A STUDY ON THE EFFECTS OF FEATURE SETS ON INTRUSION DETECTION SYSTEM

YOEKLENG KUY

A THESIS REPORT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING IN COMPUTING IN ENGINEERING SYSTEMS
INTERNATIONAL COLLEGE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2017
KMITL-2017-IC-M-11-06

A STUDY ON THE EFFECTS OF FEATURE SETS ON INTRUSION DETECTION SYSTEM

YOEKLENG KUY

A THESIS REPORT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING IN COMPUTING IN ENGINEERING SYSTEMS
INTERNATIONAL COLLEGE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2017
KMITL-2017-IC-M-11-06

THESIS TITLE	A Study on the Effects of Feature Sets on Intrusion Detection System
STUDENT NAME	Mr. Yoekleng Kuy
STUDENT ID	59610057
DEGREE	Master of Engineering
PROGRAME	Computing in Engineering Systems (International Program)
ADVISOR	Dr. Isara Anantavrasilp

Abstract

In this decade, malicious threats are the most concerning challenge in computer networks or the Internet. The mechanism of these threats are quite advanced and complicated to evade themselves from the detection mechanism of computer network protection layers. Thus, machine learning have been actively introduced into the detection mechanisms of the protection layer in such behavior-based IDS. Mainly, behavior-based IDS employs machine learning algorithms to learn on behavior characteristics or "features" of those threats. However, finding a proper set of features and the best perform classifier are still an opening issue in term of detection accuracy performance.

This thesis is to reveal the impacts of feature sets on intrusion detection system performances. The core idea is to exploit which is the smallest feature sets should be employed, then compares the strengths of each algorithm of the different background approaches to obtain a robust classifier. We extracted 41 features from a well-known intrusion detection dataset KDD. Then, ten best feature sets were selected by ReliefF. Next, the subset of datasets were divided according to combination among best ten features which are approximately 1023 sets. After that, five different machine learning algorithms such as decision tree (J48), sequential rule covering (JRip), Naïve Bayes (NB), back-propagation artificial neural network (ANN) and support vector machines (SVM) are employed. We split 50% of data for training set and remaining 50% for test set. Later, the experimental results obtained and investigated. This study reveals that **7-{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}** set of features and J48 classifier should be employed.

Acknowledgments

I am deeply owing a debt of gratitude to my supervisor Dr. Isara Anantavrasilp, who spends his precious time to instruct, advise, encourage and correct throughout this work. His course “Intelligent Network” provides deeply understand and develop my knowledge on this subject area as well this research area.

Next, I would love to express my gratitude to all lecturers and staffs at International College of King Mongkut’s Institute of Technology Ladkrabang (IC-KMITL), who support within this two years. Moreover, I would like to say thank to all my friends who always there to make good memory and fun all together.

Finally, I do express great gratitude to my parents, sister and brothers for providing me with unfailing support and continuous motivation throughout many years of study. I can say that this accomplishment would not been achieved without the best support from them all.

Yoekleng Kuy

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Nowadays, Internet has played as an integral part of everyone's live. All type of organizations such as individual business, education institution, medical center, mobile operator, financial sector, and public sector rely on the computer networks, or especially the Internet to boot up daily operation processes and services. For example, university may provide Internet access and online administrative services to students and staffs. As a result, their students and staffs can surf the Internet for their studying and working purposes or even access to the online administrative portals. The other example is digital payments or Internet-banking services. Users can pay bills easily with just a few click on their devices, whereas they do not need to pay by the cash or go to pay at the stores or banks directly. It is so convenience for end-users, but it would be harder for computer-network administrators because there are vast amounts of digital transactions are processed, shared, collected and stored.

In the way that large amounts of transaction data are generated, the protection of this data as well as the network infrastructures from unauthorized person or malicious activities is a very important task. Computer-network security experts have come up with several useful security mechanisms namely firewall, anti-virus, and intrusion detection system. The unique concept of this security mechanism is to safeguard the computer systems and networks. By its protection role, firewall blocks or filters the connections between the inside and outside zone of hosts or networks, while anti-virus acts as a protection-shield on the computing devices, which prevents those devices from

harmful software such virus or malware. Meanwhile, intrusion detection system (IDS) is added to monitor on the computer hosts and networks in order to detect malicious threats or policy-violated activities.

In general, intrusion detection system is available for both computer host and network, with respect to two major detection methods: signature-based and behavior-based. Signature-based detects malicious activities by matching that activities to signature database. If the malicious activities occur and its signatures exist in the database, IDS can detect it as known threats. However, the signatures must be specified into the IDS beforehand. This detection method may not able to detect the unknown threats. In contrast, behavior-based does not depend on known the signatures, but this detection method rather observes the behavior activities that occur on the computer hosts or networks. Mainly, this method detects the malicious behaviors by inspecting characteristics or "features". This method is more adaptive with the aiding of machine learning algorithms, which could allow IDS to capture different kind of behaviors without human intervention.

1.2 Problem Statement

Unfortunately, the spreading of malicious threats are quite rapid and sophisticated. Moreover, those malicious threats are certainly harmful to computer hosts and networks due to they could evade themselves from traditional detection mechanisms. While, an adaptive behavior-based IDS employs machine learning algorithms to learn from features and classify those malicious threats, the searching for proper set of features and robust classifier are still the concerning problem in term of performance detection accuracy.

Therefore, numerous of studies were proposed in [1–13]. Several of them were focused on the classifier by measuring the performance detection accuracy. While, the other several employed the different set of features or used feature selection algorithms to measure the discriminative of those features. Additionally, one another was presented the set of features and the accuracy performance of classifier. However, their set of features were still broadly, not the smallest one. To the best of our background knowledge, a detail study of relationship between the smallest sizes of features and performance classification accuracy is still missing. In the absence of such knowledge, we might waste time to compute too many features without improving the accuracy rates.

1.3 Objective

To address this concerning, a study on the effects of feature sets for intrusion detection system is presented. Specifically, our goal is to figure out the smallest features and classifier that should be employed.

Key contributions are:

- This thesis proposes set of features for classifying malicious threats.
- Finally, the experiment results are analyzed and shown what is the smallest feature sets and classifier should be employed.

1.4 Thesis Structure

The remaining parts of this thesis are organized into following chapters:

- Chapter 2 describes the overview of the different types of IDS and its detection methods. Then, the review of related works are described.
- Chapter 3 describes about the proposed method including dataset, feature selection, feature combination, machine learning algorithms, and measurement technique.
- Chapter 4 shows the experimental setup and results.
- Chapter 5 reveals the remark conclusion and direction for future work.

Chapter 2

Literature Review

2.1 IDS Types

Intrusion detection system is a device or software application that are used to strengthen the security of hosts and networks communication systems. IDS is a useful tool because without such tool, it would be more difficult for computer-network administrators to analyze or trace on the huge amount of transaction data traversing through the hosts or network systems. IDS is categorized into two main groups, host-based and network-based, which are briefly described below.

2.1.1 Host-based IDS

As its name suggested, a host-based intrusion detection system [14] is software package that is installed for an individual host computer. Host-based IDS monitors system state and analyzes traffic of the host where it is installed. There are several elements that host-based IDS can monitor in such network resource usages, user logs, system logs, file systems, and memory processes [14].

The benefit of host-based IDS are: since it is placed on end-point device, it has ability to access to full payload of the communication packets to that host, even if that communication is encrypted [14]. And most of malicious activities have valid header and malicious content in the packets payload. Thus, host-based IDS can analyze the payload to detect malicious actions. Usually, host-based IDS is used to protect mission-critical servers like database and web server, which is shown in Figure 2.1. As an example, Alien Vault [15] is one of commercial product of host-based IDS

which installs on the critical servers. Alien Vault could provide deep visibility of what's happening on the critical servers [15].

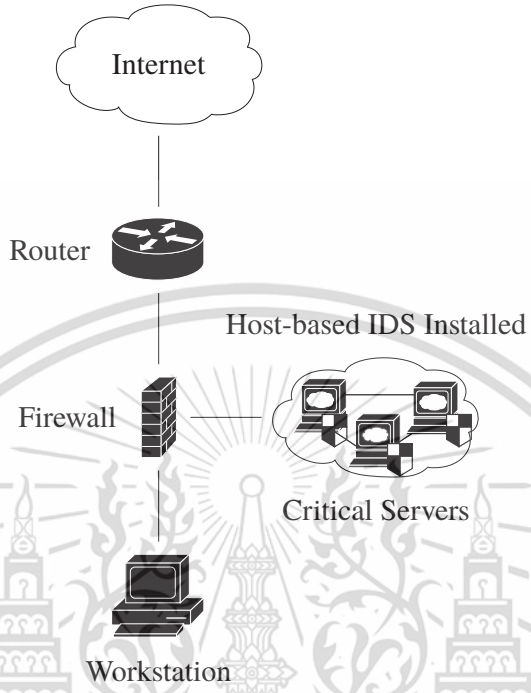


Figure 2.1: Host-based IDS Diagram

The drawback of host-based IDS is: it protects only a host where it installs on [14]. What if we need to protect wide range of host systems in network infrastructures, host-based IDS must install on each host system. If so, this task would be handy and take time.

