	97.3	62.2	6.7	0.5	74.56	81.01	74.56	77.67
SVM	74.9	92.8	63.5	0.0	0.0	71.60	65.70	71.60	68.52
DT	75.3	93.0	62.4	22.4	9.5	74.53	75.38	74.53	74.95
ANN	83.2	96.1	76.6	13.2	4.5	78.83	81.03	78.83	79.92
Proposed system	**87.0**	**99.7**	**86.9**	**89.0**	**28.5**	**92.24**	**92.68**	**92.24**	**92.46**

As shown in table (4.22), the proposed system for the (NSL_KDDTest+) dataset shows a high detection rate of low rate classes such as U2R and R2L attack types, which has detection accuracy of 28.5% and 89.0%, respectively. It is produces a higher accuracy of 92.24% than the other compared methods.

Table (4.23): Results Comparison Of Different Algorithms Using UNSW_NB15 Data Set.

Class	KNN	RF	SVM	DT	ANN	Proposed system
Analysis	16.1	1.9	00.0	1.2	0.0	87.7
Backdoors	14.0	7.8	00.0	21.0	0.0	96.4
Dos	23.2	19.4	00.0	26.5	25.3	99.1
Exploits	65.9	80.0	76.3	67.8	79.2	92.2
Fuzzers	55.7	60.3	59.8	59.8	59.8	85.5
Generic	96.1	96.2	97.2	96.7	96.1	100
Normal	85.1	94.6	85.1	91.1	87.9	100
Reconnaissance	66.0	83.5	36.7	81.0	88.0	96.9
Shellcode	23.1	62.3	00.0	62.3	30.4	94.2

Worms	00.0	00.0	00.0	12.5	0.0	62.5
Accuracy	77.51	84.59	76.38	81.79	81.72	**95.59**
Precision	82.54	86.20	75.02	85.91	84.60	**96.68**
Recall	77.51	84.59	76.38	81.79	81.72	**95.59**
f1scor	79.95	85.39	75.69	83.80	83.13	**96.13**

As shown in table (4.23), the proposed system for the (UNSW_NB15) dataset shows a high detection rate of low rate classes such as Analysis, Backdoors, Shellcode, and Worms attack types, which have detection accuracy of 87.7%, 96.4%, 94.2% and 62.5%, respectively. It produces a higher accuracy of 95.59% than the other compared methods.

4.7 Comparison with the Related Works

To prove the case of handling the imbalance problem of multi classes by enhancing the detection rate of low rate classes such as R2l, U2r, Analysis, Backdoors, Worms, and Shellcode with better performance of our proposed system. The proposed system was compared to nine intrusion detection models.

On the data sets (NSL KDDTest-21), (NSL KDDTest+), and (UNSW NB15), the proposed system is outperforms based on the Accuracy, Precision, Recall, f1scor, and detection rate of low rate classes. The Comparison results based on these datasets are shown in tables (4.24), (4.25), and (4.26), respectively.

Table (4.24): Comparison Results With Different Models Based On (NSL_KDDTest-21) Data Set.

Author	Model	Train data	Test-21 data	Accuracy	D.R of R2l	D.R of U2r
T. Ma .et al [21] (2016)	SCDNN	25192	11850	44.55	1.50	0.98

C. Yin .et al [23] (2017)	RNN-IDS	125,973	11850	64.67	N/A	N/A
Y. Yang .et al [25] (2019)	MDPCA-DBN	25192	11850	66.18	34.93	6.00
Y. Yang.et al [25] (2019)	DBN	25192	11850	57.45	13.25	0.50
C. Tang .et al [26] (2020)	SAAE-DNN	125,973	11850	77.57	N/A	N/A
Proposed system	ABA-IDPF-ANN	25192	11850	**82.47**	**75.8**	**32.5**

Table (4.25): Comparison Results With Different Models Based On (NSL_KDDTest+) Data Set.

Author	Model	Train data	Test data	Accuracy	D.R of R2l	D.R of U2r
Y. Yang .et al [25] (2019)	MDPCA-DBN	25192	22544	82.08	17.25	6.50
C. Yin .et al [23] (2017)	RNN-IDS	125,973	22544	81.29	24.69	11.50
Y. Yang.et al [25] (2019)	DBN	25192	22544	80.82	12.56	5.50
C. Tang .et al [26] (2020)	SAAE-DNN	125,973	22544	82.14	N/A	N/A
Chao Liu. et al [28] (2021)	Kmeans+RF+CNN+LSM	125,973	22544	85.24	73.84	25.79
Isra .et al [29] (2021)	MCNN-DFS	125,973	22544	81.44	N/A	N/A
Proposed System	ABA-IDPF-ANN	25192	22544	**92.24**	**89.0**	**28.5**

Table (4.26): Comparison Results with Different Models Based On (UNSW_NB15) Data Set

Author	Model	Train data	Test data	Accuracy	D.R of Analysis	D.R of Backdoors	D.R of Shellcode	D.R of Worms
N.Moustafa .et al [22] (2016)	EM clustering	105204	32932	78.47	N/A	N/A	N/A	N/A
H. M. Anwer .et al [24] (2018)	RG filter FS- J48	175341	82332	88.30	N/A	N/A	N/A	N/A
Y. Yang .et al [25] (2019)	MDPCA-DBN	35,069	16,466	90.21	0.0	0.8	39.4	11.1
S. M. Kasongo.et al [27] (2020)	XGBoost-ANN	131,506	82,332	77.51	N/A	N/A	N/A	N/A
Proposed system	ABA-IDPF-ANN	35069	16466	**95.59**	**87.7**	**96.4**	**94.2**	**62.5**

All the experimentations results that compared above shows the proposed system not only has a strong modelling ability for network intrusion detection, but also has high accuracy in both low rate and multiclass classification.

Chapter Five

Conclusions and Future Works

Chapter Five
Conclusions and Future Works

5.1 Conclusions

Network intrusion detection has become a critical component of any defense system within commercial enterprises. Anomaly Network Intrusion Detection (ANID) and Misuse Network Intrusion Detection (MNID) are major network intrusion detection types. Approaches based on knowledge discovery and data mining (KDD) play an important role in identifying and eliminating malicious packets. In the literature on the topic, hybrid approaches for intrusion detection, especially anomaly detection, have been considered. The following are the essential characteristics that have been obtained from the results of this dissertation:

1. One-hot encoding is a popular strategy for converting network data packets from nominal to numerical attributes since it is simple. However, this widely used encoding scheme assumes a flat label space, avoiding the extensive relationships between labels that might be exploited during training, improving the detection rate of low rate attacks for both datasets.

