



Department of Information Technology

**Automatic Test Data Generation for Big
Data Input Values**

Prepared by

Saoud Mohammed Abed Al-Bqur

Supervised by

Dr.Mohammad Ali H.Eljinini

Co-Supervised by

Dr.Osama Qtaish

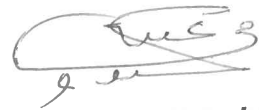
**This Thesis is submitted to the Faculty of Information Technology as a
partial fulfillment of the Requirement for Master Degree in Software
Engineering**

July 2020

أقرار تفويض

انا سعود محمد عبد البقور افوض جامعة الاسراء للدراسات العليا بتزويد نسخ من رسالتي ورقيا والكترونيا للمكتبات والمنظمات او الهيئات والمؤسسات المعنية بالأبحاث والدراسات العليا عند طلبها.

سعود محمد عبد البقور

التوقيع: 

التاريخ: 2020/7/14

Authorization statement


Saoud Mohammad abed albqour authorizes Isra University to provide to provide hard and software copies of his thesis to libraries for the institutions or individuals upon request.

Saoud mohammad abed albaqour

Signature 

Date :14/7/2020


The undersigned have examined the thesis entitled **Automatic Test Data Generation for Big Data Input Values** Presented by **SAOUD MOHAMMAD ABED ALBAQOUR**, a candidate for the degree of master in software Engineering and hereby certify acceptance.


.....
c.c./v/18
Date

Dr.Mohammad Ali ELjinini


.....
c.c./v/18
Date

Dr.Osama Qtaish


.....
c.c./v/10
Date

Dr.Belal Swan


.....
c.c./v/18
Date

Dr.Thamer ALrosan

DEDICATION

This thesis is dedicated

To the beloved wife who stands with me supports me and calls for my success.

To the candles of my life; My sons Mohammed and Mourad, and all friends who are loyal to their unlimited love and support, for all of them, I dedicate this humble work.

ACKNOWLEDGMENT

First of all, I give thanks and praise to **Allah** for his mercy and his blessing for giving me the opportunity for this achievement I would like to express my thanks and gratitude to my **supervisor Dr.Mohammad Ali H.Eljinini** for his continuous academic support, suggestions, guidance, and patience throughout the project progress, besides his brotherly care.

Thanks are also **co-supervisor Dr.Osama Qtaish, Dr.Aysh ALhroob, Dr.Wael zyadat**, and the Faculty members who were generous in their teaching and academic services.

Finally, my deepest thanks and unlimited love for my **wife and sons** and all my friends who were so helpful and supportive me to complete my thesis.

TABLE OF CONTENTS

DEDICATION	ii
Acknowledgment	iii
Table of Contents	iv
List of figures.....	vi
List of Tables	vii
ABSTRACT.....	Error! Bookmark not defined.
Chapter One	2
Introduction	2
1. 1 Overview	2
1. 2 Background Information	4
1. 3 Problem of Statement	7
1. 4 Research Questions	7
1. 5 Research Objectives	8
1. 7 Significance of Study	9
1.8 Thesis Overview	9
Chapter Two	10
Literature Review	10
2.1 Overview	10
2.2 General Intrusion Detection	10
2.3 CICIDS2017 Dataset	16
2.4 Bat Algorithm	20
2.4.1 Echolocation of Micro-bats	20
2.4.2 Bat Motion	21
2.4.3 Loudness and Pulse Emission	22
2.4.3 Intrusion Detection Using Bat Algorithm	22

Chapter Three	23
Research Methodology	23
3.1 Overview	23
3.2 General Framework	23
3.3 Dataset	25
3.4 Methodology Phases	25
3.4.1 Dataset Acquisition (Raw)	26
3.4.2 Feature Selection	26
3.4.3 Working of the Bat Algorithm	27
3.5 Data Analysis and Interpretation	27
3.5 Research Tools	29
Chapter Four	30
Results and Discussions	30
4.1 Overview	30
4.1 Working with KNIME Tool	30
Chapter Five	42
Conclusions and Future Work	42
5.1 Conclusions	42
5.2 Future Work	42
REFERENCES	44

LIST OF FIGURES

Figure 1.1: The Classification Methods for IDSs	4
Figure 3.1: General framework	25
Figure 4.1: KNIME Model	32
Figure 4.2: File Table	33
Figure 4.3: Pattern Matching	34
Figure 4.4: The Visualizes Data	35
Figure 4.5: Final Results	36

LIST OF TABLES

Table 2.1: Summary of Intrusion Detection Techniques	12
Table 2.2: Network Flow Analyzer (Panigrahi, and Borah, 2018)	15
Table 2.3: Overall characteristics of CICIDS2017 dataset	16
Table 2.4: Class wise instance occurrence of CICIDS2017 dataset	16
Table 2.5: Extracted Features Definition	17
Table 3.1: Network Flow Analyzer (Panigrahi, and Borah, 2018)	26
Table 4.1: Comparing results using Naive Bayes classifier	37
Table 4.2: Comparing results using SVM classifier	37

LIST OF ABBREVIATIONS

IDs	Intrusion Detection System
ADR	Attack Detection Rate
FAR	False Alarm Rate
SVM	Support Vector Machine
CSV	CSV Comma Separated Values
ML	Machine Learning
MOBA	multi-objective bat algorithm
PCA	Principal Component Analysis
ANN	Artificial Neural Network
CFS	Correlation Feature Selection
MLP	Multi-Layer Perceptron
RF	Random Forest
LIBSVM	Library for Support Vector Machine
NN	Neural Network
DBM	Detection-Based Method
SBM	Source-Based Methods
AI	Artificial Intelligence

Abstract

There is needed for the efficient intrusion detection system working over the network system to detect the whole possible attacks. Intrusion detection is so much popular since the last two decades, where intruders attempted to break into or misuse the system. There are many techniques used in intrusion detection (IDS) for protecting computers and networks from network-based and host-based attacks. In this thesis, the proposed approach presents a new model for IDS using a bat algorithm that aims to select the best features using big data. The proposed approach divided into several phases to extract and find all possible features that effect directly in the detection process. The proposed approach was tested using the KNIME Analytics Platform based on Support Vector Machine (SVM) and Naive base classifiers. The experiment results give a high accuracy (97.52%) with reducing the error classification into (2.47%) using the SVM classifier.

Chapter One

Introduction

1.1 Overview

In the last years, the understanding of complex systems and networks increased because of the modern science of networks. The networks are considered as a realistic model of a big system for real life. These networks contain each of the nodes and edges. The nodes are considered as the individual component in the system, while the edges are a path between nodes and represent the correlation in the system (Lang and Liu, 2019).

In recent times, with the increasing number for using Big Data, security has become the number one business concern. Because of the growth of web attacks, information security has become the overarching problem all over the world such as the intrusions over the networks (Srivastava, and Dahiya, 2018).

Intrusion detection using bat algorithm is one of the most important issues in this study, where the main idea aims to divide the network into subnetworks and groups of nodes that consist of the intra-correlations and sparse inter-correlations between each of the others. However, the problem can be represented as two types of objectives (Can, and Sahingoz, 2015). The first objective is the increase of internal links to maximize the ability of the correlation-based on the feature

selection method (CFS), while the second objective is the minimization of external links to increase the principal component analysis (PCA).

The modern networks are playing important roles in the last few decades, and the security issues have become a vital research area in the security sectors, turned into a new vision in data innovation (Alomari and Raheem, 2018). Networks securities are one of the most important problems and challenges in the field of information security. Also, network security is becoming very important to reach the safety and modernity society to ensure protection for the secret data flowing over the networks (Kumar and Kaur, 2016). The security methods mainly consist of each of the firewalls, anti-virus, and intrusion detection systems (IDSs).

Many challenges face computer engineers because hackers can make several dangerous attacks to fracture the systems and networks (Mohamed, Alsawy, and Hefny, 2018). Because of that, the need to create an efficient IDS is a primary target in this field and time.

In this thesis, for applying the automatic test data generation for big data input values, the proposed approach of IDs aims to build a new approach of IDs using a bat algorithm to extract all possible and appropriate features from a big dataset. The proposed approach divided into two main phases, the first phase is the training phase (using MATLAB), while the second phase is the testing phase

(using KNIME Analytics Platform). bat algorithm used in the training phase to extract all features from the raw dataset to build the refined dataset used in the phase. The second phase using the KNIME Analytics Platform aims to test the generated dataset (refined dataset) using two classifiers Naive Bayes, and Support Vector Machine (SVM).

1. 2 Background Information

The first work for the IDs was created in 1980 (Anderson, 1980). Several works and systems have been made since that time until now, but still affected by the increasing number of the false alarm rate that mainly works to generate many alerts for low non-threatening cases. Because of that, the increasing number of alerts are affecting for increasing the loading and efforts for the employees and ignoring some of the real attacks in some case because of the dynamic and updated structures of the networks, this case allows to generate a new different's attacks continuously, that makes a new problem in the network security and challenge for the existing IDSs, because of the lack of ability to detect unknown attacks (Lang and Liu, 2019). From this point, many researchers have focused on their studies to develop IDSs that aim to ensure higher accuracy of detection and extraction with reduced numbers of false alarms as it was in existing IDSs.

IDS are considered as one of the most important research sectors in the network and information security (Juma, Muda, Mohammad, and Yasin, 2015), Many tools are designed to stop the internet-based attacks such as firewall, intrusion prevention system and IDS. Machine learning methods (ML) are one of the artificial intelligence (AI) branches that gain information according to the training data and initial facts. ML methods are defined as a technique that allows machines to learn knowledge dynamically without being programmed according to the initial dataset, where ML methods are working to extract all knowledge from these datasets using a sequence of functions and methods (Haq, 2015). ML methods mainly focus on harbingers. It is classified into three main categories of learning known as "supervised, unsupervised, and reinforcement" (Lang and Liu, 2019).

Where the labeling process for the instances in the supervised learning (classification) starts in the training phase, in the most algorithms for the variant supervised learning (e.g. "Artificial Neural Network, Bayesian Statistics, Gaussian Process Regression, Lazy learning, Nearest Neighbor algorithm, Support Vector Machine, Hidden Markov Model, Bayesian Networks") (Akhilesh and Shrivastava, 2014).