2.1.2 Network-based IDS

Alternatively, network-based intrusion detection system [14] is available to fulfill the incompletion of host-based IDS. Network-based IDS scrutinizes related activities such as traffic volume, IP address, service port, protocol usage, etc. As an example, network-based IDS could monitor activities flows on the entire subnet and match that flows to its detection engine. Once an attack traffic is discovered, or malicious behavior activity is identified, then network-based IDS signals the alert to the screen of computer-network administrator.

Network-based IDS can deploy on the strategic points where traffic of the entire network can visibly monitor. However, we need to care about connection bandwidth to avoid the bottleneck prob-

lem. Richardson [16], discussed about the strategic points where network-based IDS is deployed. For example, network-based IDS can deploy between gateway router and firewall or firewall and switch, as shown in Figure 2.2. There are various network-based IDS appliances. Snort [17] is one of well-known network-based IDS appliance which is an open source and capable to use on both large and small networks.

Figure 2.2: Network-based IDS Diagram

2.2 Detection Methods

There are two major detection methods for IDS namely signature-based and behavior-based . IDS uses signature-based detection method is known as “Signature-based IDS”. IDS uses behavior-based detection method is known as the “behavior-based IDS”.

2.2.1 Signature-based IDS

Signature-based IDS detects a malicious action based on “signature” in the database which configures either locally or remotely. The signature-based IDS works like this: if the malicious actions occur and its signatures exist in the database, then it can mark that actions as known threats. A signature is a string, a command, a binary sequence, or a pattern which is created from the information of known vulnerability. For example, “Red Code” worm [18] is a malicious software that spread itself via the vulnerability of Microsoft IIS Web servers. Red Code contains a signature as indicated in Figure 2.3.