2. The Feature Selection based on a Proactive Bat Optimization model (ABA) potentially enhances the search progress and initial value of the population pool of swarm optimization. This model was inspired by the diverse sources of generating new solutions to select relative features. It helps make a reasonable reduction of features for both datasets. It also helps increase the homogeneity of training data distribution of the clustering process. The number of selected

features for the NSL_KDD dataset are 94 out of 122 features, while the selected features in UNSW_NB15 are 158 out of 196 features.

3. The Improved DPC clustering Algorithm enables dealing with (attacks / regular) clusters of non-spherical shapes of different sizes by dividing the complex network dataset into two distinct groups, each group as far away from the other as possible. This case will solve the imbalance problem of low rate classes. However, this will efficiently help the ANN classifier increase the detection rate of low rate attacks.

4. The highly intertwined samples increase the probability that the sample xj may not be assigned to the optimal cluster Ck. The membership function declares the degree of the test relation to cluster pool, multiplied with the corresponding output of each ANN classifier in the output aggregation method to improve detection accuracy. As a result, build a fuzzy membership matrix FFM based on the fractal factor rather than calculating the distance between each test sample and all of the training data points in each cluster, which is computationally complex and time-consuming.

5. Experimental results have shown that the proposed system detects multiple types of attacks and achieves excellent results on extensive network data efficiently as shown in table(4.21) for (NSL_KDD, UNSW_NB15) datasets. Empirical results show that the Hybrid model is promising in terms of detection accuracy of multi-classification attacks and thus amenable for real-time network intrusion detection.

5.2 Suggestions for Future Works

In this dissertation, a proactive Hybrid technique based on cluster analysis and artificial neural network classification is implemented. The following are suggestions for future works:

1. Implement the training phase of the proposed system in an apache spark environment to reduce the time complexity of the training process.

2. Suggest a NIDS model for network traffic based on next-generation Internet Protocol (IPv6), which can scan for vulnerabilities in the semantics layer to select an attack signature most suited for misuse detection, and then use network behavior to detect anomalies.

3. Designing an efficient model for attacks detection based on stream data mining techniques.

References

REFERENCES

[1] E. Biersack, F. Measurement, and D. Hutchison, "Data Traffic Monitoring and Analysis". Springer.LNCS7754,2013

[2] A. Chopra, “Security Issues of Firewall,” International Journal of P2P Network Trends and Technology (IJPTT) – Volume 22 Number 1 January 2016

[3] Mehdi Ebady and Angela Amphawan, “R EVIEW OF SYN-FLOODING ATTACK DETECTION MECHANISM,” International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.1, January 2012

[4] H. Al-Rushdan, M. Shurman, S. H. Alnabelsi, and Q. Althebyan, “Zero-day attack detection and prevention in software-defined networks,” Proc. - 2019 Int. Arab Conf. Inf. Technol. ACIT 2019, no. December, pp. 278–282, 2019.

[5] J. Jang-Jaccard and S. Nepal, “A survey of emerging threats in cybersecurity,” J. Comput. Syst. Sci., vol. 80, no. 5, pp. 973–993, 2014.

[6] Mehdi Ebady and Wesam S.Bhaya, Doctor of Philosophy ,“Dynamic DDoS Attack Detection based on Data Mining Approach,” p. 138, 2015.

[7] B. Sunny, K. Krishan, and A. Vishal, “Classification of Flood Based DDoS Attacks,” Conference: WECON-2008.

[8] Q. Niyaz, W. Sun, A. Y. Javaid, and M. Alam, “A deep learning approach for network intrusion detection system,” EAI Int. Conf. Bio-inspired Inf. Commun. Technol., 2015.

[9] J. Garcia-alfaro, Doctor of Philosophy “Anomaly-Based Network Intrusion Detection Using Machine Learning.” NNT : 2020IPPAS011

[10] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, “Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model,” J. Comput. Sci., vol. 25, no. March, pp. 152–160, 2018.

[11] V. Jyothsna, V. V. Rama Prasad, and K. Munivara Prasad, “A Review of

Anomaly based Intrusion Detection Systems," Int. J. Comput. Appl., vol. 28, no. 7, pp. 26–35, 2011.

[12] Suad Abdulelah Abdulhussein and Wesam S. Bhaya, Doctor of Philosophy "Network Intrusion Detection using BFR Algorithm," 2018.

[13] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and," Electronics.2020.

[14] R. A. R. Ashfaq, X. Z. Wang, J. Z. Huang, H. Abbas, and Y. L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," Inf. Sci. (Ny)., vol. 378, pp. 484–497, 2017.

[15] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," Sensors (Switzerland), vol. 16, no. 10, 2016.

[16] 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," Inf. Secur. J., vol. 25, no. 1–3, pp. 18–31, 2016.

[17] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," IEEE Access, vol. 5, pp. 21954–21961, 2017.

[18] H. M. Anwer, M. Farouk, and A. Abdel-Hamid, "A framework for efficient network anomaly intrusion detection with features selection," 2018 9th Int. Conf. Inf. Commun. Syst. ICICS 2018, vol. 2018-Janua, pp. 157–162, 2018.

[19] 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," Appl. Sci., vol. 9, no. 2, 2019.

[20] I. Detection, "SS symmetry SAAE-DNN : Deep Learning Method on

Intrusion Detection," Symmetry,pp. 1–20, 2020.

[21] S. M. Kasongo and Y. Sun, "Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset," J. Big Data, vol. 7, no. 1, 2020.

[22] C. Liu, Z. Gu, and J. Wang, "A Hybrid Intrusion Detection System Based on Scalable K-Means+ Random Forest and Deep Learning," IEEE Access, vol. 9, pp. 75729–75740, 2021.

[23] I. Al-Turaiki and N. Altwaijry, "A Convolutional Neural Network for Improved Anomaly-Based Network Intrusion Detection," Big Data, vol. 9, no. 3, pp. 233–252, 2021.

[24] D. Coss, "THE CIA STRIKES BACK : REDEFINING CONFIDENTIALITY , INTEGRITY AND AVAILABILITY IN,", JISSec pp. 21–45, 2014.