While in "unsupervised learning", the data cases are unlabeled. The main method focusing on this learning is the clustering technique. Also, other techniques

used in this learning are "Cluster analysis (K-means, Fuzzy set), Hierarchical, Self-organizing map, Apriori algorithm, Eclat algorithm, and Outlier detection (Local outlier factor)". The last type of learning is "reinforcement learning", where different computers are interacting with a platform to reach specific goals (Niyaz et al. 2016). Figure 1.1 shown a different algorithm of ML that used to solve the ID problem:

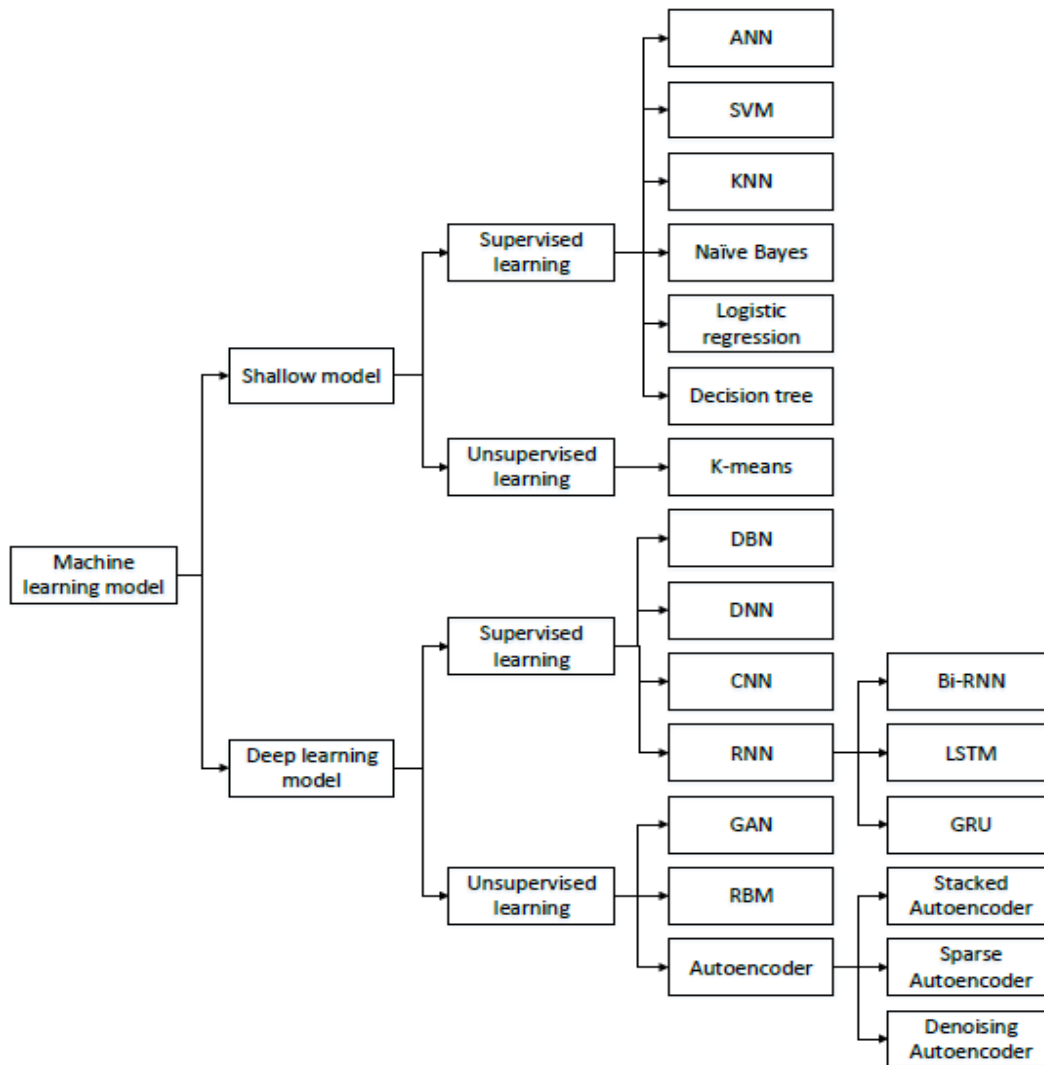


Figure 1.1: The Classification Methods for IDSs (Lang and Liu, 2019).

In intrusion detection systems, there is a type of classification method known as a "detection-based method" (DBM) and data "source-based methods" (SBM). In the DBM, the IDSs are classified into "misuse detection" and "anomaly detection", while the IDSs in the data SBM can be classified into sub-methods as "host-based" and "network-based methods" (Heberlein, Dias, Levitt, Mukherjee, Wood, and Wolber, 1990).

1. 3 Problem of Statement

Because of the increasing rate for the high false alarm which is working to generate several alerts for low non-threatening cases, and because they lack the ability to extract the variant attacks, this research aims to enhance the intrusion detection system by studying the intrusion behavior in the infected networks.

The main problem that we aspire to solve is the to the large numbers of the false alarm, and to reach a higher accuracy of attack detection to decrease the number of false alarm rates as it was in existing IDSs using bat algorithm like Alina et al. (2015) study.

1. 4 Research Questions

The proposed research will discuss the IDS and will try to answer the following questions:

- Are all the features used based on the bat algorithm necessary or should we apply feature selection methods to get the best results?
- Is the proposed methodology robust enough and more efficient with high accuracy comparing to the previous approaches of IDS?
- Are the feature selection methods efficient to get accurate results?

1.5 Research Objectives

In this thesis, three objectives need to be achieved for the proposed approach. There are as follows:

- Applying bat method for extracting the all appropriate features from a big dataset.
- Comparing the bat algorithm with existing techniques of machine learning (SVM and Naive Base).

1.6 Scope of the Study

The methodology will interest in IDS to improve the efficacy for the system using (Sharafaldin, Lashkari, and Ghorbani, 2018) dataset. The proposed dataset contains several labeled network flows and contains each of the full packet payloads in the "PCAP" format, the identical profiles, and the labeled flows. The

limitations of the dataset are the unorganized data that need pre-processing to extract useful features.

1.7 Significance of the Study

The expected outcomes of this thesis are the following:

- Because of the ability and the flexibility of the bat algorithm, the expected outcomes for the proposed method is to reach more flexible IDS with high ability of measurement.
- The more accurate result when automating the detection for the intrusion over the network, because of the interconnection between each of the classical ML algorithms (SVM and Naive Base) and bat algorithm.

1.8 Thesis Overview

The thesis is organized as follows:

In chapter one, general overview and background information about the IDS. In chapter two, the previous works upon which our research draw is introduced. In chapter three, the methodology is described in detail. In chapter four, the experiment results are introduced which are achieved due to the implementation of the proposed approach. In Chapter Five, we review the contributions and conclusion of the thesis and list of possible ideas for future research.

Chapter Two

Literature Review

2.1 Overview

The literature has been revised to review each definition, usage, and benefit of intrusion detection than to determine the existing models and discussed their advantages and weaknesses in intrusion detection. In this chapter, we showed a summary of Intrusion detection techniques, CICIDS2017 Dataset, and the description of the bat algorithm.

2.2 General Intrusion Detection

Levitt, Heberlein, and Mukherjee (1994) considered the IDS as a new modify approach for providing a feeling of security in existing machines, for identity the preferably in real-time, un legal use, misapply, and perversion of the machine systems by both of system legal users and external penetrators.

Zamani (2013) compared the performance of several schemes, and divide the schemes into different methods according to classical AI, and methods based on the "computational intelligence genetic".

Karpagam, Revathi, and Jayakumar (2015) proposed new a approach for IDs based on pertinent features for a specific attack selected. The IDs in their approach were done to help of supervised learning "Neural Network (NN)", where the feature selection based on information gain algorithm and genetic algorithm. They train and test the experiment of the relevant features using the Multi-Layer Perceptron (MLP) supervised NN.

Chowdhury, Ferens, and Ferens (2016) proposed an approach for clustering any irregular behavior for the traffic in the network, using a combination of two ML algorithms. To evaluate the detection accuracy in their approach, they used a "false-positive" rate, "false-negative" rate based on the needed time to detecting the different intrusions. The experiment results show high accuracy of detection reach to 98.76%, a lower "false-positive" rate reach to 0.09%, and a "false-negative" rate reach to 1.15%, whereas the normal support vector machine-based scheme achieved an accuracy of detection to 88.03%, the "false-positive" rate of 4.2% and "false-negative" rate of 7.77%.

Almseidin, Alzubi, Kovacs, and Alkasassbeh (2018) applied several experiments in their approach "J48, Random Forest, Random Tree, Decision Table, MLP, Naive Bayes, and Bayes Network" to estimate a different ML classifier using the KDD dataset, which gave a good understanding for different intrusive behaviors to enhance the extraction rate of the IDs during the attacks. In their

study, they focused on the performance metrics for each of the "false-negative" and the "false-positive". The experiment results show a difference in accuracy between each of the random forest and decision table classifiers. The random forest has the highest rate of accuracy more than decision table classifiers, while the decision table achieved the rate of a false negative.

Alsawy (2018) proposed an approach that aims to categorize the different activities of the network as normal activity or attack using different ML algorithms, like "Random Forest (RF)", "Multi-Layer Perceptron (MLP) ", and "Library for Support Vector Machine (LIBSVM) ". They tested their approach using a common dataset called NSL-KDD and used a different ML algorithm over the dataset. They used the Correlation Feature Selection (CFS) as a "Feature Selection" algorithm to delete some of the irrelevant features from the dataset. The experiment results show the accuracy of the "multilayer perceptron classifier" is equal to 95.7%, while the accuracy for the "Random Forest's" is equal to 99.6%, and accuracy rate for the "LIBSVM classifier" was 94.8%. "Feature Selection (CFS)" showed a low accuracy of 91.7%, but with the "LIBSVM", the accuracy rate raised to 97.2%.

Alomari and Raheem (2018) introduced learning calculation for abnormality based system interruption identification framework utilizing choice tree calculation that recognizes assaults from ordinary practices and distinguishes diverse kinds of

interruptions, which modifies the weights of the dataset in light of probabilities and split the dataset into sub-dataset until all the sub-dataset has a place" with a comparable class. The test comes about on the KDD-99 bench-mark organize interruption discovery data-set exhibit that the suggested algorithm resulted in a 98.5% discovery rate in comparison with other executing techniques.

Fattahi, Omrani, Dallali, and Rhaimi (2018) proposed a hybrid approach of the "artificial neural network (ANN) classifier" and the "support vector machine (SVM)". They compared the accuracy of their approach with the different results for several classifiers. The experiment results have proceeded with a different selected network that connected with the "NSL-KDD DARPA" dataset. The initial results have shown the merge each of "ANN" and "SVM" techniques for attack detection is a positive way direction as to future work.

Biswas (2018) proposed an IDS using machine learning with the mix of feature selection techniques to select the important features from the original data of features and classify it through studying and analyzing the popular classifiers and methods for selections. They applied a five tucks cross-validation to find the results and accuracy of the "NSL-KDD" dataset. The experiment results show that the "K-NN" classifier has a high performance and efficiency than other methods, and the information gain ratio based feature selection method is better.

Table 2.1: Summary of Intrusion Detection Techniques.