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801
%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3
%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0
```

Figure 2.3: Red Code’s Signature taken from [18]

The advantage of this detection method provides high accuracy rate of detecting known threats. However, there are several pain points in signature-based method. First, the signature rules must be predefined into the system in advance. Second, what if the attackers know clearly which pattern the IDS is inspecting into, then attackers can easily modify the properties of the malicious contents to evade the security mechanism of this IDS. Third, the signatures must update persistently. Thus, the signature-based detection method may not be able to deal with unknown threats, unless the unknown threats signatures are built or updated into its database.

2.2.2 Behavior-based IDS

Behavior-based intrusion detection system is come to fulfill this incompletion. Behavior-based IDS has been introduced in [1]. Behavior-based IDS do not rely on known the signatures, but rather observes on the connection flow activities. Principally, behavior-based IDS detects the malicious behaviors by inspecting the characteristics or "features" of activities that are occurring in host systems and networks. The characteristics or features can be number of simultaneous connections from or to the same hosts or ports, average packet sizes, flow lengths, etc. In regards of this features, behavior-based IDS can classify whether the observed flows are malicious. Behavior-based IDS could also classify the connection flows into normal or anomaly. Behavior-based IDS that classifies the observed flows into such classes, is known as anomaly-based IDS [2].

A normal is a common user connection flow. An anomaly is the malicious flow or unsure flow. A malicious flow is a dangerous connection that has characterized or intended to do harm on computer hosts and networks. While, an unsure flow is a suspicious connection that could be a normal flow or malicious flow. Thus, it is left to network administrators to take further steps to determine whether the unsure flow are threats or not. In addition, machine learning techniques have integrated into behavior-based IDS, which enables numerous of researchers to conduct their studies on this field [1–13]. Mainly, behavior-based IDS employs machine learning algorithms such decision tree, sequential rule covering, etc., to build behavioral model. Behavior-based IDS could detect unseen behaviors and classifies unknown flows automatically without human intervention.

Moreover, behavior-based IDS could classify the observed flows into different categories or classes as well. The different classes are based on the purposes such as security purpose, quality-of-service (QoS) purpose, etc, [1]. The example of different type of classes are:

- Security types: Normal or Anomaly, Threat or Non-threat etc,.
- Application types: Skype, Bit-torrent, Video Streaming, Real-time or Bulk, etc,.
- Protocol types: HTTP, SMTP, SSH, Telnet, etc,.

2.3 Related Works

For decades, intensive of research works have been reported on this research domain, several works are sorted out which importantly related to our work.

Garcia-Teodoro et al. [2] conducted a review study of the well-known behavior-based or anomaly-based IDS detection approaches. Then, available platforms, systems under development and research projects in the domain of behavior-based IDS were discussed as well. Garcia-Teodoro et al. [2] have mentioned that intrusion detection techniques are continuously evolving, with the goal of improving the security and protection of computer infrastructures [2]. Early et al. [3] have introduced a novelty technique to classify the behavior on server flow application protocols. Traditionally, client-server protocol was determined by scrutinizing on the source and destination port numbers in the TCP header. Beside checking on port, Early et al. [3] addressed the behavioral authentication of server flows by finding what is characteristics or "features" that should be employed for observing the in-coming or out-going flow of servers. Early et al. [3] categorized flows into five major server application protocols such as HTTP, FTP, Telnet, SMTP and SSH due to these protocols are widely implemented and presented the large majority of user network traffic. In their experimental, two datasets were employed such DAPAR99 [4] and the data that obtained from their own network. Decision tree (C5.0) learning algorithm was used [3].

The combination approaches of decision tree and Naïve Bayes for behavior-based IDS or anomaly-based IDS was proposed by Benferhat and Tabia [5]. Authors chose these two classifiers because these two classifiers build model on different methodology backgrounds and have good complementarity/error diversity. Benferhat and Tabia [5] evaluated hybrid classifiers on multi-class KDD99 [4] dataset which contained normal flow and four main attacks namely DoS, R2L, U2R and Probe. Then, authors tried to reduce the false negative rates by reanalyzing the output of hybrid classifiers. Two sophisticated machine learning algorithms support vector machines (SVM) and back-propagation artificial neural network (ANN) were employed to classify network traffics into seven classes: FTP, WWW, P2P, NetBIOS, DNS, Mail and Telnet was presented by Pradhan [6]. Pradhan [6] captured a raw network data on his own network by using Wireshark tool and Perl script was used to derive flows. After that, SVM and ANN on WEKA 3.6.1 were employed to build behavior model on the 66% training set and evaluated on the remaining test set.

Pervez and Fraid [8], carried out a study of flow classification by employing feature selection and SVM on multi-class NSL-KDD [7] dataset. The experimental was run on machine, CPU Intel Core i7 2.7GHz and RAM 16GB. Pervez and Fraid [8] implemented their proposed method by employing library of SVM classifier from WEKA3 in JAVA programming language. Then, 10-fold cross validation was applied, nine sets used for training and the remaining one was test set. The process repeated 10 times and measured the performance of SVM classifier by the mean of accuracy rate. They selected different sizes of features; 3, 36, and 41 features, however authors did not show which feature in each size of feature set. The accuracy rates obtained from aforementioned feature sets were 91%, 99% and 99% respectively [8]. Although, authors evaluated their classifier on different feature sets, the number of feature in each set is still too broad and only one machine learning algorithm is employed. Another study of behavior-based IDS by using artificial neural network (ANN) was conducted on both binary-class and multi-class NSL-KDD dataset as well [9]. For ANN learning methods, authors employed Back-Propagation (BGF) and Levenberg-Marquardt (LM) in MATLAB. The machine, CPU Intel Core 2 Duo 2.2GHz and RAM 4GB was used to perform all underlying tasks. ANN trained the network based on the default training set of NSL-KDD and evaluated on default test set. Then, the classification accuracy rates were obtained. The accuracy rates of their classifier could yield 81.2% using 29 features against binary-class. On multi-class dataset could achieve 79.9% accuracy with 41 features [9].

A network anomaly traffic detection system integrated both Entropy Theory and Support Vector Machines (SVM) was presented in [10]. There are six features including source IP, destination IP, source port, destination port, packet size, and packet type were employed. The entropy theory was used to measure the network traffics or flows features, while SVM was used to detect the network anomaly by perform the classification task. Authors employed KDD99 dataset for their proposed method [10]. Miao et al. [11] proposed network traffic classification using six machine learning algorithms such as Naïve Bayes, Decision Tree (C5.0), Nearest Neighbor (NN), Random Forest, SVM and H_2O (deep learning). Principle Component Analysis (PCA) feature selection was employed to reduce the dimension of feature spaces in their work as well. The performance of all six classifiers with PCA and without PCA were investigated base on accuracy rates [11]. Boger et al. [12] employed K-means, an unsupervised machine learning algorithm, to classify network traffic. Their dataset was obtained raw packet traces from the National Collegiate Cybersecurity

Defense Competition (NCCDC). The experiments were run on machine, CPU quad core 2.4GHz and 4GB of RAM. K-means was run for K equals 4, 5, 6, 7, 8 and 9. They found that $K = 5$ provided the most natural of the dataset. Frank [13] carried out a survey of artificial intelligence (AI) methods which was used in anomaly-based IDS. Frank [13] also presented a use case of feature selection to improve the performance classification of network flows. Table 2.1, is shown the comparison of related works between methods and classes.

Table 2.1: The comparison of related works by proposed methods and classes

Previous Works	**Proposed Methods**	**Classes**
Early et al. [3],	Decision Tree (C5.0)	HTTP, FTP, SMTP, SSH and Telnet
Benferhat and Tabia [5]	Decision Tree and Naïve Bayes	Normal, DoS, U2R, R2L and Probe
Pradhan [6]	SVM and Back-propagation ANN	FTP, WWW, P2P, NetBIOS, DNS, Mail and Telnet
Pervez and Fraid [8]	Feature selection and SVM	Normal, DoS, U2R, R2L and Probe
Ingre and Yadav [9]	Back-propagation ANN	Normal, DoS, U2R, R2L and Probe

Chapter 3

Methodology

This chapter, we provide the detail information about the element of our proposed method. The main idea is to investigate on what is the smallest set of features and the learning algorithms that yield desired performance. This study consists of six main components namely, dataset, feature selection, features combination, learners, classifiers and results. The definitions are described as following:

- Dataset: is the data in format *.csv or *.arff file, contains the number of connection flows which are extracted from packet trace files.