[25] S. Qadir and S. M. K. Quadri, "Information Availability : An Insight into the Most Important Attribute of Information Security," no. April, pp. 185–194, 2016.

[26] P. Oscarson, "Information security fundamentals." 2003 - Springer pp 95–107.

[27] Bjo¨rn Lundgren and N. Mo, "Defining Information Security ¨," Sci Eng Ethics 25:419–441.2017.

[28] G. Kumar, "Denial of service attacks – an updated perspective,", Systems Science & Control Engineering, Taylor & Francis Group. vol. 2583, 2016.

[29] A. Abraham, U. States, C. Grosan, and Y. Chen, "Cyber Security and the Evolution of Intrusion Detection Systems," , i-Manager's Journal on Future Engineering and Technology ,no. May 2014, 2005.

[30] D. T. Britt and C. Matthews, "Front cover TCP / IP Tutorial and," redbooks, vol. 1, p. 38, 2006.

[31] N. N. Nazmi and W. I. Omi, "A Comparative Study of Transmission

Control Protocols,", IJERT, vol. 4, no. 06, pp. 434–439, 2015.

[32] A. S. Guide, S. Edition, and C. Press, Designing for Cisco Internetwork Solutions (DESGN). Cisco Press,2008 .

[33] S. Jonnalagadda, "Introduction to TCP / IP Protocol Suite Srinivas Jonnalagadda , Ph . D . Siri Technologies,2003.

[34] Craig Hunt, TCP/IP Network Administration. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.2003.

[35] Suhail Qadir, S. M. K. Quadri, "Information Availability: An Insight into the Most Important Attribute of Information Security" Journal of Information Security, 2016, 7, 185-194.

[36] W. A. H. M. Ghanem and A. Jantan, A new approach for intrusion detection system based on training multilayer perceptron by using enhanced Bat algorithm, vol. 2. Springer London, 2019.

[37] G. González-granadillo, S. González-zarzosa, and R. Diaz, "Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures," Sensors 2021, 21(14), 4759.

[38] Peddisetty Naga Raju, "State-of-the-art Intrusion Detection : Technologies , Challenges , and Evaluation," LiTH-ISY-EX-3586-2005.

[39] W. Bul, A. James, and M. Pannu, "Journal of Computer and System Sciences Improving network intrusion detection system performance through quality of service configuration and parallel technology," vol. 81, pp. 981–999, 2015.

[40] N. M. Jacob and M. Y. Wanjala, "A Review of Intrusion Detection Systems," Global Journal of Computer Science and Technology:, 2017.

[41] W. Wang, "Processing of massive audit data streams for real-time anomaly intrusion detection,", Elsevier B.V.. vol. 31, pp. 58–72, 2008.

[42] S. Anwar, J. M. Zain, M. F. Zolkipli, and Z. Inayat, "From Intrusion Detection to an Intrusion Response System : Fundamentals , Requirements , and Future Directions.", Algorithms 2017,

[43] N. Ramesh, “A SURVEY OF DIFFERENT TYPES OF NETWORK SECURITY,” , International Journal of Advanced Computational Engineering and Networking, pp. 28–31, 2013.

[44] A. Wahid, “A Survey On Attacks , Challenges and Security Mechanisms In Wireless Sensor Network,” IJIRST, vol. 1, no. 8, pp. 189–196, 2015.

[45] E. Mohammed, “A Survey on Latest DoS Attacks : Classification and Defense Mechanisms A Survey on Latest DoS Attacks : Classification and Defense Mechanisms.”, IJMES,2013.

[46] K. Chatterjee, “DoS / DDoS Attacks Using IPtables Firewall,” vol. 4, no. 3, ijcst ,2013.

[47] Essa Yahya M Muharish ,E. Theses, “PACKET FILTER APPROACH TO DETECT DENIAL OF SERVICE,” 2016.

[48] Mustafa Khambatta . Cloud State Comparative Analysis Based on Survey of DDOS Attacks ’ Detection Techniques at Transport , Network , and Application Layers,” .theRepository at St ,2019.

[49] N. Ahmad,Master Thesis “Analysis of Network Security Threats and Vulnerabilities by Development & Implementation of a Security Network Monitoring Solution,” , Electrical Engineering Thesis No: MEE10:76 Sep 2010.

[50] I. Journal and R. Technology, “An Efficient Classifier for U2R, R2L, DoS Attack,” Int. J. Recent Technol. Eng., vol. 9, no. 1, pp. 644–647, 2020.

[51] D. Hassan, “Cost-Sensitive Access Control for Detecting Remote to Local (R2L) and User to Root (U2R) Attacks,” Int. J. Comput. Trends Technol., vol. 43, no. 2, pp. 124–129, 2017.

[52] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” 2015 Mil. Commun. Inf. Syst. Conf. MilCIS 2015 - Proc., no. November, 2015.

[53] Y. A. Alsariera, “Detecting Generic Network Intrusion Attacks using Tree-based Machine Learning Methods,” vol. 12, no. 2, pp. 597–603, IJACSA ,2021.

[54] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, A. Hotho, and C. R. Mar, “A Survey of Network-based Intrusion Detection Data Sets,” Computer & Security,2019,pp. 1–15.

[55] D. D. Proti, “REVIEW OF KDD CUP ‘ 99 , NSL-KDD AND KYOTO 2006 + DATASETS,” pp. 580–596. MILITARY TECHNICAL COURIER, 2018,

[56] M. S. Al-daweri, K. Akram, Z. Ari, and S. Abdullah, “An Analysis of the KDD99 and UNSW-NB15 Datasets for the Intrusion Detection System,”. Symmetry 2020,

[57] Z. Zoghi, G. Serpen, and C. Science, “UNSW-NB15 Computer Security Dataset : Analysis through Visualization.”. Xiv:2101.05067, 2021 - arxiv.org

[58] M. Gerhard, S. Li, and G. Carle, “Traffic Anomaly Detection Using K-Means Clustering.”. citeseerx.2007.

[59] C. A. Palacios, J. A. Reyes-suárez, L. A. Bearzotti, V. Leiva, and C. Marchant, “Knowledge Discovery for Higher Education Student Retention Based on Data Mining : Machine Learning Algorithms and Case Study in Chile,” Entropy ,pp. 1–23, 2021.

[60] B. Ahmad, W. Jian, and Z. A. Ali, “Role of Machine Learning and Data Mining in Internet Security : Standing State with Future Directions,” vol. 2018, 2018.