Reference	Algorithm	Accuracy	Weakness
Karpagamet al. (2015)	<ul style="list-style-type: none"> - NN - GA 	<ul style="list-style-type: none"> - Increase the Detection Rate (DTR). - higher accuracy for R2L, U2R, and DoS attacks. 	
Chowdhury et al. (2016)	<ul style="list-style-type: none"> - SVM 	<ul style="list-style-type: none"> - Accuracy reach 98.76% 	<ul style="list-style-type: none"> - The lack of features to improve more efficiency
Almseidin et al. (2018)	<ul style="list-style-type: none"> - "Random Forest" - "Random Tree" - "Decision Table" - "MLP" - "Naive Bayes" - "Bayes Network" 	<ul style="list-style-type: none"> - Accuracy rate reach 93.77% for the Random forest 	<ul style="list-style-type: none"> - It can't increase the efficiency of the different types of attacks using a specific algorithm.
Alsawy et al. (2018)	<ul style="list-style-type: none"> - Random Forest - Multi-Layer Perceptron (MLP) - SVM 	<ul style="list-style-type: none"> - High accuracy reach (99.6%) - enhance the performance by applying the normalizing and scaling data 	<ul style="list-style-type: none"> - The complexity of runtime

Alomari et al. (2018)	- Decision Tree	- Detection rate reaches 98.5%	- Weakness with a small dataset.
Fattahi et al. (2018)	- ANN - SVM	- improve the complexity of the performance.	- The lack of classification for the supervised and unsupervised techniques
(Biswas, 2018)	- "Naive Bayes" - "Support Vector Machine" - "Decision Tree" - "Neural Network" - "KNN"	- Increase the combination of IGR "feature selection" using KNN.	- The lack of getting high efficiency using all techniques of features selections

Table 2.1 shows the different studies that solve the intrusion detection problem using different techniques. In the proposed approach will use the CICIDS2017 dataset, and we will use the bat algorithm for applying extracting features and building the filtered and refined dataset. In the proposed approach, we will be comparing the expected results with previous works and will try to reduce the weakness points in the previous works to gain high accuracy in intrusion detection over the networks.

2.3 CICIDS2017 Dataset

In this section, we will show the latest research work on intrusion detection using the CICIDS2017 dataset, and showing the latest threats, attacks, and features.

CICIDS2017 dataset contains the labeled network flows, where it classifies into benign class and up-to-date common attacks. The dataset used a PCAP format for loads of packet, and the CSV format that used with ML methods, where the PCAP file format is defined as a standard for capturing packets of network data. In the proposed approach, the original (raw) CICIDS2017 dataset will be used. The CICIDS2017 dataset contains eight different files containing the traffic data for attacks and five normal days of the Canadian Institute of Cybersecurity. Table 2.3 shown a description of dataset files.

Panigrahi and Borah (2018) explored the elaborate features of the "CICIDS2017" dataset and the related attributes between these features. They also present a mutual dataset through experiments and eliminating such attributes and features for better classification and detection using the different IDs. They also relabel the dataset based on the Canadian Institute of Cybersecurity, which it considers the reason for reducing the class imbalance issue.

Boukhamla and Coronel (2018) describe and optimize the CICIDS2017 dataset. They tried to reduce the dimensionality of the attributes and records using

the "Principal Component Analysis (PCA)" for the increasing the efficacy for the different processes, without losing the important features that contain more sensitivity of results. They evaluate the optimized dataset using three different classifiers ("KNN, C4.5, and Naïve Bayes"). The results of the evaluation shown that the optimized dataset maintains the same specificity and sensitivity comparing with the original dataset, and giving a better and faster IDS validation.

Table 2.2: Network Flow Analyzer (Panigrahi, and Borah, 2018).

Files	Ext.	Day	Attacks Found					
File1	CSV	Monday	Benign					
File 2	CSV	Tuesday	Benign	"FTP-Patator"	"SSH-Patator"			
File 3	CSV	Wednesday	Benign	Heartbleed	DoS Hulk	DoSslowloris	DoSSlowht tptest	DoS Golde nEye
File 4	CSV	Thursday	Benign	"Web Attack – Brute Force"	"Web Attack SQL Injection"	"Web Attack – XSS"		
File 5	CSV	Thursday	Benign	Infiltration				
File 6	CSV	Friday	Benign	Bot				
File 7	CSV	Friday	Benign	PortScan				
File 8	CSV	Friday	Benign	DDoS				

After merging the files presented in the previous table, and removing some of the missing instances, Panigrahi, and Borah (2018) extracted the characteristics of the combined dataset as shown in Table 2.4. The different labels and the number of instances for the "CICIDS2017 " dataset are shown in Table 2.5.

Table 2.3: Overall characteristics of the CICIDS2017 dataset.

The Name of Dataset	"CICIDS2018"
The Type of Dataset	"Multi-class"
Version	"2017"
Number of instances	"2830540"
Feature Count	"83"
Classes Count	"15"

Table 2.4: Labels and instances for the "CICIDS2017".

Class Labels	Number of instances
"BENIGN"	"2359087"
"DoS Hulk"	"231072"
"PortScan"	"158930"
"DDoS"	"41835"
"DoS GoldenEye"	"10293"
"FTP-Patator"	"7938"
"SSH-Patator"	"5897"
"DoS slowloris"	"5796"
"DoS Slowhttpstest"	"5499"
"Bot"	"1966"
"Web Attack – Brute Force"	"1507"
"Web Attack – XSS"	"652"

For more details, Appendix (A) contains a table and shown the extracted features definition for the most features in the CICIDS2017 dataset.

2.4 Bat Algorithm

Optimization problems are very common to be found in many different applications (Tsai, 2015), in the proposed approach, we are using the bat algorithm and extend it to increase the efficacy of multi-objective optimization problems, because of the high efficiency for bat algorithm for dealing with complex problems, and the ability for reducing run-time during using huge dataset. The proposed "multi-objective bat algorithm (MOBA)" is first validated against a subset of test functions (Yang, 2012), and then applied to solve multi-objective design problems. In our approach, we will use the bat algorithm as a feature selection method to get accurate results. We aim to get efficient results using the Micro-bats type of bat, which it depends on echolocation.

2.4.1 Echolocation of Micro-bats

In the proposed approach, the "bat algorithm" structure depends on the behavior of echolocation for the micro-bats with the different pulse rates of emission and loudness, echolocation is consisting of two steps:

1. Emitting sound pulse.
2. Detecting surrounding objects from the reflected echo.

Where each virtual bat flies randomly with a velocity \mathbf{V}_i at position \mathbf{X}_i with frequency \mathbf{F}_{\min} and loudness \mathbf{A}_0 , it changes frequency \mathbf{F}_{\max} , loudness \mathbf{A}_{\max} and pulses emission rate \mathbf{r} where $\mathbf{r} \in [0,1]$ (Yang, 2012).

2.4.2 Bat Motion

The rules of bats movement with specific positions \mathbf{X}_i to other point and their velocities \mathbf{V}_i for multi iterations is modeled as shown in equations 1, 2, 3:

$$f_i = f_{\min} + (f_{\max} - f_{\min}) \beta \quad (1)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) f_i \quad (2)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (3)$$

Where $\beta \in [0,1]$ is an empirical parameter working as a random vector drawn from a uniform distribution, \mathbf{t} is the number of iterations, and \mathbf{x}^* is the best solution of all the previous iterations. For selection in local search for one solution (best solution of the current iteration):

$$\mathbf{X}_{\text{new}} = \mathbf{X}_{\text{old}} + \mathbf{E} \mathbf{A}^t \quad (4)$$

Where \mathbf{E} is a random number vector drawn from $[-1, 1]$.

2.4.3 Loudness and Pulse Emission

The update of loudness and pulse rate is working accordingly as the iterations proceed, when the bat found its prey, the loudness will decrease automatically and its chosen as any value of convenience, but the rate of pulse emission will increase.

$$A_i^{t+1} = a + A_i^t \quad (5)$$

2.5 Intrusion Detection Using Bat Algorithm

Alina, Adryana, and Valentin (2015) proposed an approach that aims to reach the intelligent feature selection method for IDs using bat algorithm and the NSL-KDD dataset. In their approach, they worked that combines two ML algorithms with an improved version of the Binary "Bat Algorithm". The experiment results show the ability of features with almost 60% and obtain good results in terms of attack detection rate and false alarm rate, even for unknown attacks.

In this thesis, the proposed approach aims to build IDs using a bat algorithm to extract all possible and appropriate features from a big dataset and to increase the accuracy comparing with Alina et al. (2015) study.

Chapter Three

Research Methodology

3.1 Overview

This chapter presents the proposed methodology that will be used for the intrusion detection model using a bat algorithm. The first step in method aims to extract the feature selection that selects a representative set of attributes from the set of original attributes (using big data dataset). Where the representative set will keep only the relevant and important attributes from the raw dataset. The next phase in the proposed approach will be for using the classifier which categorizes unseen patterns in suitable classes (Chen, Huang, Tian and Qu, 2009).

3.2 General Framework

Figure 3.1 represents the architecture and the general framework for the proposed approach.