- Feature selection: is the process of selecting feature with regard to the value of discriminability of that feature and along with the class. Selecting features of the flows requires deep domain knowledge, sometimes trial and error method is employed.
- Feature combination: is the process that separates sub-dataset according to the combination set of features that opted by feature selection.
- Learners: are the observer that observe on the data (training set). Then, learner could produce the model or patterns.
- Classifiers: are the evaluator that evaluate the model with unseen data (test set).
- Results: are the experimental result reports that are obtained based on classification accuracy rates. Last, these results are analyzed to find out which is the smallest feature sets and the best performance classifier.

Figure 3.1 illustrates components of our proposed method.

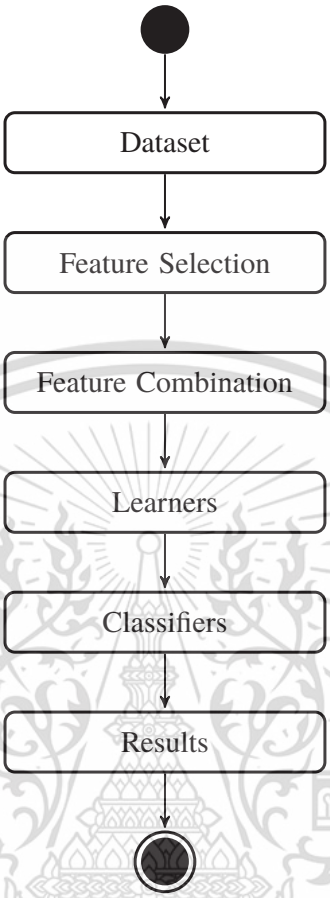


Figure 3.1: The components of proposed method

3.1 Dataset

A dataset is extracted from packet traces which contains the number of connection flows. The dataset is in format of *.csv (comma-separated values) or *.arff (attribute-relation file format). A dataset is the finite multi-set over $X \times C$, where X is the set of feature vectors and C is the set of classes [19]. Each element of dataset is called data instance or example. Notation D is the set of all datasets. The equation of dataset represents by:

$$D = \{\langle x_1, c_1 \rangle, \ldots, \langle x_m, c_m \rangle\} \tag{3.1}$$

such that, $\langle x_i, c_i \rangle \in X \times C$, and $1 \leq i \leq m$.

Basically, the dataset is divided into training set and test set. The training set are used to build the model for machine algorithms. Then, test set is for evaluating the model. In this thesis, NSL-KDD multi-class is employed. As a few revision of NSL-KDD multi-class dataset and for our convenience, we shorten the name NSL-KDD multi-class and call it as KDD. The brief description of datasets are provided in this section. In 1998, the original packet traces DARPA98 was simulated and captured by MIT Lincoln Laboratories [4]. Then, KDD99 [4] was extracted from DARPA98 packet traces in the competition of 3^{rd} international knowledge discovery and data mining tool.

In 2009, Tavallaee et al. [7] revised and refined KDD99 by removing redundant instances and the instances that contain missing value of features. This refined dataset is called NSL-KDD [7]. NSL-KDD available in two forms: binary-class and multi-class. Each instance in the dataset is labeled with a class and represented by 41 features. The features are categorized into three following groups:

- Basic TCP/IP features: is the individual TCP/IP connection as indicated in Table 3.1.
- Content features: is the content of individual connection is suggested by domain knowledge as indicated in Table 3.2.
- Traffic features: is computed using a two-second time window in a certain of connection as indicated in Table 3.3.

Table 3.1: Basic features of individual TCP connection were taken from [4]

No.	Feature	Description
1	duration	duration of connection
2	protocol_type	connection protocol i.e. TCP, UDP, ICMP
3	service	destination service i.e. HTTP, FTP, etc,.
4	flag	status flag of the connection
5	src_bytes	bytes sent from source to destination
6	dst_bytes	bytes sent from destination to source
7	land	1 if connection is from/to the same host/port; 0 otherwise
8	wrong fragment	# of wrong fragments
9	urgent	# of urgent packets

Table 3.2: Content features within a connection suggested by domain knowledge were taken from [4]

No.	Feature	Description
10	hot	# of “hot” indicators
11	num_failed_logins	# of failed logins
12	logged_in	1 if successfully logged in; 0 otherwise
13	num_compromised	# of “compromised” conditions
14	root_shell	1 if root shell is obtained; 0 otherwise
15	su_attempted	1 if “su root” command attempted; 0 otherwise
16	num_root	# of “root” access
17	num_file_creations	# of file creation operations
18	num_shells	# of shell prompts
19	num_access_files	# of operations on access control files
20	num_outbound_cmds	# of outbound commands in a ftp session
21	is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise
22	is_guest_login	1 if the login is a “guest” login; 0 otherwise

Totally, KDD contains 148,517 instances. The distribution of instances are shown in Table 3.4. KDD categorizes into four main type of attacks:

- Denial of Service (DoS): a kind of attack that the attacker tries to exhaust the network resources or services of the target system.
- User to Root (U2R): the attacker attempts to gain access to the system as a normal user, then exploit the vulnerability of systems to gain access as super users.
- Remote to Local (R2L): the attacker remotely sends packets to the target machine and tries to exploit the vulnerabilities to access victim machine.
- Probing (Probe): the attacker attempts to scan on the target systems and looks for weaknesses or vulnerabilities.

3.2 Feature Selection

Feature section is a technique that uses to eliminate dimensionality of feature spaces. In the dataset, there might be several features are redundant, non-discriminative or irrelevant which can be removed before fed into the classifier. Besides, some features might not comply with one another

Table 3.3: Traffic features computed using a two-second time window were taken from [4]

No.	Feature	Description
23	count	# of connections to the same host as the current connection in the past two seconds
24	srv_count	# of connections to the same service as the current connection in the past two seconds
25	serror_rate	% of connections that have “SYN” errors (same-host connections)
26	srv_serror_rate	% of connections that have “SYN” errors (same-service connections)
27	rerror_rate	% of connections that have “REJ” errors (same-host connections)
28	srv_rerror_rate	% of connections that have “REJ” errors (same-service connections)
29	same_srv_rate	% of connections to the same service
30	diff_srv_rate	% of connections to different services
31	srv_diff_host_rate	% of connections to different hosts
32	dst_host_count	count of connections having the same destination host
33	dst_host_srv_count	count of connections having the same destination host and using the same service
34	dst_host_same_srv_rate	% of connections having the same destination host and using the same service
35	dst_host_diff_srv_rate	% of different services on the current host
36	dst_host_same_src_port_rate	% of connections to the current host having the same src port
37	dst_host_ srv_diff_host_rate	% of connections to the same service coming from different hosts
38	dst_host_ serror_rate	% of connections to the current host that have an “S0” error
39	dst_host_ srv_serror_rate	% of connections to the current host and specified service that have an “S0” error
40	dst_host_ rerror_rate	% of connections to the current host that have an “RST” error
41	dst_host_ srv_rerror_rate	% of connections to the current host and specified service that have an “RST” error

Table 3.4: Number of KDD instances

Class	**# Instances**
Normal	77,054
DoS	53,385
U2R	252
R2L	3,749
Probe	14,077
Total	**148,517**

feature, so feature selection can help to depict this hidden correlation as well. Moreover, feature selection can be used to enhance the computational speed with minimum accuracy rate reduction. Feature selection can perform independently in preprocessing task. Meanwhile, machine algorithms identify flow behaviors based on characteristics or features of flows. By using feature selection could eliminate the several information-less features and increase the classifier performances. Truthfully, adding features into consideration might not always yield better classification accuracy and selecting the smallest set of features is imperative. In our work, we employed ReliefF [21] feature selection algorithm, to rank the best ten features which derive from 41 features in KDD dataset. The brief background of ReliefF [21] are described as below.

Kira and Rendell [20] proposed a feature selection algorithm known as Relief which is based on instance-based learning to update feature weight. Relief selected feature sets depend on a statistical method that does not rely on heuristics search. The complexity run-time is only linear time in the number of given features and the number of training data. However, it works only with binary-class problem. ReliefF [21] is an extension version of Relief and it is applicable on multi-class problems. ReliefF [21] was proposed by Kononenko. It was one of the most popular feature selection algorithms [22] and due to its high performance [23]. Therefore, it is used to select most discriminative sets of features. ReliefF algorithm attempts to compute and rank the weights of features in the case of multi-class problems. The higher the weight assigned to the feature, the more discriminative the feature [24].

Given the sampling instances $I_1, I_2, \ldots, I_n$ are described by a vector of feature A_i, where $i = 1, \ldots, a$ (a is the total number of explanatory features), and labeled with the target class C_j. Such that, the sampling instance $I_1, I_2, \ldots, I_n$ are the point in a-dimensional. As shown in Figure 3.2, initially, ReliefF sets all feature weights $W[A]$ to 0. Next, algorithm iteratively selects a ran-

Input: for each training instance a vector of feature attributes and the class value
Output: the vector W of estimations of the qualities of feature attributes

1: set all weights $W[A]$ = 0.0;
2: **for** $i = 1 \rightarrow m$ **do** begin
3: randomly select an instance R_i;
4: find k nearest hits H_j;
5: **for each** class $C \neq class(R_i)$ **do**;
6: from class C find k nearest misses $M_j(C)$;
7: **for** $A = 1 \rightarrow a$ **do**
8: $W[A] = W[A] - \sum_{j=1}^{k} diff(A, R_i, H_j)/(m \cdot k)$ +
9: $\sum_{c \neq class(R_i)} \left[\frac{P(C)}{1 - P(class(R_i))} \sum_{j=1}^{k} diff(A, R_i, M_j(C)) \right] /(m \cdot k)$;
10: **end**;

Figure 3.2: Pseudo code of ReliefF algorithm was taken from [24]