[61] S. Soheily-khah and B. Nicolas, “Intrusion detection in network systems through hybrid supervised and unsupervised machine learning process : a case study on the ISCX dataset,” 2018.

[62] Daniel T. Larose • Chantal D. Larose, Discovering knowledge in data. Wiley Series on Methods and Applications in Data Mining.2014.

[63] K. Potdar, “A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers,”, International Journal of Computer Applications , vol. 175, no. 4, pp. 7–9, 2017.

[64] C. Science and S. Publications, “Data Mining : A Preprocessing Engine Luai Al Shalabi , Zyad Shaaban and Basel Kasasbeh Applied Science University , Amman , Jordan,” vol. 2, no. 9, pp. 735–739, 2006.

[65] W. Wang, X. Zhang, S. Gombault, and S. J. Knapskog, “Attribute normalization in network intrusion detection,” I-SPAN 2009 - 10th Int. Symp. Pervasive Syst. Algorithms, Networks, no. November 2014, pp. 448–453, 2009.

[66] R. A. I. Alhayali, M. Aljanabi, A. H. Ali, M. A. Mohammed, and T. Sutikno, “Optimized machine learning algorithm for intrusion detection,” Indones. J. Electr. Eng. Comput. Sci., vol. 24, no. 1, pp. 590–599, 2021.

[67] K. Transactions and O. N. Internet, “Feature Selection Algorithms in Intrusion Detection System : A Survey,” vol. 12, no. 10, pp. 5079–5099, 2018 KSII.

[68] A. A. Salih and M. B. Abdulrazaq, “Combining Best Features Selection Using Three Classifiers in Intrusion Detection System,” 2019 Int. Conf. Adv. Sci. Eng. ICOASE 2019, pp. 94–99, 2019.

[69] N. El, “Review on Wrapper Feature Selection Approaches.”, 2016 IEEE

[70] Salam Saad Alkafagi 1, Rafah M.Almuttairi, “PAPER • OPEN ACCESS A Proactive Model for Optimizing Swarm Search Algorithms for Intrusion Detection System,” IOP Publishing,2021.

[71] A. Yahya Zebari, S. M. Almufti, and C. Mohammed Abdulrahman, “Bat algorithm (BA): review, applications and modifications,” Int. J. Sci. World, vol. 8, no. 1, p. 1, 2020.

[72] A. M. Taha, A. Mustapha, and S. Der Chen, “Naive Bayes-guided bat algorithm for feature selection,” Sci. World J., vol. 2013, 2013.

[73] R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P.

Papa, and X. S. Yang, “BBA: A binary bat algorithm for feature selection,” Brazilian Symp. Comput. Graph. Image Process., no. August, pp. 291–297, 2012.

[74] X. S. Yang, “Bat algorithm: Literature review and applications,” Int. J. Bio-Inspired Comput., vol. 5, no. 3, pp. 141–149, 2013.

[75] S. E. E. Profile, “Evaluation Metrics for Intrusion Detection Systems - A Study,” , IJCSMA,2014.

[76] L. Dalton et al., “Clustering Algorithms : On Learning , Validation , Performance , and Applications to Genomics,” pp. 430–445, 2009.

[77] J. Han, Micheline Kamber, Jian Pei, Data Mining. ISBN 978-0-12-381479-1.

[78] S. M. Shareef and S. H. Hashim, “An Approach Based on Decision Tree and Self-Organizing Map For Intrusion Detection,” Iraqi J. Sci., vol. 58, no. 3B, pp. 1503–1515, 2017.

[79] S. Shareef and S. Hashim, “Proposed Hybrid Classifier to Improve Network Intrusion Detection System using Data Mining Techniques,” Eng. Technol. J., vol. 38, no. 1B, pp. 6–14, 2020.

[80] H. Wilhelmiina, “Descriptive and Predictive Modelling Techniques for Educational Technology,” no. March, 2006.

[81] K. Features, M. J. Zaki, C. Science, and N. York, Data mining and analysis. University of Cambridge.2014 .

[82] Jiawei Han and Micheline Kamber and Jian Pei, INTRODUCTION TO DATA MINING. Elsevier Inc.2011.

[83] N. A. Shah and R. Paul, “Survey On Different Grid Based Clustering Algorithms Of Data Mining,” IJARIIE-ISSN(O)-2395-4396no. 1, pp. 1118–1123, 2017.

[84] S. M. McElwee, “Probabilistic Clustering Ensemble Evaluation for Intrusion Detection,” ProQuest Diss. Theses, no. 1044, p. 166, 2018.

[85] Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman “Mining of Massive

Datasets," stanford.edu,2014.

[86] N. Bouhmala, "How Good is the Euclidean Distance Metric for the Clustering Problem" DOI: 10.1109/IIAI-AAI.2016.26.

[87] A. Bermanis, G. Wolf, and A. Averbuch, "Diffusion-based kernel methods on Euclidean metric measure spaces," 2015 Elsevier vol. 41, pp. 190–213.

[88] X. Ye and T. Sakurai, "Robust Similarity Measure for Spectral Clustering Based on Shared Neighbors," ETRI Journal,2016.

[89] K. M. Ting, "IMPROVING THE EFFECTIVENESS AND EFFICIENCY OF STOCHASTIC NEIGHBOUR EMBEDDING WITH ISOLATION KERNEL," , jair,2021.

[90] M. Noe and S. Garcia, Doctor of Philosophy "FRACTAL DIMENSION FOR CLUSTERING AND UNSUPERVISED AND SUPERVISED FEATURE SELECTION," UMI U585577,no. March, 2011.

[91] D. Al-Shammary, I. Khalil, and Z. Tari, "A distributed aggregation and fast fractal clustering approach for SOAP traffic," J. Netw. Comput. Appl., vol. 41, no. 1, pp. 1–14, 2014.

[92] J. Lin, "Accelerating Density Peak Clustering Algorithm," pp. 1–18, Symmetry 2019.

[93] S. D. Autoencoders, B. Duan, L. Han, Z. Gou, Y. Yang, and S. Chen, "Clustering Mixed Data Based on Density Peaks and Stacked Denoising Autoencoders" Symmetry 2019,.

[94] Z. Jiang, X. Liu, and M. Sun, "A Density Peak Clustering Algorithm Based on the K-Nearest Shannon Entropy and Tissue-Like P System," Hindawi, 2019.