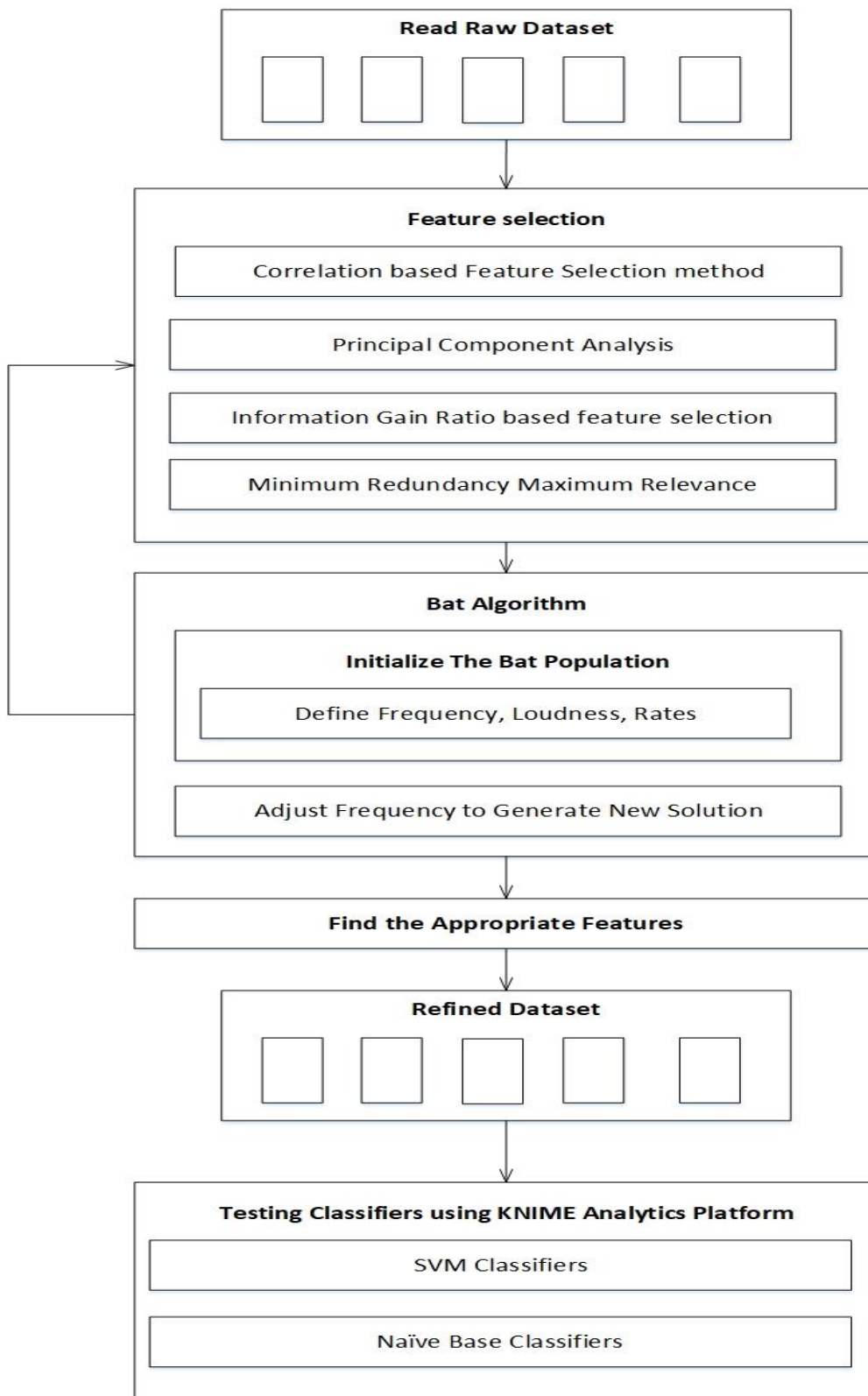


Figure 3.1: General framework

3.3 Dataset

In the proposed approach, we will use the CICIDS2017 dataset, which considering as a big data and containing the classes for the network flow, using the "PCAP" format, and the CSV files for ML methods. The PCAP file format is defined as a standard for capturing packets of network data. In the proposed approach, the original (raw) CICIDS2017 dataset will be used. The CICIDS2017 dataset contains eight different files containing the traffic data for attacks and five normal days of the Canadian Institute of Cybersecurity.

3.4 Methodology Phases

Our proposed methodology consists of the following phases:

1. Read Raw Dataset
2. Feature selection:
 - "Correlation-based feature selection method".
 - "Principal Component Analysis".
 - Information Gain Ratio based feature selection.
 - Minimum redundancy maximum relevance.

3. Bat Algorithm:

- Initialize the bat population.
- Define Frequency, Loudness, Rates.
- Adjust Frequency to Generate New Solution

4. Find the Appropriate Features

5. Refined Dataset.

6. Testing Classifiers using the KNIME Analytics Platform.

3.4.1 Dataset Acquisition (Raw)

This phase aims to use and apply the original dataset of the CICIDS2017 dataset as we downloaded. The proposed method will extract useful features using the dataset pre-processing phase.

3.4.2 Feature Selection

Feature selection is the set of techniques that are aimed at reducing the complexity of the dataset by eliminating some of the non-descriptive attributes. It's working to select all the representative set of attributes from the set of raw attributes (raw dataset). Where this representative set is working to keeps only the relevant and important attributes. In the proposed approach, we will use several techniques ("Correlation-based Feature Selection method, Principal Component

Analysis, Information Gain Ratio based feature selection, Minimum Redundancy Maximum Relevance") with bat algorithm for selecting the features from raw dataset to gain more facilitates data visualization and data understanding (Hamid et al. 2016).

3.4.3 Working of the Bat Algorithm

Optimization problems are very common to be found in many different applications (Tsai, 2015). In this proposed approach, we will use a bat algorithm as classifier method for improving the IDs performance vastly, and extend it to increase the performance for the multi-objective optimization problems.

3.4.4 Testing Classifiers using KNIME Analytics Platform

Testing classifiers using the KNIME Analytics Platform, the proposed approach will use the KNIME Analytics Platform for testing the refined dataset generated from the testing dataset using the Matlab algorithm. In this phase, the proposed approach will use two classifiers (SVM and Navie Base) for testing the accuracy for selecting features based on the bat algorithm. The Naive Bayes classifier is a typical generative classifier, which can be regarded as a special case of Bayesian network classifiers. While the SVM classifier is a typical discriminative classifier. Different from the generative classifier, it mainly focuses

on how well they can separate the positives from the negatives and does not try to understand the basic information of the individual classes (Shin, and Liu, 2011).

3.5 Data Analysis and Interpretation

In the proposed approach, we will use four of the common information retrieval evaluation metrics using the KNIME Analytics Platform, as following:

- Precision (Pr) or Positive Predictive value: It is the ratio of correctly classified attacks flows (TP), in front of all the classified flows (TP+CF).

$$\text{Pr} = \frac{TP}{TP + CF} \quad (6)$$

- Recall (Rc) or Sensitivity: In information retrieval, recall is the fraction of the relevant documents that are successfully retrieved. In the proposed approach, it is the ratio of correctly classified attack flows (TP), in front of all generated flows (TP+FN).

$$\text{Rc} = \frac{TP}{TP + Fn} \quad (7)$$

- F-Measure (F1): It is a hybrid combination of precision and recalls into one measure.

$$F1 = \frac{2}{\frac{1}{Pr} + \frac{1}{Rc}} \quad (8)$$

- The percentage of correct classification (PCC).

$$PCC = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

Where TP is the number of samples which is well classified as normal, TN is the number of samples which is well classified as Intrusion, FP is the number of samples classified as Intrusion but they were normal, and FN is the number of samples classified as Normal but they were attacks.

3.6 Research Tools

Research tools that will be used in this research are the following:

- Matlab for expatiating features.
- KNIME Analytics Platform for testing the features.

Chapter Four

Results and Discussions

4.1 Overview

In this section, we will show the experiment results for the proposed approach, which is divided into two phases as testing and training.

Using MATLAB code, we used the feature selections to get the appropriate features from the dataset using a bat algorithm, and then we worked with the KNIME tool for testing the results and accuracy for the bat model. MATLAB phase will generate a new Excel file that contains the new features generated from the bat algorithm.

4.1 Working with KNIME Tool

KNIME (Konstanz Information Miner) is a modular computational environment, which allows developers to get easily visual assembly, interactive data analysis, and data processing (Feltrin, 2015). It is an open-source predictive analytics platform (released under the GNU General Public License v3) suited to process a variety of data formats, from basic CSV or XLSX files to more complex

data structures such as XML, URL and relational databases (e.g., db2, Oracle, MySQL).

In our approach, we build a model for testing the accuracy using KNIME using two different classifiers (Naive Bayes and SVM) as shown in figure 4.1:

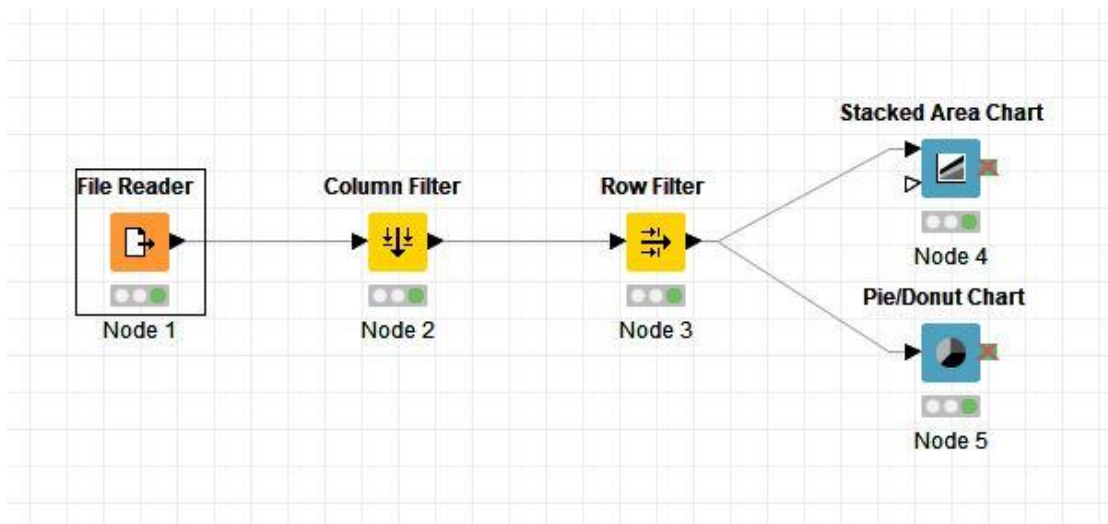


Figure 4.1: KNIME Model

In file reader node, we select the generated file from bat algorithm, to test the accuracy for the results, we can see the data by clicking right click on file reader and select file table option, the inserted data will be as it is shown in the figure below:

Row ID	Destina...	Flow D...	Total F...	Total B...	Total L...	Total L...	Fwd Pa...	Fwd Pa...	Fwd Pa...	Fwd Pa...	Bwd Pa...	Bwd Pa...	Bwd Pa...	Bwd Pa...	Flow By...	Flow Pa...	Flow
Row0	54865	3	2	0	12	0	6	6	6	0	0	0	0	0	4,000,000	666,666.667	3
Row1	55054	109	1	1	6	6	6	6	6	0	6	6	6	0	110,091.743	18,348.624	109
Row2	55055	52	1	1	6	6	6	6	6	0	6	6	6	0	230,769.231	38,461.538	52
Row3	46236	34	1	1	6	6	6	6	6	0	6	6	6	0	352,941.177	58,823.529	34
Row4	54863	3	2	0	12	0	6	6	6	0	0	0	0	0	4,000,000	666,666.667	3
Row5	54871	1022	2	0	12	0	6	6	6	0	0	0	0	0	11,741.683	1,956.947	1,022
Row6	54925	4	2	0	12	0	6	6	6	0	0	0	0	0	3,000,000	500,000	4
Row7	54925	42	1	1	6	6	6	6	6	0	6	6	6	0	285,714.286	47,619.048	42
Row8	9282	4	2	0	12	0	6	6	6	0	0	0	0	0	3,000,000	500,000	4
Row9	55153	4	2	0	37	0	31	6	18.5	17.678	0	0	0	0	9,250,000	500,000	4
Row10	55143	3	2	0	37	0	31	6	18.5	17.678	0	0	0	0	12,300,000	666,666.667	3
Row11	55144	1	2	0	37	0	31	6	18.5	17.678	0	0	0	0	37,000,000	2,000,000	1
Row12	55145	4	2	0	37	0	31	6	18.5	17.678	0	0	0	0	9,250,000	500,000	4
Row13	55254	3	3	0	43	0	31	6	14.333	14.434	0	0	0	0	14,300,000	1,000,000	1.5
Row14	36206	54	1	1	0	0	0	0	0	0	0	0	0	0	37,037.037	54	54
Row15	53524	1	2	0	0	0	0	0	0	0	0	0	0	0	2,000,000	1	1
Row16	53524	154	1	1	0	0	0	0	0	0	0	0	0	0	12,987.013	154	154
Row17	53526	1	2	0	0	0	0	0	0	0	0	0	0	0	2,000,000	1	1
Row18	53526	118	1	1	0	0	0	0	0	0	0	0	0	0	16,949.153	118	118
Row19	53527	239	1	1	0	0	0	0	0	0	0	0	0	0	8,368.201	239	239
Row20	53528	1	3	0	0	0	0	0	0	0	0	0	0	0	3,000,000	0.5	0.5
Row21	53527	1	2	0	0	0	0	0	0	0	0	0	0	0	2,000,000	1	1
Row22	55035	4	2	0	248	0	217	31	124	131.522	0	0	0	0	62,000,000	500,000	4
Row23	55275	5	3	0	254	0	217	6	84.667	115.284	0	0	0	0	50,800,000	600,000	2.5
Row24	55277	4	2	0	12	0	6	6	6	0	0	0	0	0	3,000,000	500,000	4
Row25	8850	4	3	0	43	0	31	6	14.333	14.434	0	0	0	0	10,800,000	750,000	2
Row26	43248	54	1	1	0	0	0	0	0	0	0	0	0	0	37,037.037	54	54
Row27	8678	42	3	0	43	0	31	6	14.333	14.434	0	0	0	0	1,023,809.524	71,428.571	21
Row28	55063	4	2	0	37	0	31	6	18.5	17.678	0	0	0	0	9,250,000	500,000	4
Row29	55203	3	2	0	37	0	31	6	18.5	17.678	0	0	0	0	12,300,000	666,666.667	3
Row30	55140	3	2	0	37	0	31	6	18.5	17.678	0	0	0	0	12,300,000	666,666.667	3
Row31	55180	737	2	1	37	6	31	6	18.5	17.678	6	6	6	0	58,344.64	4,070.556	368.5
Row32	55156	3	2	0	37	0	31	6	18.5	17.678	0	0	0	0	12,300,000	666,666.667	3
Row33	55096	3	2	0	37	0	31	6	18.5	17.678	0	0	0	0	12,300,000	666,666.667	3
Row34	55085	3	2	0	37	0	31	6	18.5	17.678	0	0	0	0	12,300,000	666,666.667	3
Row35	8689	276	3	0	43	0	31	6	14.333	14.434	0	0	0	0	155,797.101	10,869.565	138
Row36	8817	3	2	0	12	0	6	6	6	0	0	0	0	0	4,000,000	666,666.667	3
Row37	8816	3	2	0	12	0	6	6	6	0	0	0	0	0	4,000,000	666,666.667	3