dom observation instance R_i, then: finds for k of its nearest neighbors from the same class, called "nearest hits" H_j and also k nearest neighbors from each of the different classes, called "nearest misses" $M_j(C)$. After that, ReliefF updates the quality estimation weight value $W[A]$ for all features A depending on their values for R_i, hits H_j and misses $M_j(C)$ (lines 7, 8 and 9). The update formula is similar to Relief [20], except that the contribution of all the hits and all the misses are averaged. The prior probability of that class $P(C)$ is applied to weigh the contribution for each class of the misses which estimated based on the training set [24]. The contributions of hits and misses in each step is range between $[0; 1]$ and also symmetric the misses' probability weights is sum to 1. Due to, the class of hits is missing in the sum, the dividing of each probability weight with factor $1 - P(class(R_i))$ [24]. The whole process is repeated for m times.

3.3 Feature Combination

After, feature selection is used to select ten most relevant features. Then, we apply the feature combination function which is represented by this formula:

$$C(n, k) = \frac{n!}{k!(n-k)!} \tag{3.2}$$

where,

- n is the total number of feature sets $n = 10$
- k is combination feature set such as 1-feature-set-combination, 2-features-set-combination, $\cdots$, n-features-set-combination $1 \leq k \leq n$.

3.4 Machine Learning Algorithms

Machine learning is one study area of artificial intelligence (AI) that empowers computer systems the ability to learn and improve from the experience without being explicitly programmed. In the main while, there are various applications of machine learning namely, pattern recognition, virtual personal assistant, search engine result refining, support decision making, and especially intrusion detection system. The primary task of machine learning is to train the computer systems to make prediction base on the given dataset, and then adjust or correct errors accordingly. Machine learning algorithms are separated into two main groups such as supervised and unsupervised. Yet, there is one another type of learning method that combines the two main types. It is called semi-supervised learning. This learning approach has been studied intensively in the last few years, but in our literature reviews, we do not notice much of semi-supervised learning in the field of intrusion detection system. Therefore, the definition of the two major groups are limned as follow.

Supervised learning approach requires all instance records have predefined classes. The reason why it is called supervised learning because its learning process is like the process of a kid learns to recognize objects under the supervising of his teacher. First, teacher shows him picture of several objects. Then, the teacher tells him a word corresponding to each object. The kid tries to learn and memorize those objects by its properties such sharp, color, size etc. Later, the kid will be asked to identify an object like the ones in his memories. Expectantly, he will able to speak the name or word of that object. The teacher knows the correct answers. The learning process iteratively makes predictions on the training data and is corrected by the teacher. This learning process stops when the kid can achieve an acceptable level of performance. Similarly, supervised machine learning algorithm builds a model which should be able to divide the example instances with different classes. Supervised learning can be further grouped into two learning process problems:

- Classification: is when the output class label is a category such as normal and anomaly.
- Regression: is when the output class label is a real value such as weight value.

Unsupervised approach does not require class labels for instances in dataset. It is called unsupervised learning because in contrast to supervised learning, there is no correct answers and there is no teacher. Unsupervised algorithm lets machine to discover the structure in the dataset and defines clusters or groups by itself. Unsupervised learning algorithm problems can be further categorized into problems:

- Cluster: is where you want to discover the inherent clustering groups in the data such as clustering groups of normal behavior and anomaly behavior.
- Association rule: is where you want to discover rules that describe large portions of data such as customers that buy X item tend to buy Y item as well.

Five supervised machine learning algorithms are described in this section. These algorithms are based on different computational concepts namely decision tree, sequential rule covering, Naïve Bayes, Back-propagation artificial neural network and support vector machines.

3.4.1 Decision Tree

Decision tree is the simplest and most intuitive machine learning algorithm which is a powerful analytical tool for making decision. As its name stated, decision tree builds a tree-like structure of the dataset. The tree model contains two elements such as nodes and branches. A node is associated with a feature, while a branch is labeled with the values of associated feature. The tree ends with the "leaf nodes" which represent the different classes. The instances are classified into those different classes. To build a tree model, the algorithm selects the starting nodes or "root node" which its feature value has the highest discriminability. Next, a branch is created for each value of the selected feature. Each sub-branch is created by tested next feature. The process is repeated until the data cannot be divided further. The instances in the dataset are divided and assigned into classes by the values associated of feature.

In our work, we employed decision tree algorithm which written in JAVA as part of WEKA machine learning framework, known as J48 [25]. J48 is a variant of C4.5 decision tree algorithm

which is proposed by Quinlan [26]. This method based on "information gain" to select the feature to be tested at each step while the tree is being grown. Information gain is obtained from the concept of "entropy". Entropy is a measure of impurity or uncertainty of a given set of data, which is introduced by Shannon in 1948 [27]. More precisely, C4.5 main concept is based on the entropy of the label classes in the dataset. We would like to know which features can separate a set of data into smaller subsets such that each of them contains as little entropy as possible. Several auxiliary equations required to describe the entropy followed by the definition of entropy has been introduced in detail by Anantavrasilp [19]. The equations are used to calculate entropy and information gain as following:

- Entropy $H(D)$: is a measurement of the uncertainty values in the dataset D

$$H(D) = \sum_{c \in C} -p(c) \log_2 p(c) \tag{3.3}$$

  where,

  - D is the dataset for which entropy is being computed
  - C is the set of classes in D
  - $p(c)$ is the proportion of the number of elements in class c to the number of elements in set D, while $H(D) = 0$; the set D is perfectly classified

- Information gain $IG(A)$ is the measure of the difference in entropy from before to after the set D is split on an attribute A or how much uncertainty in D was reduced after splitting set D on attribute A.

$$IG(A, D) = H(D) \sum_{t \in T} p(t) H(t) \tag{3.4}$$

  where,

  - $H(D)$ is entropy of set D
  - T is the subsets created from dividing set D by feature A
  - $p(t)$ is the proportion of the number of elements in t to the number of elements in set D
  - $H(t)$ is entropy of subset t

3.4.2 Sequential Rule Covering

Decision tree splits the set of instances into smaller sets according to the value of the corresponding features. In these smaller sets, the processes are repeated until the data cannot be split any further. This technique is known as a "divide-and-conquer". This section, sequential rule covering technique is explained. Each class label of dataset is considered separately by producing a set of rule. This set of rule will cover characteristics or features of each class. When a rule is created, the instances are "covered" by the rule. Those instances are sliced out from the dataset. The iteration steps process until all instance examples are covered. The rule contains two primary parts namely precondition and conclusion. The precondition consists of one or multiple logical criteria or tests, which the instance to be classified must meet. Then the conclusion defines the class labels that assign to the instances. If its' feature values adhere to the logical criteria in the precondition part and its class matches the class specified in the conclusion part of the rule, then this instance is said to be covered by a rule. The algorithms should find the rule that covers as many instances.

In this work, a sequential rule covering known as JRip in WEKA machine learning framework is employed [25]. JRip is an implementation of sequential rule covering algorithm Repeated Incremental Pruning to Produce Error Reduction (RIPPER) in JAVA programming language. RIPPER is an optimized version of Incremental Reduced Error Pruning (IREP) which is proposed by William W. Cohen [28]. The concept of the algorithm is to grow the classification rule sets and prune their conditions to reduce classification errors [25]. In [19], the RIPPER algorithm such how it creates a rule set from a dataset and the method that are used to grow as well as prune the rule are described:

1. A target class c is given, RIPPER creates the rules by dividing dataset into D^c & D^{c-}
   (a) D^c is the number of instances in class c
   (b) D^{c-} is the number of instances in different classes
2. D is separated into such growing and pruning sets
   (a) Grow a rule by adding conditions to the rule until the rule is converge
   (b) Prune a rule by eliminating some conditions to make the rule more generic
   (c) Go to $2.a$ repeatedly, until the instances in of target class c are covered by the rules

When adding a condition to a rule, RIPPER attempts all feature values to get the rule that yields the highest information gain. Precisely, given r is a rule, l is a condition to be added to the rule r. Thus the $GrowEval$ equation is:

$$GrowEval(l, r) = t^c \left(log_2 \frac{t'^c}{t'^c + t'^{c-}} - log_2 \frac{t^c}{t^c + t^{c-}} \right) \tag{3.5}$$

where,

- t^c and t^{c-} are instances of the target and non-target classes covered by r