[95] Salam saad. Alkafagi and R. M. Almuttairi, "Enhance Density Peak Clustering Algorithm for Anomaly Intrusion Detection System," Period. Eng. Nat. Sci., vol. 9, no. 2, pp. 965–975, 2021.

[96] M. Du, S. Ding, and Y. Xue, "A robust density peaks clustering algorithm

using fuzzy neighborhood," Int. J. Mach. Learn. Cybern., vol. 0, no. 0, p. 0, 2017.

[97] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," Science (80-.)., vol. 344, no. 6191, pp. 1492–1496, 2014.

[98] M. Salem, "Adaptive Real-time Anomaly-based Intrusion Detection using Data Mining and Machine Learning Techniques " University of Kassel, 2014.

[99] N. Oliveira, I. Praça, E. Maia, and O. Sousa, "Intelligent cyber attack detection and classification for network-based intrusion detection systems," Appl. Sci., vol. 11, no. 4, pp. 1–21, 2021.

[100] A. Drewek, O. Mariusz, and P. Jacek, "A survey of neural networks usage for intrusion detection systems," Journal of Ambient Intelligence and Humanized Computing pp. 497–514, 2020.

[101] A. H. Alsaeedi, A. H. Aljanabi, M. E. Manna, and A. L. Albukhnefis, "A Proactive Metaheuristic Model for Optimizing Weights of Artificial Neural Network," Bulletin of Electrical Engineering and Informatics vol. 9, no.3, pp. 1–9, 2020.

[102] M. M. Lau and K. H. Lim, "Investigation of Activation Functions in Deep Belief Network," 2nd International Conference on Control and Robotics Engineeringvol. 1, pp. 201–206, 2017.

[103] S. Sharma and S. Sharma, "ACTIVATION FUNCTIONS IN NEURAL NETWORKS," IJEAST ,vol. 4, no. 12, pp. 310–316, 2020.

[104] O. Bonde and L. Karlsson, "A Comparison of Selected Optimization Methods for Neural Networks," DEGREE PROJECT IN TECHNOLOGY, FIRST CYCLE, 15 CREDITS STOCKHOLM, SWEDEN 2020

[105] N. Munaiah, A. Meneely, R. Wilson, and B. Short, "Are Intrusion Detection Studies Evaluated Consistently ? A Systematic Literature Review A Systematic Literature Review," scholarworks.rit.edu 2016.

Appendix A

NSL-KDD Dataset

The features name and description are shows below:

no	*feature name*	*description*	*type*
1	duration	length (number of seconds) of the connection	continuous
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	nominal
3	service	network service on the destination, e.g., http, telnet, etc.	nominal
4	flag	normal or error status of the connection	nominal
5	src_bytes	number of data bytes from source to destination	continuous
6	dst_bytes	number of data bytes from destination to source	continuous
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong_fragment	number of ``wrong'' fragments	continuous
9	urgent	number of urgent packets	continuous
10	hot	number of ``hot'' indicators	continuous
11	num_failed_logins	number of failed login attempts	continuous
12	logged_in	1 if successfully logged in; 0 otherwise	discrete
13	num_compromised	number of ``compromised'' conditions	continuous
14	root_shell	1 if root shell is obtained; 0 otherwise	discrete

15	su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
16	num_root	number of ``root'' accesses	continuous
17	num_file_creations	number of file creation operations	continuous
18	num_shells	number of shell prompts	continuous
19	num_access_files	number of operations on access control files	continuous
20	num_outbound_cmds	number of outbound commands in an ftp session	continuous
21	is_hot_login	1 if the login belongs to the ``host'' list; 0 otherwise	discrete
22	is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete
23	count	number of connections to the same host as the current connection in the past two seconds	continuous
24	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
25	serror_rate	% of connections that have ``SYN'' errors	continuous
26	srv_serror_rate	% of connections that have ``SYN'' errors	continuous
27	rerror_rate	% of connections that have ``REJ'' errors	continuous
28	srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
29	same_srv_rate	% of connections to the same service	continuous

30	diff_srv_rate	% of connections to different services	continuous
31	srv_diff_host_rate	% of connections to different hosts	continuous
32	dst_host_count	% of connections from the same host to destination host	continuous
33	dst_host_srv_count	% of connections from the same host with same service to destination host	continuous
34	dst_host_same_srv_rate	% of connection to same service ports from destination host	continuous
35	dst_host_diff_srv_rate	% of different services on the current host	continuous
36	dst_host_same_src_port_rate	% of connection to same service port from destination host	continuous
37	dst_host_srv_diff_host_rate	% of connection to the same service come from different host	continuous
38	dst_host_serror_rate	% of connection to current host that has an so error	continuous
39	dst_host_srv_serror_rate	% of connection to same service that has SYN errors from destination host	continuous
40	dst_host_rerror_rate	% of connection to the current host has an RST error	continuous
41	dst_host_srv_rerror_rate	% of connection to the current host and specified service that has an RST error	continuous

The features name and description after the One-Hot encoding process for all features of NSL-KDD dataset are shows below:

No	*feature name*	*description*	*type*
0	duration	length (number of seconds) of the connection	continuous
1	src_bytes	number of data bytes from source to destination	continuous
2	dst_bytes	number of data bytes from destination to source	continuous
3	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
4	wrong_fragment	number of ``wrong'' fragments	continuous
5	urgent	number of urgent packets	continuous
6	hot	number of ``hot'' indicators	continuous
7	num_failed_logins	number of failed login attempts	continuous
8	logged_in	1 if successfully logged in; 0 otherwise	discrete
9	num_compromised	number of ``compromised'' conditions	continuous
10	root_shell	1 if root shell is obtained; 0 otherwise	discrete
11	su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
12	num_root	number of ``root'' accesses	continuous
13	num_file_creations	number of file creation operations	continuous
14	num_shells	number of shell prompts	continuous
15	num_access_files	number of operations on access control files	continuous
16	num_outbound_cmds	number of outbound commands in an ftp session	continuous
17	is_hot_login	1 if the login belongs to the ``host'' list; 0 otherwise	discrete