Figure 4.2: File Table

Figure 4.2 shows all features selected using the bat algorithm in the refined dataset phase, this data will be used in the testing phase using system classifiers (SVM, and Naive Base). In the next step, we should identify the pattern matching for the inserted data. The patterns are categorized in the MATLAB phase based on the class of attack (Label):

- FTP-Patator
- Bot
- Web Attack

- PortScan
- DDoS
- BENIGN

The figure below shown how selecting the pattern matching:

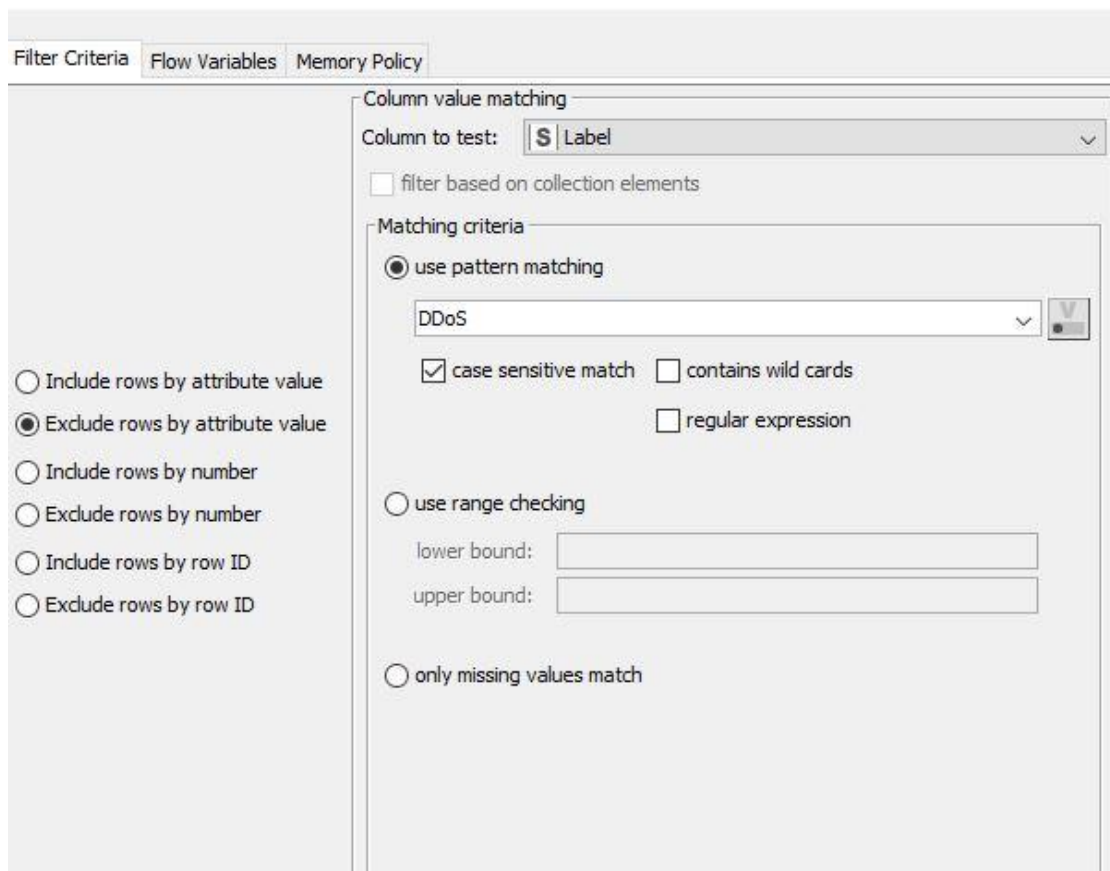


Figure 4.3: Pattern Matching

Using the Stacked Area Chart node, which it considers as a chart for visualizes numerical values from multiple columns as stacked areas. Different

stacking types can be chosen to guide the user's interpretation: The node can display Stacked Area Charts, Percentage Area Charts, and Stream Graphs. Different interpolation methods can be chosen to give the graph an organic appearance.

figure 4.4 shows the visualizes data for the proposed model using the KNIME for testing the bat algorithm results:

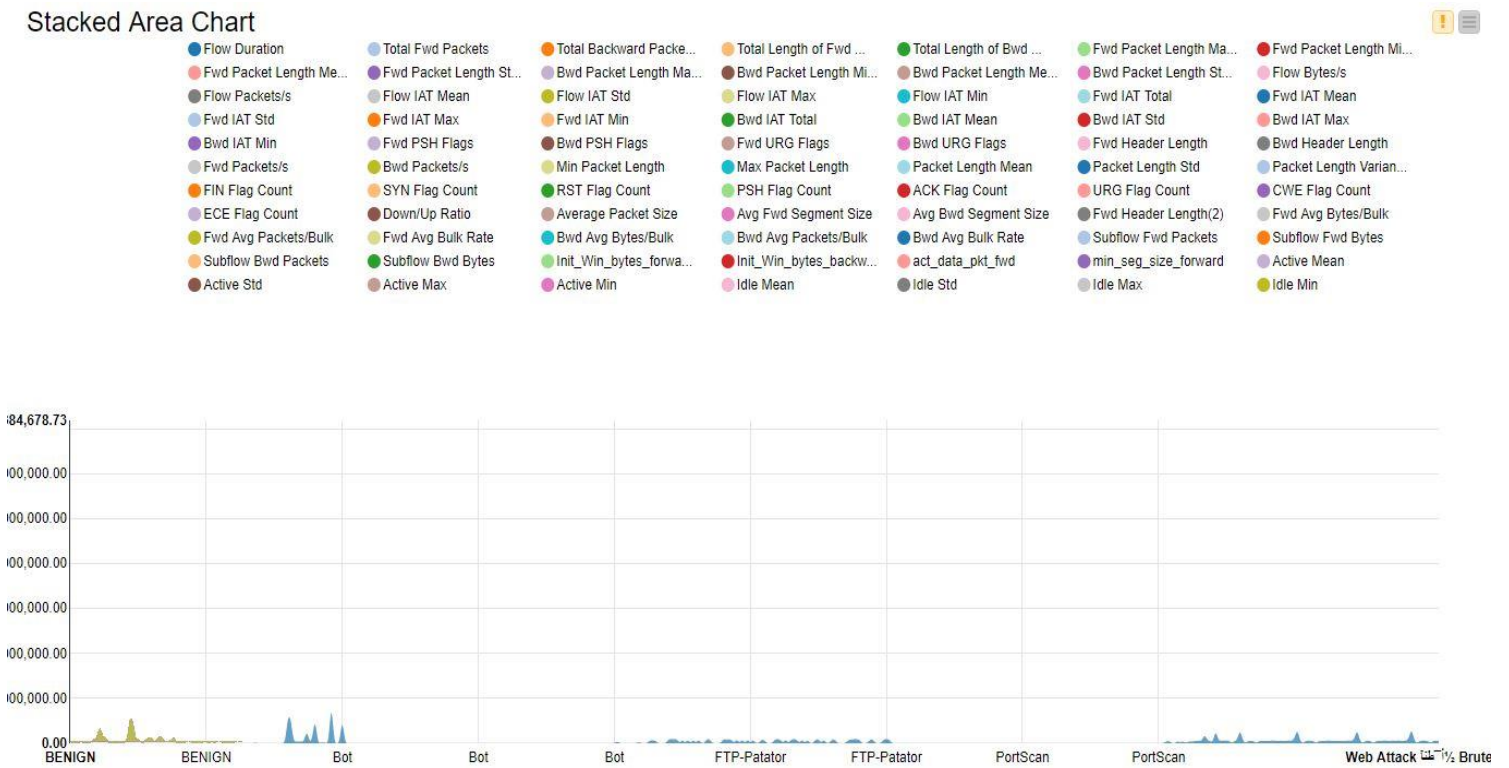


Figure 4.4: The visualizes data

The previous figure shows all features selected using the bat algorithm with the different classes of attacks.

According to the previous model, table 4.1 shown the final results for testing the accuracy, correct classified, wrong classified, and the error using the SVM classifier:

Table 4.1: SVM Classifier Results

	FTP-Patator	Bot	Web Attack	PortScan	DDoS	BENIGN
FTP-Patator	59	0	0	0	0	0
Bot	0	60	0	0	0	0
Web Attack	0	0	61	0	0	0
PortScan	0	0	0	60	0	0
DDoS	0	0	0	0	61	0
BENIGN	1	3	1	0	2	1

As we see from the previous figure, the proposed KNIME model gets a high accuracy (97.52%) for the testing dataset using the SVM classifier, which means the features selected from the bat algorithm gives a high efficiency with reducing the error (2.47%) for the SVM classifier. While the table below shows the results for testing the Bat algorithm using the Naive Bayes classifier:

Table 4.1: Naïve Base Classifier Results

	FTP-Patator	Bot	Web Attack	PortScan	DDoS	BENIGN
FTP-Patator	59	0	0	0	0	0
Bot	0	59	0	0	0	2
Web Attack	0	0	54	0	0	7
PortScan	0	0	0	61	0	0
DDoS	0	0	0	0	61	0
BENIGN	1	0	0	0	1	58

As we see from the previous table, the proposed KNIME model gets a high accuracy (96.97%) for the testing dataset, which means the features selected from the bat algorithm gives a high efficiency with reducing the error (3.03%).