- t'^c and t'^{c-} are instances of target and non-target classes after l is added to r respectively

$GrowEval$ examines on the information before and after a condition is applied to a rule. A rule is grown repeatedly until that rule is greedily grown or converge. However, Those rules could be over-fitted. Thus, we need to cut some conditions out, this action is called pruning. When new conditions are added to a rule, some of them could be cut out to avoid over-fitted. RIPPER uses the pruning evaluation metric to test the quality of the rule:

$$PruneEval(r) = \frac{t^p - t^{p-}}{t^p + t^{p-}} \tag{3.6}$$

where,

- t^p and t^{p-} are instances in $Prune^c$ and $Prune^{c-}$ cover by r respectively
- The antecedents will be removed from the rule until the $PruneEval(r)$ decreases.

3.4.3 Naïve Bayes

Naïve Bayes (NB) is a classification algorithm that computes probability of an instance belonging to each class based on Bayes probability theorem [29]. In a nutshell, NB is probabilistic classifier which classifies an instance into the class that contains most probable value. Despite its simplicity, NB produces the competitive results comparing to other complex learning algorithms [25] such as artificial neural networks or support vector machines. The reason behind it is called "Naïve" because it just assumes that occurrence of one feature is completely independent to the occurrence

of another feature. NB computes the conditional probability (known as likelihoods) as the product of the individual probabilities for each feature. Let $P(c)$ denotes the prior probability of the given target class c. Thus, we can assume that probability of all classes is the same by the equation below:

$$P(c_i) = P(c_j), \forall c_i = c_j \in c \tag{3.7}$$

where, i and j is index of each class c.

We have a feature vector x, so the Bayes theorem equation is represented by:

$$P(c \mid x) = \frac{P(c)P(x \mid c)}{P(x)} \tag{3.8}$$

where,

- $P(x)$ is the probability that x is observed
- $P(x \mid c)$ is the probability of observing x in domain where c holds (observing x, in class c)
- $P(c \mid x)$ is the probability of being c give x

3.4.4 Artificial Neural Network

Artificial neural network (ANN) is one of computational learning method that replicated the nature of human brain [31]. ANN structures model of mathematics for computing in form of neurons confectionist. ANN contains a set of connected nodes known as neurons. Each connection acts as the synapses in a biological brain, which broadcasts the electrochemical signal from a neuron to other neurons. A neuron that receives a signal, it can process the signal further to another neurons that are connected to it. In general, the connections between neurons are known as edges. An edge contains a weight value, which can be adjusted as learning proceeds. The weight value is increased or decreased the strength of the signal at a connection. Typically, ANN is separated into graph of layers such input layer, hidden layers and output layer. Each layer performs different types of transformation tasks on the given inputs. Signals travel from the input layer to the output layer possibly after traversing the multiple hidden layers. ANN consists of three elements [25]:

- Activation function that transforms neuron's net input signal into a single output signal. The function resides in a computation "node".
- Network topology that dictates the structure of neurons network, such as number of input nodes, hidden (middle) layers, output nodes and how the nodes are connected.
- Training algorithm specifies how weight of the input signal is set and updated.

In this work, back-propagation ANN is occupied. The structure of back-propagation is shown in Figure 3.3 in regard of three layers such input, hidden, and output layer. Let $x_1, x_2, \ldots, x_n$ are the

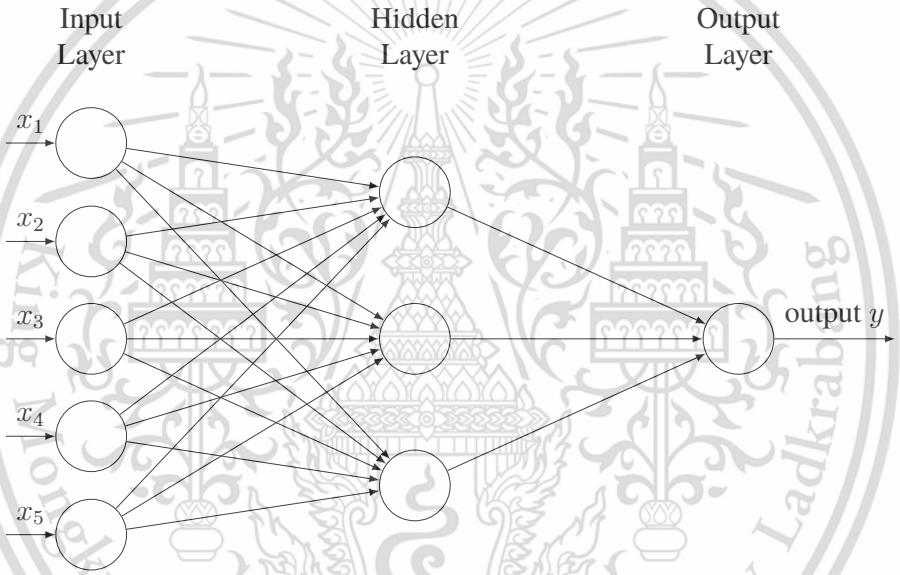


Figure 3.3: Back-propagation ANN's structure

inputs of the network. Hidden layer contains number of neuron nodes. Each neuron nodes attach with summation and activation function. Summation function performs the sum of multiplication of each input x_i and weight w_i (i is the index of each input $i = 1, \ldots, n$). The activation function $F(z)$ computes the transformation input to the output where, z is the output of each neuron in hidden layer. An example of activation function is sigmoid function represented in equation 3.9.

$$F(z) = \frac{1}{1 + e^{-z}} \tag{3.9}$$

The summation function computed by equation 3.10.

$$output = F\left(\sum_{i=1}^{n} x_i w_i\right) \tag{3.10}$$

3.4.5 Support Vector Machines

Support vector machine (SVM) is one of the most widely used machine learning algorithm for classification problem [32]. The SVM learning algorithm combines from two learning aspects, instance-based nearest neighbor and linear regression. This combination is extremely powerful which allows SVMs to model highly complex relationships of the given dataset. The mathematics that drive SVMs may be somewhat complex and difficult, but the basic concepts are understandable. The concept of SVM is to find a flat boundary decision known as "hyperplane" which divides the space to partition of data into groups of similar class values. In linear separable data, the hyperplane is a line, which is shown in Figure 3.4. In, non-linear separable data hyperplane is a plane, which is shown in Figure 3.5.

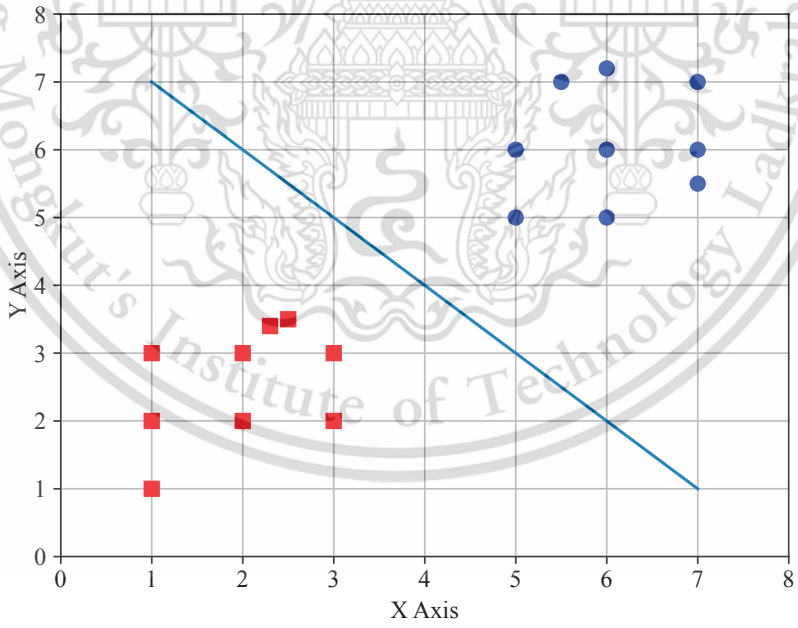


Figure 3.4: Two-dimensions Hyperplane

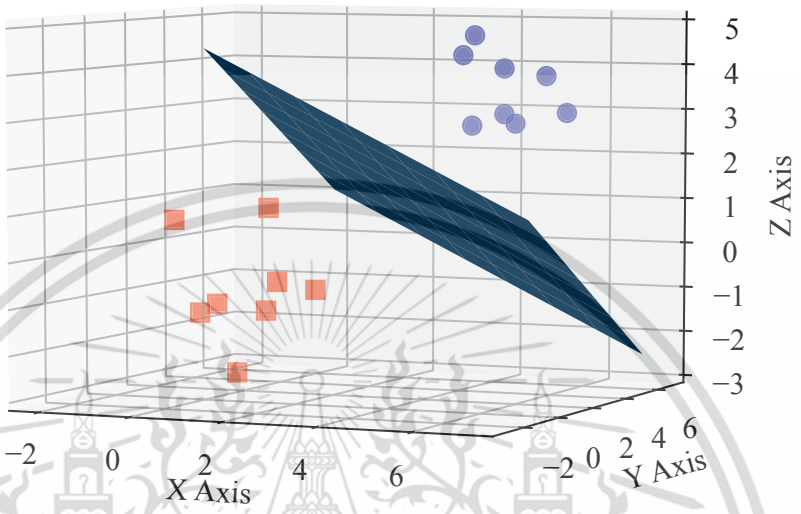


Figure 3.5: Three-dimensions Hyperplane

The most optimal hyperplane must provide the maximum margin that separate the data. For higher-dimension of data, the hyperplane is a surface in a higher-dimension space which can be difficult to visualize. Kernel trick [25] is applied to map data into higher dimensional such as $X \mapsto \phi(X)$. Well-known kernels are:

- Linear kernel

$$K(x_i, x_j) = (x_i \cdot x_j) \tag{3.11}$$

- Polynomial kernel

$$K(x_i, x_j) = (1 + x_i \cdot x_j)^p \tag{3.12}$$

where, p is degree of Polynomial Kernel.

- Sigmoid kernel

$$K(x_i, x_j) = \tanh\left(\beta_0(x_i \cdot x_j + \beta_1\right) \tag{3.13}$$

where, β_0 and β_1 is multiplicative and additive parameters respectively.

- Gaussian RBF kernel

$$K(x_i, x_j) = \exp\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{3.14}$$

where, σ is the nearest distance of support vectors

3.5 Performance Measurement

As number of related works have described, this field is still widely open to further investigations especially in regard of the accuracy of classifier performances. The performance of classifier is normally examined by computing distribution of instances that are either correctly classified or incorrectly classified. The confusion matrix is described about the performance of a classifier on the test set for which is shown in Table 3.5.

Table 3.5: The confusion matrix

		Actual Class	
		Negative (normal)	Positive (anomaly)
Predicted Class	Negative (normal)	True Negative (TN)	False Negative (FN)
	Positive (anomaly)	False Positive (FP)	True Positive (TP)

We compute the accuracy from this confusion matrix which has its formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.15}$$

- False Positive (FP): is the instances that are classified by IDS as being malicious when in fact they are normal.

- True Positive (TP): is the instances that are classified by IDS as being malicious and that they are malicious.

- False Negative (FN): is the instances that are classified by IDS as being normal when in fact, they are malicious.
- True Negative (TN): is the instances that are classified by IDS as being normal and that they are normal.

Chapter 4

Experimental Setup and Results

4.1 Experimental Setup

Mainly, our aim is to investigate on the impact of opted feature sets that could achieve a desired classification accuracy rate. Then, the robust classifier is revealed as well. First, ReliefF is employed to select 10 best features out of 41 features in KDD dataset. The best 10 features are ranked and shown in Table 4.1.

Then, we process to sort the most discriminative 10 features along with classes and save it into new dataset. Note: we keep all the total instances. Subsequently, that new dataset is divided into the other 1023 subsets by applying the feature combination function.

Table 4.1: The 10 best features ranking by ReliefF

Rank	**Feature**
1^{st}	service
2^{nd}	same_srv_rate
3^{rd}	dst_host_serror_rate
4^{th}	dst_host_cout
5^{th}	count
6^{th}	flag
7^{th}	dst_host_srv_count
8^{th}	dst_host_diff_srv_rate
9^{th}	dst_host_rerror_rate
10^{th}	dst_host_srv_serror_rate

For instance, one feature set combination of 10 best features could produce 10 subsets of indi-

vidual feature with class such as:

- Subset 1. {service, class}
- Subset 2. {flag, class}
- Subset 3. {count, class}
- Subset 4. {same_srv_rate, class}
- Subset 5. {dst_host_count, class}
- Subset 6. {dst_host_srv_count, class}
- Subset 7. {dst_host_diff_srv_rate, class}
- Subset 8. {dst_host_serror_rate, class}
- Subset 9. {dst_host_srv_serror_rate, class}
- Subset 10. {dst_host_rerror_rate, class}

Two feature sets combination of 10 best features contains 45 subsets and three feature sets combination of the 10 best features contains 120 subsets. Four, five, six, seven, eight and nine combination out of 10 contains 210, 252, 210, 120 and 10 subsets respectively. Due to the list of latter is pretty long, we do not list its' subsets. The last one is ten feature sets combination of 10 is one subset as:

- Subset 1.{service, flag, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, class}

Once, all the 1023 subsets data are divided, several machine learning algorithms including decision tree (J48), sequential rule covering (JRip), Naïve Bayes (NB), back-propagation artificial neural network (ANN), and support vector machines (SVM) are employed to perform the classification tasks. To evaluate each algorithm, we divide 50% of each subset and use as training set while keeping the other half as test sets. All classifiers built the classification models on the training set and evaluate on the test set. Each classifier performance is deemed on accuracy rate, which is explained in Chapter 3 Section 3.5. Last, the experimental results are collected and analyzed. Our experiments were conducted on WEKA experimenter version 3.8.2 [25].