18	is_guest_login	1 if the login is a ``guest"login; 0 otherwise	discrete
19	count	number of connections to the same host as the current connection in the past two seconds	continuous
20	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
21	serror_rate	% of connections that have ``SYN" errors	continuous
22	srv_serror_rate	% of connections that have ``SYN" errors	continuous
23	rerror_rate	% of connections that have ``REJ" errors	continuous
24	srv_rerror_rate	% of connections that have ``REJ" errors	continuous
25	same_srv_rate	% of connections to the same service	continuous
26	diff_srv_rate	% of connections to different services	continuous
27	srv_diff_host_rate	% of connections to different hosts	continuous
28	dst_host_count	% of connections from the same host to destination host	continuous
29	dst_host_srv_count	% of connections from the same host with same service to destination host	continuous
30	dst_host_same_srv_rate	% of connection to same service ports from destination host	continuous

31	dst_host_diff_srv_rate	% of different services on the current host	continuous
32	dst_host_same_src_port_rate	% of connection to same service port from destination host	continuous
33	dst_host_srv_diff_host_rate	% of connection to the same service come from different host	continuous
34	dst_host_serror_rate	% of connection to current host that has an so error	continuous
35	dst_host_srv_serror_rate	% of connection to same service that has SYN errors from destination host	continuous
36	dst_host_rerror_rate	% of connection to the current host has an RST error	continuous
37	dst_host_srv_rerror_rate	% of connection to the current host and specified service that has an RST error	continuous
38	icmp	type of the protocol	nominal
39	udp	type of the protocol	nominal
40	tcp	type of the protocol	nominal
41	other	network service on the destination	nominal
42	link	network service on the destination	nominal
43	Netbios_ ssn	network service on the destination	nominal
44	smtp	network service on the destination	nominal
45	netstat	network service on the destination	nominal
46	ctf	network service on the destination	nominal

47	ntp_u	network service on the destination	nominal
48	harvest	network service on the destination	nominal
49	efs	network service on the destination	nominal
50	klogin	network service on the destination	nominal
51	systat	network service on the destination	nominal
52	exec	network service on the destination	nominal
53	nntp	network service on the destination	nominal
54	pop_3	network service on the destination	nominal
55	printer	network service on the destination	nominal
56	vmnet	network service on the destination	nominal
57	netbios_ns	network service on the destination	nominal
58	urh_i	network service on the destination	nominal
59	ssh	network service on the destination	nominal
60	http_8001	network service on the destination	nominal
61	iso_tsap	network service on the destination	nominal
62	aol	network service on the destination	nominal

63	sql_net	network service on the destination	nominal
64	shell	network service on the destination	nominal
65	supdup	network service on the destination	nominal
66	auth	network service on the destination	nominal
67	whois	network service on the destination	nominal
68	discard	network service on the destination	nominal
69	sunrpc	network service on the destination	nominal
70	urp_i	network service on the destination	nominal
71	Rje	network service on the destination	nominal
72	ftp	network service on the destination	nominal
73	daytime	network service on the destination	nominal
74	domain_u	network service on the destination	nominal
75	pm_dump	network service on the destination	nominal
76	time	network service on the destination	nominal
77	hostnames	network service on the destination	nominal
78	name	network service on the destination	nominal

79	ecr_i	network service on the destination	nominal
80	bgp	network service on the destination	nominal
81	telnet	network service on the destination	nominal
82	domain	network service on the destination	nominal
83	ftp_data	network service on the destination	nominal
84	nnsp	network service on the destination	nominal
85	courier	network service on the destination	nominal
86	finger	network service on the destination	nominal
87	uucp_path	network service on the destination	nominal
88	X11	network service on the destination	nominal
89	imap4	network service on the destination	nominal
90	mtp	network service on the destination	nominal
91	login	network service on the destination	nominal
92	tftp_u	network service on the destination	nominal
93	kshell	network service on the destination	nominal
94	private	network service on the destination	nominal

95	http_2784	network service on the destination	nominal
96	echo	network service on the destination	nominal
97	http	network service on the destination	nominal
98	idap	network service on the destination	nominal
99	tim_i	network service on the destination	nominal
100	netbios_dgm	network service on the destination	nominal
101	uucp	network service on the destination	nominal
102	eco_i	network service on the destination	nominal
103	Remote_job	network service on the destination	nominal
104	IRC	network service on the destination	nominal
105	http_443	network service on the destination	nominal
106	red_i	network service on the destination	nominal
107	Z39_50	network service on the destination	nominal
108	Pop_2	network service on the destination	nominal
109	gopher	network service on the destination	nominal
110	Csnet_ns	network service on the destination	nominal

111	OTH	normal or error status of the connection	nominal
112	S1	normal or error status of the connection	nominal
113	S2	normal or error status of the connection	nominal
114	RSTO	normal or error status of the connection	nominal
115	RSTRs	normal or error status of the connection	nominal
116	RSTOS0	normal or error status of the connection	nominal
117	SF	normal or error status of the connection	nominal
118	HS	normal or error status of the connection	nominal
119	REJ	normal or error status of the connection	nominal
120	S0	normal or error status of the connection	nominal
121	S3	normal or error status of the connection	nominal

UNSW-NB15 Dataset

The UNSW-NB15 Source Files

The raw network packets (Pcap files) of the UNSW-NB 15 data set is created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic contemporary attack activities. The UNSW-NB15 source files are provided in different formats, Pcap files, BRO files, Argus Files and CSV files. The source files

of the data set were divided based in the date of the simulation 22-1-2015 and 17-2-2015, respectively. The descriptions of these simulations are provided in the report files to show the network configurations and the actions of the attack types during the simulation.

The folders of the UNSW-NB15 data set can be outlined as follows.

1. **Pcap files** 1.1 Pacps 22-1-2015 **1.2** Pcaps 17-2-2015	2. **BRO files** 2.1 BRO 22-1-2015 **2.2** BRO 17-2-2015
3. **Argus files** 3.1 Argus 22-1-2015 **3.2** Argus 17-2-2015	4. **Report Files** 4.1 Report 22-1-2015 **4.2** Report 17-2-2015
5. **CSV Files (for classification purposes)** **5.1 As in this paper:** Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)."Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015.	

Note: Free use of the UNSW-NB15 dataset for academic research purposes is hereby granted in perpetuity. Use for commercial purposes is strictly prohibited. Nour Moustafa and Jill Slay have asserted their rights under the Copyright.

The features name and description are shown below.