To evaluate the proposed model, we test the final results for the two classifiers (Naive Bayes and SVM) and comparing it with Alina et al. (2015) results using the same classifiers based on the performance measures:

- **"Attack Detection Rate (ADR)":** the ability of the proposed model to classify the alarms for the different intrusions.
- **"False Alarm Rate (FAR)":** the number of errors for classifying alarms for the normal intrusions.
- **"Nb. of Features":** the number of features and attributes for the generated dataset.

Results from table 4.1 and table 4.2 shown a comparing between Alina et al. (2015) approach for the two classifiers (Naive Bayes and SVM):

Table 4.1: Comparing results using Naive Bayes classifier

"Algorithm"	"NB"	"ADR"	"FAR"	"Time"
"BBAL"	53	91.62	5.73	764
"BBA"	70	89.73	7.24	793
"BPSO"	80	89.44	7.86	829
"Simple NB"		90.53	6.66	1019
"Proposed Approach"	77	96.97	3.03	681

Table 4.2: Comparing results using SVM classifier

"Algorithm"	"NB"	"ADR"	"FAR"	"Time"
"BBAL"	4	95.78	2.89	68768
"BBA"	6	95.03	3.28	78251
"BPSO"	10	94.03	4.01	80726
"Simple NB"		89.64	6.88	82603
"Proposed Approach"	65	97.52	2.47	60421

Figure 4.7 shows the comparison for ADR measure with the Alina et al. (2015) study using the Naive Bayes classifier. Where figure showed the preference of results for the proposed approach compared with other approaches.

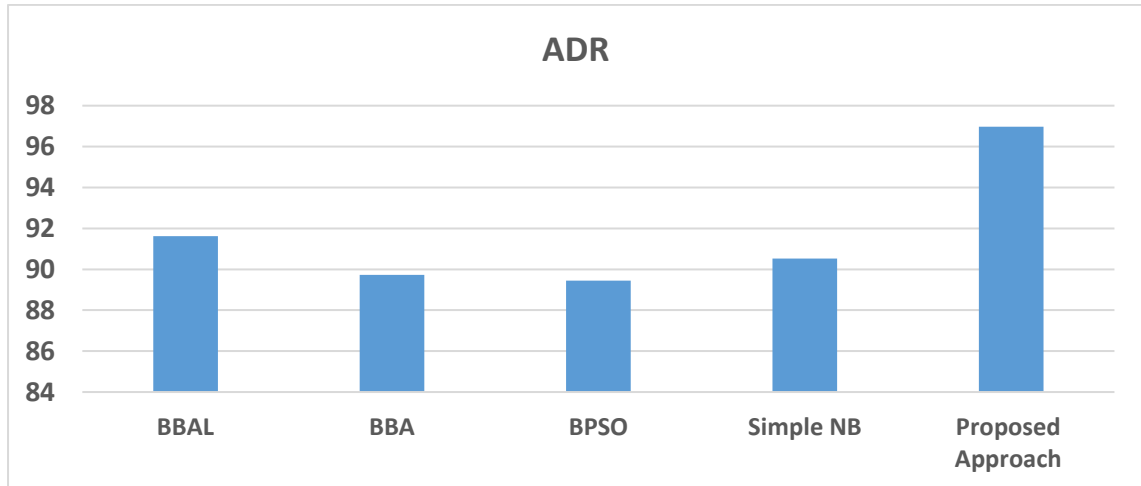


Figure 4.7: The comparison ADR with Naive Bayes classifier

Figure 4.8 shows the comparison for FAR measure with the Alina et al. (2015) study using the Naive Bayes classifier. Where the figure showed the best

results with minimum errors for the proposed approach comparing with other approaches.

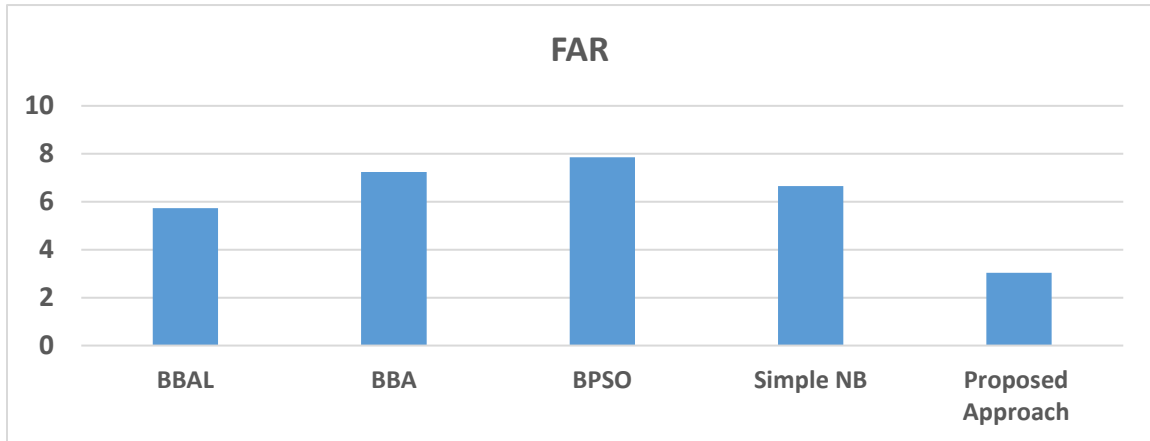


Figure 4.8: The comparison FAR with Naive Bayes classifier

The figure below showed the run time for the different measures using the Naive Bayes classifier, where the proposed approach needs the minimum run time comparing with other measures.

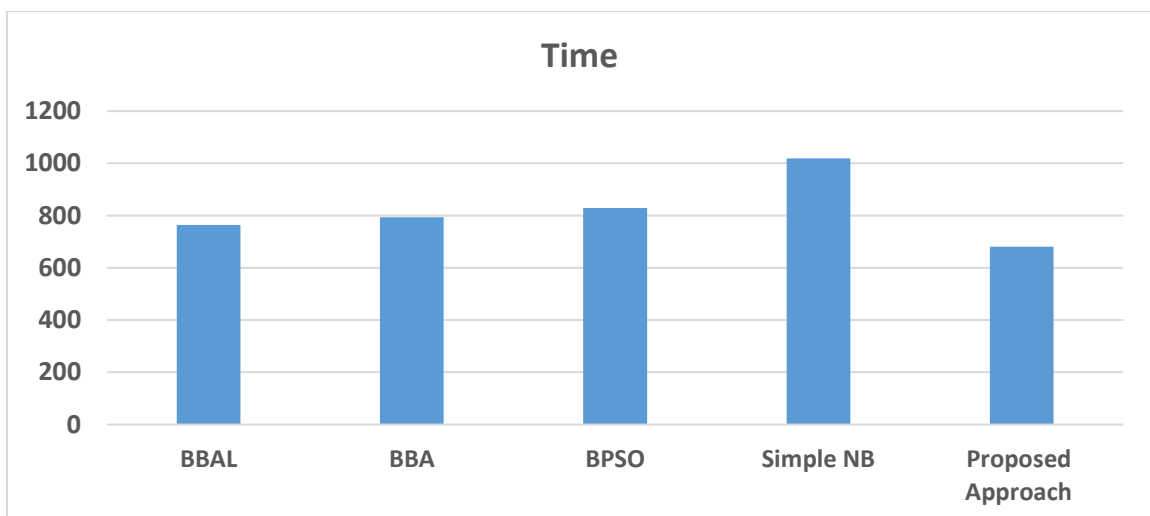


Figure 4.9: The comparison Time with Naive Bayes classifier

Figure 4.10 shows the comparison for ADR measure with the Alina et al. (2015) study using the SVM classifier. Where figure showed the preference of results for the proposed approach compared with other approaches.

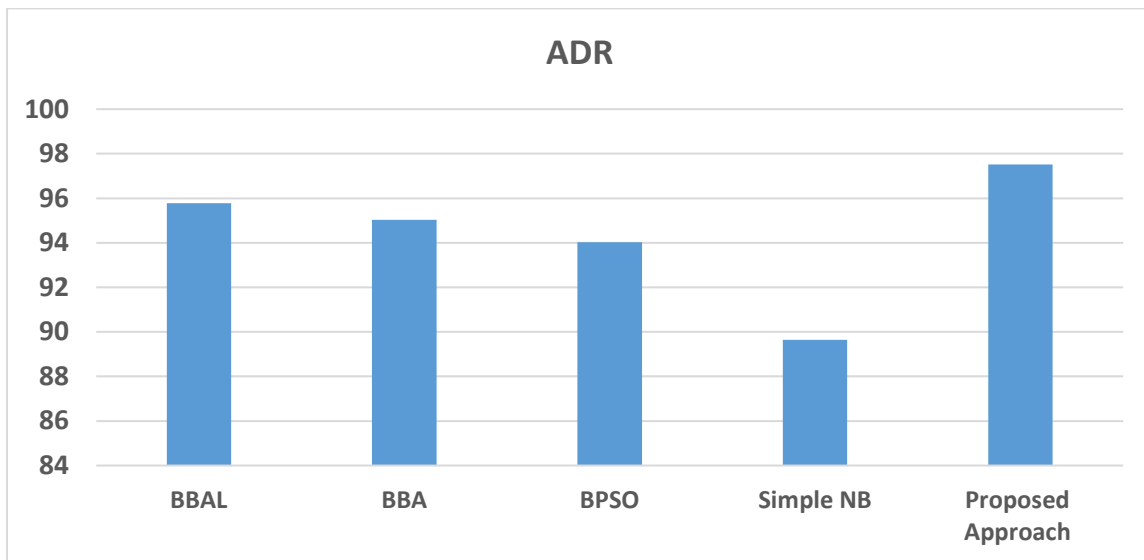


Figure 4.10: The comparison ADR with SVM classifier

Figure 4.11 shows the comparison for FAR measure with the Alina et al. (2015) study using the SVM classifier. Where the figure showed the best results with minimum errors for the proposed approach comparing with other approaches.