4.2 Experimental Results

The set of features accompanied by the accuracy of each classifier are reported as follow:

- The combination set of one feature: **{service}** feature yields the highest classification rate among the other subset of feature for all classifiers, while {flag} feature is the second highest. The highest rate of each classifier is in bold text highlight which is shown in Table 4.2.

Table 4.2: The combination set of one feature accuracy rates

Set of Feature	**J48**	**JRip**	**NB**	**ANN**	**SVM**
{service}	**84.09**	**83.10**	**84.09**	**84.05**	**84.09**
{flag}	82.58	82.27	82.58	82.54	82.58
{count}	78.02	77.26	77.92	77.57	76.81
{same_srv_rate}	82.02	81.96	82.02	81.60	80.31
{dst_host_count}	65.76	65.76	65.76	65.31	62.70
{dst_host_srv_count}	77.73	76.61	77.67	77.27	73.38
{dst_host_diff_srv_rate}	80.35	80.32	80.17	79.27	79.00
{dst_host_serror_rate}	76.74	76.61	76.72	76.67	75.62
{dst_host_srv_serror_rate}	76.11	76.11	76.09	76.10	76.03
{dst_host_rerror_rate}	60.05	59.27	59.91	59.05	57.14

- The combination set of two features: since two and the other feature sets combination of 10 best features contain large number of subsets, we do not list them all. The corresponding remaining of table results, we will list only the set of features that yields the highest accuracy of each classifier. In Table 4.3, three classifiers: NB, ANN and SVM could achieve highest performance on {service, flag} feature sets. J48 and JRip are highest performance on {service, dst_host_diff_srv_rate} feature sets. However, accuracy rate of J48 and JRip on {service, flag} feature sets are not far behind accuracy rate on {service, dst_host_diff_srv_rate} feature sets. In addition, based on one feature set results, {service} and {flag} are the first and second highest. Thus, it could imply that two feature sets combination **{service, flag}** is the highest performance feature sets.

- The combination set of three features: in three feature sets combination of 10 best features, JRip, ANN and SVM are high perform on the same combination feature sets {service, flag, dst_host_diff_srv_rate}. Although J48 and NB achieve on the other sets, the performance

Table 4.3: The combination set of two features along with the highest accuracy rates for each classifier

Set of Features	J48	JRip	NB	ANN	SVM
{service, flag}	91.28	89.09	**89.86**	**91.12**	**90.97**
{service, dst_host_diff_srv_rate}	**91.63**	**89.93**	87.09	90.50	87.14

of J48 and NB on {service, flag, dst_host_diff_srv_rate} are followed slightly behind. Thus, the highest performance feature sets is **{service, flag, dst_host_diff_srv_rate}**. Table 4.4 indicates the feature combination sets and each classifier accuracy rate.

Table 4.4: The combination set of three features along with the highest accuracy rates for each classifier

Set of Features	J48	JRip	NB	ANN	SVM
{service, flag, dst_host_diff_srv_rate}	94.99	**94.45**	89.96	**94.50**	**92.50**
{service, count, dst_host_diff_srv_rate}	**95.01**	93.66	89.09	92.96	88.89
{service, dst_host_srv_serror_rate, dst_host_rerror_rate}	93.35	92.27	**90.91**	91.81	89.53

- The combination set of four features: in four feature set combination of 10 best features, J48 and JRip are high perform on the same feature sets, while NB, ANN and SVM on the each different set. Based on the observation among this high perform feature sets, the set **{service, flag, dst_host_diff_srv_rate, dst_host_count}** is the highest performance feature set for all classifiers.
- The combination set of five features: all classifiers show the highest performance on each different feature set. Moreover, the accuracy of each classifier for each feature sets is not

Table 4.5: The combination set of four features along with the highest accuracy rates for each classifier

Set of Features	J48	JRip	NB	ANN	SVM
{service, flag, dst_host_diff_srv_rate, dst_host_count}	96.33	95.52	90.52	**95.65**	92.72
{service, flag, dst_host_count, dst_host_rerror_rate}	95.74	94.85	**92.36**	94.54	91.64
{service, flag, dst_host_diff_srv_rate, dst_host_rerror_rate}	96.37	95.62	87.92	94.99	**92.9**
{service, count, dst_host_diff_srv_rate, dst_host_rerror_rate}	**96.63**	**95.76**	89.58	94.44	89.35

Table 4.6: The combination set of five features along with the highest accuracy rates for each classifier

Set of Features	**J48**	**JRip**	**NB**	**ANN**	**SVM**
{service, flag, same_srv_rate, dst_host_diff_srv_rate, dst_host_rerror_rate}	96.70	96.16	87.47	94.03	**93.30**
{service, flag, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate}	97.23	96.75	88.71	**96.40**	93.19
{service, flag, dst_host_count, dst_host_srv_count, dst_host_rerror_rate}	97.49	**96.90**	90.34	96.36	92.17
{service, flag, dst_host_count, dst_host_serror_rate, dst_host_rerror_rate}	96.20	95.48	**92.59**	94.85	92.12
{service, count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}	**97.66**	96.78	87.78	94.62	90.03

significantly different as well. It is hard to distinguish which set of feature should be elect for all classifiers. Based on the Table 4.6, we select the **{service, flag, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate}** feature set for all classifiers.

- The combination set of six features: similarly to five feature sets, all classifiers are higher performance on different set of features and also the accuracy of each classifier for each feature sets is not far different. Based on the Table 4.7, we select the **{service, flag, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}** feature set for all classifiers.

- The combination set of seven features: same as five and six combination feature sets, all classifiers are higher performance on different set of features and also the accuracy of each classifier for each feature sets is not far different. We elect **{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}** as feature sets for all classifiers. Table 4.8 shows the set of features and accuracy of all classifiers.

- The combination set of eight features: in eight feature set combination of 10 best features, J48 and JRip achieve the high accuracy on the same set {service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}, whereas NB, ANN and SVM yield high accuracy at each different sets. As shown in Table 4.9, the accuracy of NB, ANN and SVM on the same feature sets as J48 and JRip are closely behind each other. Thus, **{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate,**

Table 4.7: The combination set of six features along with the highest accuracy rates for each classifier

Set of Features	**J48**	**JRip**	**NB**	**ANN**	**SVM**
{service, flag, count, dst_host_count, dst_host_serror_rate, dst_host_rerror_rate}	97.32	96.74	**92.21**	95.98	92.75
{service, flag, same_srv_rate, dst_host_diff_srv_rate, dst_host_srv_serror_rate, dst_host_rerror_rate}	96.86	96.34	87.49	95.56	**93.61**
{service, flag, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}	**98.19**	97.75	88.91	96.82	93.50
{service, count, dst_host_count, dst_host_srv_count, dst_host_serror_rate, dst_host_rerror_rate}	98.14	**97.88**	90.74	96.75	91.68
{service, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	98.15	97.78	88.58	**97.08**	92.72

Table 4.8: The combination set of seven features along with the highest accuracy rates for each classifier

Set of Features	**J48**	**JRip**	**NB**	**ANN**	**SVM**
{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}	**98.52**	98.11	90.13	96.74	93.41
{service, flag, count, dst_host_count, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate}	97.40	96.77	**91.91**	95.40	92.23
{service, flag, same_srv_rate, dst_host_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	97.49	96.96	90.19	95.06	**93.62**
{service, flag, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	98.26	97.83	88.89	**97.51**	93.52
{service, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	98.50	**98.27**	89.90	97.27	92.85

Table 4.9: The combination set of eight features along with the highest accuracy rates for each classifier

Set of Features	J48	JRip	NB	ANN	SVM
{service, flag, count, same_srv_rate, dst_host_count, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate}	97.74	97.13	**90.83**	96.20	92.57
{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	**98.56**	**98.35**	89.63	96.75	93.53
{service, flag, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	98.28	98.12	88.52	97.00	**93.66**
{service, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	98.50	98.21	89.20	**97.46**	92.83

dst_host_serror_rate, dst_host_rerror_rate} is opted as set of features that could provides higher accuracy rate among all classifiers.

- The combination set of nine features: similarly to eight combination sets, J48 and JRip could yield the highest accuracy rates on the same sets, while the other classifier are best perform on each different set. As shown in Table 4.10, the accuracy of ANN and SVM, on the same set with J48 and JRip are slightly different, expect NB is not. **{service, flag, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}**, this set of feature is an outstanding set over the other.
- The combination set of ten features: the results are displayed in Table 4.11.

After the combination feature sets which yield the highest accuracy rate of each classifier are revealed, we collect them all into the Table 4.12. As shown in Table 4.12, the highest performance of each classifier can achieve corresponding with number of features are listed below:

- J48: 98.64% using 9 features.
- JRip: 98.44% using 9 features.
- NB: 90.52% using 4 features.
- ANN: 97.44% using 10 features.

Table 4.10: The combination set of nine features along with the highest accuracy rates for each classifier

Set of Features	J48	JRip	NB	ANN	SVM
{service, flag, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	**98.64**	**98.44**	88.91	97.11	93.62
{service, flag, count, same_srv_rate, dst_host_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate}	98.18	97.60	**90.07**	96.13	93.63
{service, flag, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate}	98.41	97.91	88.46	96.49	**93.70**