No.	Name	Description
1.	Dur	record total duration
2.	Proto	Transaction protocol
3.	State	the state and its dependent protocol
4.	Service	HTTP,FTP SMTP
5.	Sbytes	Source to destination transaction byte
6.	Dbytes	Destination to source transaction bytes
7.	Sttl	Source to destination time to live value
8.	Dttl	Destination to source time to live value
9.	Sloss	Source packets retransmitted or dropped
10.	Dloss	Destination packets retransmitted or dropped

11.	Sload	Source bits per second
12.	Dload	Destination bits per second
13.	Spkts	Source to destination packet count
14.	Dpkts	Destination to source packet count
15.	Swin	Source TCP window advertisement
16.	Dwin	Destination TCP window advertisement
17.	Stcpb	Source TCP base sequence number
18.	Dtcpb	Destination TCP base sequence number
19.	Smean	Mean of the packets transmitted by src
20.	Dmean	Mean of the packets transmitted by des
21.	Trans_depth	depth the connection of http request/response transaction
22.	Res_bdy_len	uncompressed content size of the data
23.	Sjit	Source jitter
24.	Djit	Destination jitter
25.	Sintpkt	source inter packet arrival time
26.	Dintpkt	destination inter packet arrival time
27.	Tcprtt	TCP connection setup round-trip time
28.	Synack	TCP connection setup time
29.	Ackdat	TCP connection setup time
30.	Is_sm_ips_ports	If source 1 and destination 3, IP addresses equal and port numbers 2,4 equal then, this variable takes value 1 else 0
31.	Ct_state_ttl	No. for each state (6) according to specific range of values for source/destination time to live (10) (11)
32.	Ct_flw_http_mthd	No. of flows that has methods like Get and Post in http
33.	Is_ftp_login	If ftp session is accessed by user and password then 1 else0
34.	Ct_ftp_cmd	No of flows that has a command in ftp session
35.	Ct_srv_src	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).

36.	Ct_srv_dst	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
37.	Ct_dst_ltm	No. of connections of the same destination address (3) in 100 connections according to the last time (26)
38.	Ct_src_ ltm	No. of connections of the same source address (1) in 100 connections according to the last time (26).
39.	Ct_src_dport_ltm	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26)
40.	Ct_dst_sport_ltm	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
41.	Ct_dst_src_ltm	No of connections of the same source (1) and the destination (3) address in 100 connections according to the last time (26).
42.	Attack_cat	The name of each attack category. In this dataset, nine categories, e.g. Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and worm
43.	Label	binary,0 for normal and 1 for attack records

Appendix B

A Proactive Model for Optimizing Swarm Search Algorithms for Intrusion Detection System

Salam Saad Alkafagi [1], Rafah M.Almuttairi[2]

College of Information Technology, University of Babylon, Babylon, 51002, Iraq.

*Corresponding author: salam.saad@student.uobabylon.edu.iq
Rafah@babylon.edu.iq

Abstract: This paper proposes a novel a Proactive model for Swarm Optimization for feature selection (PMSO) of intrusion detection system. The proposed model potentially enhances the search process and initial value of population for swarm optimization algorithms. The swarm optimization often suffering from falling in a local optimum, one of the main causes of this problem is the initial value of the population. . Our proposed model is applied with the aim to address these challenges by restarting generate the population when system sensed the search process go forward stagnation. This process will give high exploration and reduce the probability of Stagnation in local optima. Benchmark datasets, namely, NSL-KDD datasets is used to demonstrate and validate the performance of the proposed model for intrusion detection. Experimentally, the machine learning accuracy has provided better classification results with proposed system (%2) than swarm optimization (Particle Swarm Optimization and Bat Algorithm).

Keywords: machine learning, Particle swarm algorithm, Bat algorithm, Feature selection, Intrusion Detection System, Swarm Optimization.

Periodicals of Engineering and Natural Sciences ISSN 2303-4521
Vol. 9, No. 2, June 2021, pp.965-975

Enhance density peak clustering algorithm for anomaly intrusion detection system

Salam Saad Alkafagi [1], Rafah M.Almuttairi[2]
[1,2]College of Information Technology, University of Babylon, Babylon, 51002, Iraq