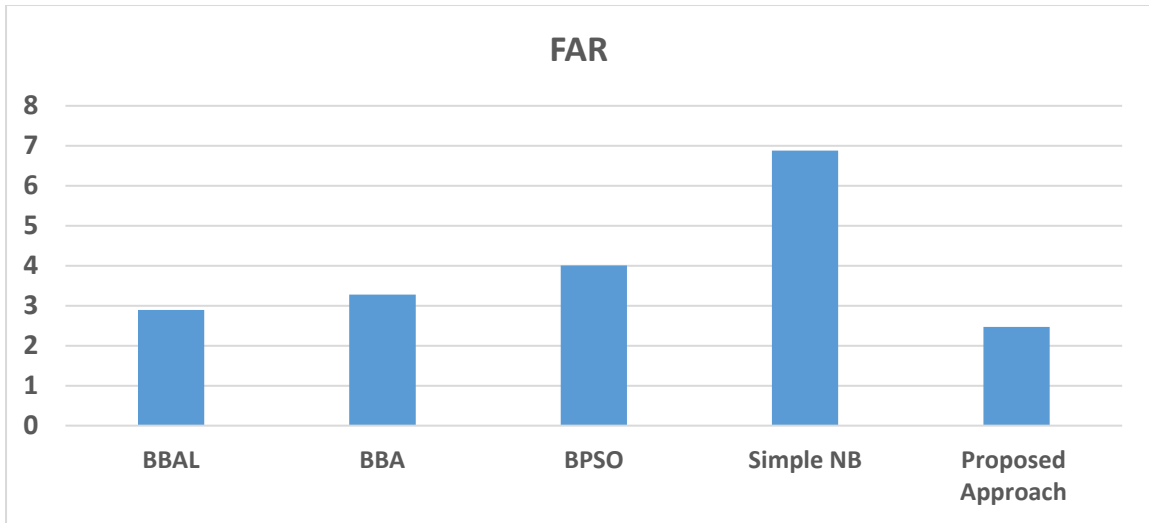


Figure 4.11: The comparison FAR with SVM classifier

The figure below showed the run time for the different measures using the SVM classifier, where the proposed approach needs the minimum run time comparing with other measures.

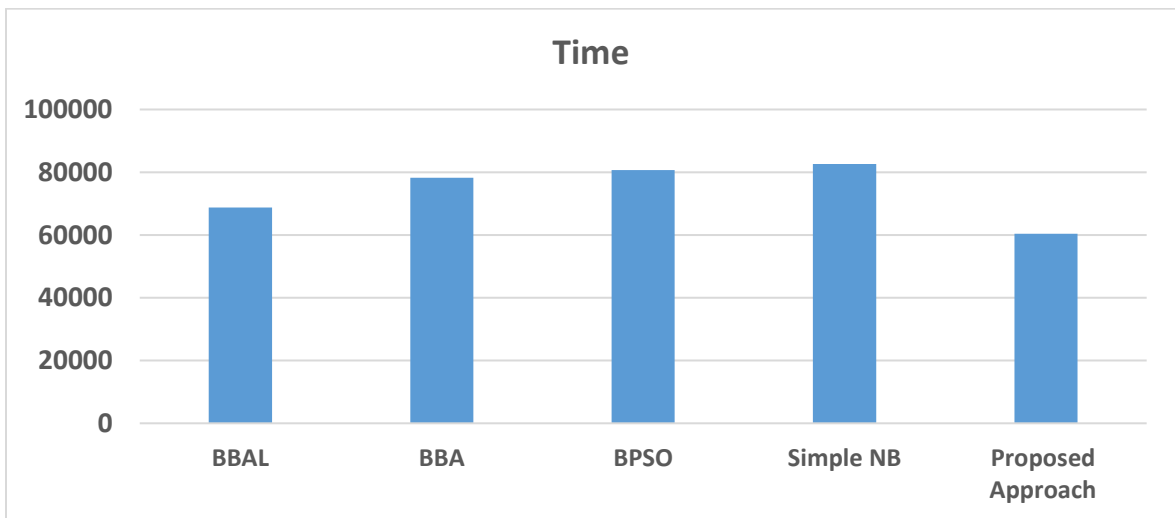


Figure 4.12: The comparison Time with Naive Bayes classifier

Chapter Five

Conclusions and Future Work

In this chapter, the conclusions, achievements, and the recommended future works were proposed.

5.1 Conclusions

Finally, we built an efficiency model of incursion detection using a bat algorithm to extract the apocopate features from the raw dataset. The method was tested using the KNIME tool based on two classifiers (SVM, and Naïve Base), and the experiment results give a high accuracy (97.52%) with reducing the error classification into (2.47%) comparing with Alina et al. (2015) study, that it reached (95.78%) in the best cases. We can notice also, the classifier SVM gives high accuracy results comparing to the Naïve Base classifier, that considered as a contribution to the proposed methodology for enhancing the efficiency and accuracy in the intrusion detection field.

5.2 Future Work

In future work, we hope to apply and reach the below:

- Increasing the scalability for the system with newer technologies and methods in this sector.
- Developing new algorithms that working as a system assessment, and considering as risk management tools to reduce the possible errors and bugs.
- Upgrade new scenarios using different algorithms working in artificial intelligence, aims to increase the accuracy for intrusion detection.

REFERENCES

Chen, J., Huang, H., Tian, S., and Qu, Y. (2009). Feature selection for text classification with Nave Bayes. *Expert Systems with Applications*, vol. 36, issue 3, pp. 54325435.

Panigrahi, P., & Borah, S. (2018). A detailed analysis of the CICIDS2017 dataset for designing Intrusion Detection Systems. *International Journal of Engineering & Technology*, (3.24) (2018) 479-482.

Hamid, H., Sugumaran, M., and Journaux, L. (2016). Machine Learning Techniques for Intrusion Detection: A Comparative Analysis. *Conference: The International Conference*, DOI: 10.1145/2980258.2980378.

Feltrin, L. (2015), KNIME an Open Source Solution for Predictive Analytics in the Geosciences [Software and Data Sets], *IEEE Geoscience and Remote Sensing Magazine*, 10.1109/MGRS.2015.2496160.

Raheem, A., and Alomari, E. (2018). An Adaptive Intrusion Detection System by using a decision tree. *Journal of AL-Qadisiyah for computer science and mathematics*. Vol.10 No.2.

Boukhamla, A., and Coronel, J. (2018) CICIDS2017 Dataset: Performance Improvements and Validation as a Robust Intrusion Detection System Testbed. *International Journal of Information and Computer Security*.

Panigrahi, P., & Borah, S. (2018). A detailed analysis of the CICIDS2017 dataset for designing Intrusion Detection Systems. *International Journal of Engineering & Technology*. (3.24) (2018) 479-482.

- Biswas, S.K. (2018). Intrusion Detection Using Machine Learning: A Comparison Study. *International Journal of Pure and Applied Mathematics*. Volume 118 No. 19.
- Fattahi, J., Omrani, T., Dallali, A., & Rhaimi, B. (2018). A fusion of ANN and SVM Classifiers for Network Attack Detection. arXiv:1801.02746v2 [cs.CR] 10 Jan 2018.
- Almseidin, M., Alzubi, M., Kovacs, S., and Alkasassbeh, M. (2018). Evaluation of Machine Learning Algorithms for Intrusion Detection System. *International Symposium on Intelligent Systems and Informatics (SISY)*.1949-0488.
- Chowdhury, M. N., Ferens, H., and Ferens, M. (2016). Network Intrusion Detection Using Machine Learning. *Int'l Conf. Security and Management*.
- Karpagam, S., Revathi, T., and Jayakumar, K. (2015). Intrusion Detection using Artificial Neural Networks with Best Set of Features. *The International Arab Journal of Information Technology*. Vol. 12, No. 6A.
- Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *4th International Conference on Information Systems Security and Privacy (ICISSP)*, Portugal, January 2018.
- Zamani, M. (2013). Machine Learning Techniques for Intrusion Detection. *CoRR*, abs/1312.2177.
- Heberlein, L.T., Dias, G.V., Levitt, K.N., Mukherjee, B., Wood, J., Wolber, D. (1990). A network security monitor. *In Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, USA, 7–9 May 1990; pp. 296–304.

Levitt, K., Heberlein, L., and Mukherjee, B. (1994). Network Intrusion Detection. *IEEE Network* May/June 1994.

Akhilesh Kumar Shrivastava, A. K. (2014). An Ensemble Model for Classification of Attacks with Feature Selection based on KDD99 and NSL-KDD Data Set. *International Journal of Computer Applications*.

Haq, N. F. et al. (2015). Application of Machine Learning Approaches in Intrusion Detection System: A Survey. (*IJARAI*) *International Journal of Advanced Research in Artificial Intelligence*. Vol. 4, No.3.

Juma, S., Muda, Z., Mohammad, M. A., and Yasin, W. (2015). Machine Learning Techniques for Intrusion Detection: A Review. *Journal of Theoretical and Applied Information Technology*. Vol.72 No.3.

Lang, B., & Liu, H. (2019). Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *International Journal of Engineering & Technology*. (3.24) (2018) 479-482.

Anderson, J.P. Computer Security Threat Monitoring and Surveillance, Technical Report; James P. Anderson Company: Philadelphia, PA, USA, 1980.

Mohamed, H., Hefny, H., and Alsawy, A. (2018). Intrusion Detection System Using Machine Learning Approaches. *Egyptian Computer Science Journal*. Vol. 42 No.3.

Kumar, E. A., Kaur, J., and Kaur, I. (2016). Intrusion Detection System by Machine Learning Review. *International Journal of Advanced Research, Ideas, and Innovations in Technology*.

Chen, J., Huang, H., Tian, S., and Qu, Y. (2009). Feature selection for text classification with Naive Bayes. *Expert Systems with Applications*. vol. 36, issue 3, pp. 5432-5435.

Can, O., & Sahingoz, O. (2015). A survey of intrusion detection systems in wireless sensor networks. 10.1109/ICMSAO.2015.7152200.

Alina, P., Adryana, E., and Valentin, S. (2015). Intelligent feature selection method rooted in Binary Bat Algorithm for intrusion detection. 10.1109/SACI.2015.7208259.

Niyaz, Q., Sun, W., Javaid, A., and Alam, M. (2016). A Deep Learning Approach for Network Intrusion Detection System.

Shi H., Liu Y. (2011) Naïve Bayes vs. Support Vector Machine: Resilience to Missing Data. In: Deng H., Miao D., Lei J., Wang F.L. (eds) Artificial Intelligence and Computational Intelligence. AICI 2011. Lecture Notes in Computer Science, vol 7003. Springer, Berlin, Heidelberg.

Srivastava, D., and Dahiya, P. (2018). Network Intrusion Detection in Big Dataset Using Spark, International Conference on Computational Intelligence and Data Science (ICCIDS 2018), 132 (2018) 253–262.

APPENDIX A

Feature Name	State In	Description
"F_Fe_duration"	"Microsecond "	"Duration of the flow"
"total_FWwd_Pkt"	"forward direction "	"Total packets"
"total_Bwd_Pkt"	"backward direction"	"Total packets"
"total_Length_of_Fwd_Pkt"	"forward direction"	"Total size of packet"
"total_Length_of_Bwd_Pkt"	"backward direction"	"Total size of packet"
"Fwd_Pkt_Length_Min"	"forward direction"	"The minimum size of a packet"
"Fwd_Pkt_Length_Max"	"forward direction"	"Maximum size of a packet"
"Fwd_Pkt_Length_Mean"	"forward direction"	"Mean size of the packet"
"Fwd_Pkt_Length_Std"	"forward direction"	"Standard deviation size of the packet"
"Bwd_Pkt_Length_Min"	"backward direction"	"The minimum size of the packet "
"Bwd_Pkt_Length_Max"	"backward direction"	"Maximum size of the packet"
"Bwd_Pkt_Length_Mean"	"backward direction"	"Mean size of the packet"
"Bwd_Pkt_Length_Std"	"backward direction"	"Standard deviation size of the packet"
"F_Byte/s"	"per second"	"Number of flow packets"
"F_Pkt/s"	"per second"	"Number of flow bytes"
"F_IAT_Mean"	"flow"	"Meantime between two packets sent"
"F_IAT_Std"	"flow"	"Standard deviation time between two packets sent"
"F_IAT_Max"	"flow"	"Maximum time between two packets sent"
"F_IAT_Min"	"flow"	"Minimum time between two packets sent"
"Fwd_IAT_Min"	"forward direction"	"Minimum time between two packets sent"
"Fwd_IAT_Max"	"forward direction"	"Maximum time between two packets sent"
"Fwd_IAT_Mean"	"forward direction"	"Meantime between two packets sent"
"Fwd_IAT_Std"	"forward direction"	"Standard deviation time between two packets sent"
"Fwd_IAT_Total"	"forward direction"	"Total time between two packets sent"
"Bwd_IAT_Min"	"backward direction"	"Minimum time between two packets sent"
"Bwd_IAT_Max"	"backward direction"	"Maximum time between two packets sent"
"Bwd_IAT_Mean"	"backward direction"	"Meantime between two packets sent"
"Bwd_IAT_Std"	"backward direction"	"Standard deviation time between two packets sent"
"Bwd_IAT_Total"	"backward direction"	"Total time between two packets sent"
"Fwd_PSH_FLG"	"forward direction (0: UDP) "	"Number of times the PSH flag was set in packets traveling"
"Bwd_PSH_FLG"	"backward direction (0: UDP) "	"Number of times the PSH flag was set in packets traveling"
"Fwd_URG_FLG"	"forward direction (0: UDP) "	"Number of times the URG flag was set in packets traveling"
"Bwd_URG_FLG"	"backward direction (0: UDP)"	"Number of times the URG flag was set in packets traveling"
"Fwd_Hdr_Length"	"forward direction"	"Total bytes used for headers"
"Bwd_Hdr_Length"	"backward direction"	"Total bytes used for headers"
"FWD_Pkt/s"	"per second"	"Number of forwarding packets"
"Bwd_Pkt/s"	"per second"	"Number of backward packets"
"Min_Pkt_Length"	"packet"	"Minimum length"

"Max_Pkt_Length"	"packet"	"Maximum length"
"Pkt_Length_Mean"	"packet"	"Mean length"
"Pkt_Length_Std"	"packet"	"Standard deviation length"
"Pkt_Length_Variance"	"packet"	"Variance length"
"FIN_FLG_Count"	"FIN"	"Number of packets"
"SYN_FLG_Count"	"SYN"	"Number of packets"
"RST_FLG_Count"	"RST"	"Number of packets"
"PSH_FLG_Count"	"PUSH"	"Number of packets"
"ACK_FLG_Count"	"ACK"	"Number of packets"
"URG_FLG_Count"	"URG"	"Number of packets"
"CWR_FLG_Count"	"CWE"	"Number of packets"
"ECE_FLG_Count"	"ECE"	"Number of packets"
"down_Up_Ratio"	"ratio"	"Download and upload "
"Average_Pkt_S"	"packet"	"Average size"
"Avg_Fwd_Segment_S"	"forward direction"	"Average size observed"
"AVG_Bwd_Segment_S"	"forward direction"	"The average number of bytes bulk rate"
"Fwd_Hdr_Length"	"forward packet"	"Length of header"
"Fwd_Avg_Bytes_Bulk"	"forward direction"	"The average number of bytes bulk rate"
"Fwd_AVG_Pkt_Bulk"	"forward direction"	"The average number of packets bulk rate"
"Fwd_AVG_Bulk_Rate"	"forward direction"	"The average number of bulk rate"
"Bwd_Avg_Bytes_Bulk"	"backward direction"	"The average number of bytes bulk rate"
"Bwd_AVG_Pkt_Bulk"	"backward direction"	"The average number of packets bulk rate"
"Bwd_AVG_Bulk_Rate"	"backward direction"	"The average number of bulk rate"
"SubF_Fwd_Pkt"	"forward direction"	"The average number of packets in a sub-flow"
"SubF_Fwd_Bytes"	"forward direction"	"The average number of bytes in a sub-flow"
"SubF_Bwd_Pkt"	"backward direction"	"The average number of packets in a sub-flow"
"SubF_Bwd_Bytes"	"backward direction"	"The average number of bytes in a sub-flow"
"Init_Win_bytes_forward"	"forward direction"	"The total number of bytes sent in the initial window"
"Init_Win_bytes_backward"	"backward direction"	"The total number of bytes sent in the initial window"
"Act_data_pkt_forward"	"forward direction"	"Count of packets with at least 1 byte of TCP data payload"
"min_seg_S_forward"	"forward direction"	"Minimum segment size observed"
"Active_Min"	"flow"	"Minimum time active before becoming idle"
"Active_Mean"	"flow"	"Meantime active before becoming idle"
"Active_Max"	"flow"	"Maximum time active before becoming idle"
"Active_Std"	"flow"	"Standard deviation time active before becoming idle"
"Idle_Min"	"flow"	"Minimum time idle before becoming active"
"Idle_Mean"	"flow"	"Meantime idle before becoming active"
"Idle_Max"	"flow"	"Maximum time idle before becoming active"
"Idle_Std"	"flow"	"Standard deviation time idle before becoming active"
"total_fPkt"	"forward direction"	"Total packets"
"total_bPkt"	"backward direction"	"Total packets"
"total_fpktl"	"forward direction"	"The total size of the packet"
"total_bpktl"	"backward direction"	"The total size of the packet"
"min_fpktl"	"forward direction"	"The minimum size of the packet"
"min_bpktl"	"backward direction"	"The minimum size of the packet"

"max_fpctl"	"forward direction"	"Maximum size of the packet"
"max_bpctl"	"backward direction"	"Maximum size of the packet"
"mean_fpctl"	"forward direction"	"Mean size of the packet"
"mean_bpctl"	"backward direction"	"Mean size of the packet"
"std_fpctl"	"forward direction"	"Standard deviation size of the packet"
"std_bpctl"	"backward direction"	"Standard deviation size of the packet"
"total_fiat"	"forward direction"	"Total time between two packets sent"
"total_biat"	"backward direction"	"Total time between two packets sent"
"min_fiat"	"forward direction"	"Minimum time between two packets sent"
"min_biat"	"backward direction"	"Minimum time between two packets sent"
"max_fiat"	"forward direction"	"Maximum time between two packets sent"
"max_biat"	"backward direction"	"Maximum time between two packets sent"
"mean_fiat"	"forward direction"	"Meantime between two packets sent"
"mean_biat"	"backward direction"	"Meantime between two packets sent"
"std_fiat"	"forward direction"	"Standard deviation time between two packets sent"
"std_biat"	"backward direction"	"Standard deviation time between two packets sent"
"fpush_cnt"	"forward direction (0: UDP) "	"Number of times the PSH flag was set in packets traveling"
"bpush_cnt"	"backward direction (0: UDP) "	"Number of times the PSH flag was set in packets traveling"
"furg_cnt"	"forward direction (0: UDP) "	"Number of times the URG flag was set in packets traveling"
"burg_cnt"	"backward direction (0: UDP)"	"Number of times the URG flag was set in packets traveling"
"total_fhlen"	"forward direction"	"Total bytes used for headers"
"total_bhlen"	"backward direction"	"Total bytes used for headers"
"fPktPerSecond"	"per second"	"Number of forwarding packets"
"bPktPerSecond"	"per second"	"Number of backward packets"
"FPktPerSecond"	"per second"	"Number of flow packets"
"FBytesPerSecond"	"per second"	"Number of flow bytes"
"min_Fpctl"	"flow"	"Minimum length"
"max_Fpctl"	"flow"	"Maximum length"
"mean_Fpctl"	"flow"	"Mean length"
"std_Fpctl"	"flow"	"Standard deviation length"
"min_Fiat"	"packet"	"Minimum inter-arrival time"
"max_Fiat"	"packet"	"Maximum inter-arrival time"
"mean_Fiat"	"packet"	"Mean inter-arrival time"
"std_Fiat"	"packet"	"Standard deviation inter-arrival time"
"F_fin"	"FIN"	"Number of packets"
"F_syn"	"SYN"	"Number of packets"
"F_rst"	"RST"	"Number of packets"
"F_psh"	"PUSH"	"Number of packets"
"F_ack"	"ACK"	"Number of packets"
"F_urg"	"URG"	"Number of packets"
"F_cwr"	"CWE"	"Number of packets"
"F_ece"	"ECE"	"Number of packets"

"downUpRatio"	"ratio"	"Download and upload"
"avgPktize"	"packet"	"Average size "
"fAvgSegmentS"	"forward direction"	"Average size observed"
"fAvgBytesPerBulk"	"forward direction"	"The average number of bytes bulk rate"
"fAvgPktPerBulk"	"forward direction"	"The average number of packets bulk rate"
"fAvgBulkRate"	"forward direction"	"The average number of bulk rate"
"bAvgSegmentS"	"backward direction"	"Average size observed"
"bAvgBytesPerBulk"	"backward direction"	"The average number of bytes bulk rate"
"bAvgPktPerBulk"	"backward direction"	"The average number of packets bulk rate"
"bAvgBulkRate"	"backward direction"	"The average number of bulk rate"
"sF_fPkt"	"forward direction"	"The average number of packets in a sub-flow"
"sF_fbytes"	"forward direction"	"The average number of bytes in a sub-flow"
"sF_bPkt"	"backward direction"	"The average number of packets in a sub-flow"
"sF_bbytes"	"backward direction"	"The average number of bytes in a sub-flow"
"min_active"	"flow"	"Minimum time active before becoming idle"
"mean_active"	"flow"	"Meantime active before becoming idle"
"max_active"	"flow"	"Maximum time active before becoming idle"
"std_active"	"flow"	"Standard deviation time before becoming idle"
"min_idle"	"flow"	"Minimum time idle before becoming active"
"mean_idle"	"flow"	"Meantime idle before becoming active"
"max_idle"	"flow"	"Maximum time idle before becoming active"
"std_idle"	"flow"	"Standard deviation time idle before becoming active"
"Init_Win_bytes_forward"	"forward direction"	"The total number of bytes sent in the initial window"
"Init_Win_bytes_backward"	"backward direction"	"The total number of bytes sent in the initial window"
"Act_data_pkt_forward"	"forward direction"	"Count of packets with at least 1 byte of TCP data payload"
"min_seg_S_forward"	"forward direction"	"Minimum segment size observed"