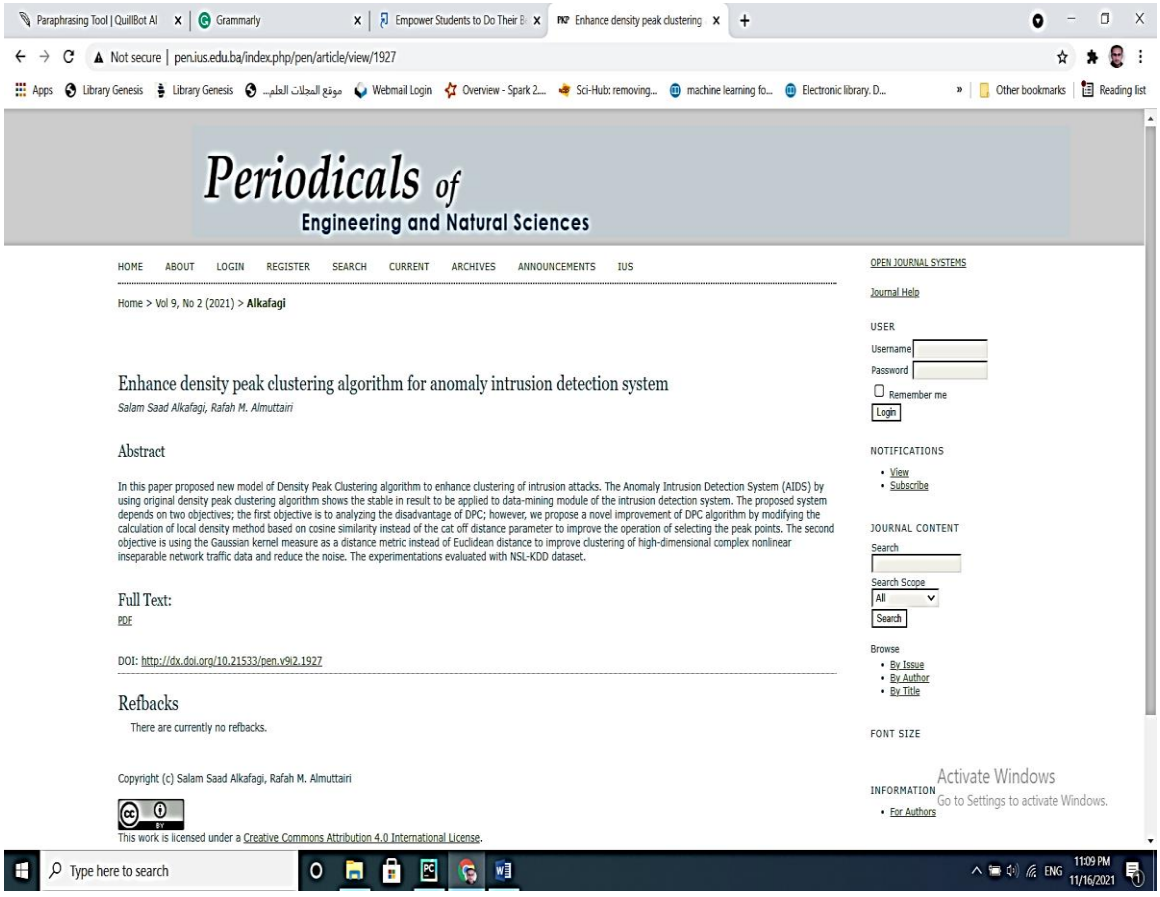
ABSTRACT

In this paper proposed new model of Density Peak Clustering algorithm to enhance clustering of intrusion attacks. The Anomaly Intrusion Detection System (AIDS) by using original density peak clustering algorithm shows the stable in result to be applied to data-mining module of the intrusion detection system. The proposed system depends on two objectives; the first objective is to analyzing the disadvantage of DPC; however, we propose a novel improvement of DPC algorithm by modifying the calculation of local density method based on cosine similarity instead of the cat off distance parameter to improve the operation of selecting the peak points. The second objective is using the Gaussian kernel measure as a distance metric instead of Euclidean distance to improve clustering of high-dimensional complex nonlinear inseparable network traffic data and reduce the noise. The experimentations evaluated with NSL-KDD dataset.

Keywords: Data Mining, Anomaly Intrusion Detection System, Density Peak Cluster algorithm

Corresponding Author:

Salam Saad Alkafagi
College of Information Technology
University of Babylon
Babylon, Iraq
salam.saad@student.uobabylon.edu.iq



الخلاصة

يمكن تعريف أي طريقة أو عملية أو وسيلة من شأنها إضعاف أمان الشبكة بشكل ضار على أنها هجوم على الشبكة. تعد أنظمة الكشف عن اختراق الشبكة (NIDS) حاسمة في الكشف عن عمليات اختراق الشبكة في مجال الأمن القومي. تعتمد أنظمة الكشف عن اختراق الشبكة على النماذج التقليدية مثل التعلم الآلي أو خوارزميات التجميع التي تعاني من انخفاض دقة التصنيفات المتعددة وايضا تعاني بشكل كبير من تقليل اختلال التوازن لفئات الهجوم ذات معدل التدفق المنخفض في بيانات الشبكة .

في هذه الاطروحة ، تم تطوير نظام اكتشاف اختراق الشبكة استنادًا إلى تقنيات الجمع بين مجموعات التنقيب عن البيانات غير الخاضعة للإشراف والتعلم الآلي الخاضع للإشراف للكشف عن أنواع متعددة من الهجمات في تدفق الشبكة. يتكون النظام المقترح من أربع مراحل رئيسية:

المرحلة الأولى هي المعالجة المسبقة ، وتتكون من خطوتين فرعيتين رئيسيتين هما طريقة (One-hot encoding) وطريقة (Min-Max).

المرحلة الثانية هي اختيار الميزات ذات الصلة باستخدام خوارزمية سرب الخفافيش BAT الاستباقي كطريقة لاختيار الصفات. تتمثل واجبات خوارزمية اختيار الصفات في التخلص من السمات الزائدة وغير المؤثرة وتقليل تأثير هذه السمات على تجانس توزيع بيانات التدريب في التجميع وعملية التنبؤ النهائية.

المرحلة الثالثة هي نموذج اكتشاف الشذوذ ، والذي يستخدم خوارزمية تجميع ذروة الكثافة المحسّنة(improved density peak clustering) لتجميع عينات البيانات المنتظمة عن بيانات الهجمات لكسر عدم توازن مجموعة البيانات. ثم يتم استخدام مصنف الشبكة العصبية الاصطناعية (multilayer perceptron)للتنبؤ بفئات الهجوم غير المرنة ومنخفضة السرعة بمساعدة درجة العضوية الفركتالية(fractal fuzzy membership).

تقويم النظام هو المرحلة الرابعة ، حيث يتم تقويم النظام المقترح باستخدام مؤشرات الأداء مثل الدقة ومعدل الكشف (الاسترجاع) والدقة ودرجة F1. يتم استخدام مجموعتين من بيانات حركة مرور الشبكة: NSL_KDD و UNSW_NB15 . يتكون كل منها من حزم هجوم منتظمة ومتعددة الأنواع. أظهرت النتائج التي تم الحصول عليها أن أفضل نظام كشف يتم تحقيقه باستخدام تقنيات الجمع القائمة على خوارزمية مجموعة ذروة الكثافة المحسّنة (IDPC) وخوارزمية تصنيف multilayer perceptron بمساعدة وظيفة درجة العضوية (fractal fuzzy membership). تبلغ الدقة الإجمالية لمجموعة بيانات NSL_KDDTest-21

82.47٪ ، والدقة 83.44٪ ، والاسترجاع (معدل الكشف) 82.47٪ ، و f1.score 82.95٪ والدقة الإجمالية لمجموعة بيانات NSL_KDDTest + 92.24٪، والدقة 92.68٪ ، و الاسترجاع (معدل الكشف) 92.24٪ ، الدرجة الأولى 92.46٪. في حين أن الدقة الإجمالية في مجموعة البيانات UNSW_NB15 هي 95.59٪ ، والدقة 96.68٪ ، والاسترجاع (معدل الكشف) 95.59٪ ، و f1.score 96.13٪ ، ويتم ذلك في وقت فعال (بالثواني).

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم البرمجيات

تحسين نظام كشف الاختراق الشبكي اعتماداً على تقنية اكتشاف المعرفة الاستباقية

اطروحة مقدمة لقسم البرمجيات/ كلية تكنولوجيا المعلومات/ جامعة بابل

كجزء من اكمال متطلبات منح درجة الدكتوراه

فلسفة في تكنولوجيا المعلومات/ برمجيات

مِن قِبَل

سلام سعد محمد علي عبد الله

بأشراف

أ.د. رفاه محمد كاظم خضير

2022 A.D. **1443 A.H.**