{service, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate}	98.54	98.34	88.84	**97.53**	92.97

Table 4.11: The combination set of ten features along with the highest accuracy rates for each classifier

Set of Features	J48	JRip	NB	ANN	SVM
{service, flag, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate}	98.58	98.34	88.93	97.44	93.65

- SVM: 93.65% using 10 features.

J48, ANN and SVM could achieve over 90% accuracy using only two features, with JRip and NB are slightly close behind at 89.09% and 89.86% respectively. Accuracies of almost all classifiers, except NB, increase proportional to number of employed features. However, starting from 7 features, the accuracies of all classifiers do not show significant increment. J48 and JRip algorithms do not gain any improvements after 9 features. NB accuracies are even dropped after 4 features. Thus, we believe that large number of features are not necessary, only few features with high discriminability value are sufficient. In consideration of feature sets, to the best of our knowledge **7-{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}** is the minimal sets that should be employed.

Table 4.12: The highest performance of each combination feature sets along with classifiers

The Best Set of Features	**J48**	**JRip**	**NB**	**ANN**	**SVM**
1-{service}	84.09	83.10	84.09	84.05	84.09
2-{service, flag}	91.28	89.09	89.86	91.12	90.97
3-{service, flag, dst_host_diff_srv_rate}	94.99	94.45	89.96	94.50	92.50
4-{service, flag, dst_host_diff_srv_rate, dst_host_count}	96.33	95.52	**90.52**	95.65	92.72
5-{service, flag, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate}	97.23	96.75	88.71	96.40	93.19
6-{service, flag, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}	98.19	97.75	88.91	96.82	93.50
7-{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}	98.52	98.11	90.13	96.74	93.41
8-{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	98.56	98.35	89.63	96.75	93.53
9-{service, flag, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	**98.64**	**98.44**	88.91	97.11	93.62
10-{service, flag, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate}	98.58	98.34	88.93	**97.44**	**93.65**

Considering the overall performance J48 is the best classifier. Although J48 shows the best performance in our experiments, the performances other learning algorithms are not far behind. Both J48 and JRip show more than 98% correctness while ANN achieves more than 97%. It is also interesting to note that accuracies of NB are holding around 88-90% after 4 features. We believe this is because the selected features are not correlating to each other, resulting in inconsistent probability values.

Chapter 5

Conclusion

5.1 Conclusion

This thesis presents a study on the impact of feature sets for intrusion detection system. Mainly, our aim is to figure out the robust classifier and the most discriminative feature set which is the smallest set that should be employed for classifying malicious threats.

KDD dataset, ReliefF feature selection algorithm, feature combination function, machine learning such as J48, JRip, NB, ANN, and SVM were employed in our work. All classifiers were measured based on the accuracy performance. Experimental results show that 90% classification accuracies can be achieved using only two features. Accuracies of almost all classifiers, except NB, increase proportional to number of employed features. However, the accuracies do not significantly improve after seven features and stop improving entirely after nine features. In consideration of feature sets, **7-{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}** is the minimal sets that should be employed. Considering the overall performance of classifier J48 is the robust classifier. However, the performances other of classifiers are slightly different.

Based on the experimental results, it is safe to say that by choosing a proper set of features (that could be minimal sets) and classifier, one can achieve desired classification performance without wasting computational effort to process non-discriminative features. More importantly, adding more features into consideration would yield better performances up to a certain point. In some cases, adding more features might even lead to inferior results.

To the best of our knowledge, the study of the investigations on the effect of the sizes of feature sets and classification accuracies similar to ours have never been carried out before. We believe that our works would lead to more efficient intrusion detection system and be beneficial in related areas especially behavior-based IDS, malicious flows detection or Internet traffic classification.

5.2 Potential Direction

In any case, despite promising results, there are still rooms for improvements. For future work should be examined the impact of elected feature sets by employing unsupervised machine learning approaches and extend the analysis to another intrusion detection dataset.

References

[1] I. Anantavrasilp and T. Schoeler, "Automatic flow classification using machine learning," in 2007 15th International Conference on Software, Telecommunications and Computer Networks, 2007, pp. 1-6.

[2] P. Garcia-Teodoro, J. Diaz-Verdejo, G.Macia-Fernandez, and E.Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," Comput. Secur., vol. 28, pp. 18-28, 2009.

[3] J. P. Early, C. E. Brodley, and C. Rosenberg, "Behavioral authentication of server flows," in 19th Annual Computer Security Applications Conference, 2003. Proceedings., 2003, pp. 46-55.

[4] S. J. Stolfo, F. Wei, L. Wenke, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: results from the JAM project," in DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings, 2000, pp. 130-144 vol.2.

[5] S. Benferhat and K. Tabia, "On the combination of naive Bayes and decision trees for intrusion detection," in International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), 2005, pp. 211-216.

[6] A. Pradhan, "Network Traffic Classification using Support Vector Machine and Artificial Neural Network," 2011.

[7] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 dataset," in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1-6.

[8] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," in The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), 2014, pp. 1-6.

[9] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in 2015 International Conference on Signal Processing and Communication Engineering Systems, 2015, pp. 92-96.

[10] G. Yan, "Network Anomaly Traffic Detection Method Based on Support Vector Machine," in 2016 International Conference on Smart City and Systems Engineering (ICSCSE), 2016, pp. 3-6.

[11] Y. Miao, Z. Ruan, L. Pan, J. Zhang, Y. Xiang, and Y. Wang, "Comprehensive Analysis of Network Traffic Data," in 2016 IEEE International Conference on Computer and Information Technology (CIT), 2016, pp. 423-430.

[12] M. Boger, T. Liu, J. Ratliff, W. Nick, X. Yuan, and A. Esterline, "Network traffic classification for security analysis," in SoutheastCon 2016, 2016, pp. 1-2.

[13] J. Frank, "Artificial intelligence and intrusion detection: Current and future directions," in Proceedings of the 17th national computer security conference, 1994, pp. 1-12.

[14] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," Computer Networks, vol. 31, pp. 805-822, 1999/04/23/ 1999.

[15] Alien Vault, "Host-based Intrusion Detection System," https://www.alienvault.com/solutions/host-intrusion-detection-system, Webpage, Last accessed date: Jul-10-2018.

[16] S. Richardson, "Network Based IDS Sensor Placement," Mar-10-2018, https://www.cc-expert.us/security-policy/figure-1011-networkbased-ids-sensor-placement.html Webpage, Last accessed date: Jul-10-2018.

[17] M. Roesch, "Snort: Lightweight intrusion detection for networks," 1999.

[18] eEye Digital Security, "ANALYSIS: .ida Code Red Worm," Jul-17-2001, https://web.archive.org/web/20110722192419/http://www.eeye.com/Resources/Security-Center/Research/Security-Advisories/AL20010717, Webpage, Last accessed date: Jul-10-2018.

[19] I. Anantavrasilp, "Supervised Machine Learning Assisted Real-Time Flow Classification System," Technische Universität München, 2010.

[20] K. Kira and L. A. Rendell, "The feature selection problem: traditional methods and a new algorithm," presented at the Proceedings of the tenth national conference on Artificial intelligence, San Jose, California, 1992.

[21] I. Kononenko, "Estimating attributes: Analysis and extensions of RELIEF," in Machine Learning: ECML-94: European Conference on Machine Learning Catania, Italy, April 6–8, 1994 Proceedings, F. Bergadano and L. De Raedt, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 171-182.

[22] J. Howcroft, "Evaluation of Wearable Sensors as an Older Adult Fall Risk Assessment Tool," Doctor of Philosophy, University of Waterloo, 2016.

[23] M. Zhang and A. A. Sawchuk, "A feature selection-based framework for human activity recognition using wearable multimodal sensors," presented at the Proceedings of the 6th International Conference on Body Area Networks, Beijing, China, 2011.

[24] M. Robnik-Šikonja and I. Kononenko, "Theoretical and Empirical Analysis of ReliefF and RReliefF," Machine Learning, vol. 53, pp. 23-69, October 01 2003.

[25] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, "Data mining : practical machine learning tools and techniques," 2017.

[26] J. R. Quinlan, "C4.5: programs for machine learning," Morgan Kaufmann Publishers Inc., 1993.

[27] C. E. Shannon "A mathematical theory of communication (Part 1)," Bell Syst. Tech. J. vol. 27 pp. 379-423 1948

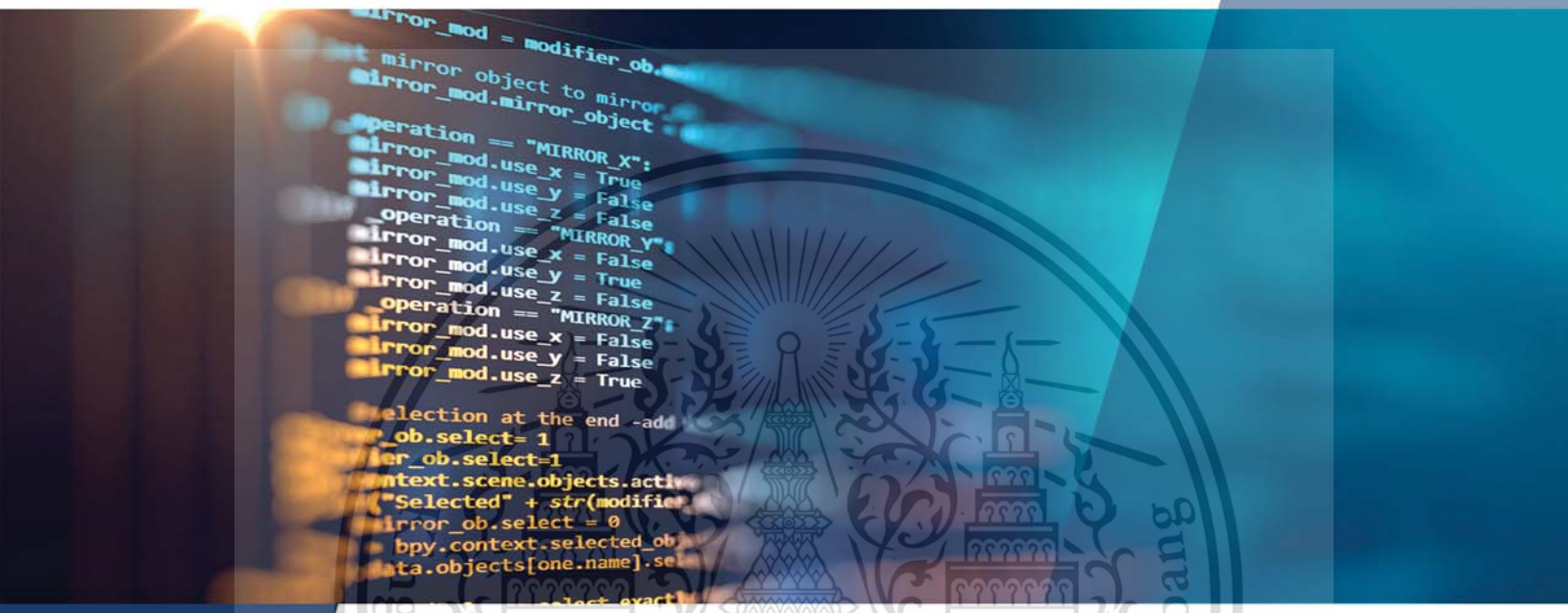
[28] W. W. Cohen, "Fast effective rule induction," in Proceedings of the twelfth international conference on machine learning, 1995, pp. 115-123.

[29] C. Lesmeister, "Mastering machine learning with R : master machine learning techniques with R to deliver insights for complex projects," 2015.

[30] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, 1995, pp. 338-345.

[31] R. Lippmann, "An introduction to computing with neural nets," IEEE Assp magazine, vol. 4, pp. 4-22, 1987.

[32] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," IEEE Intelligent Systems and their Applications, vol. 13, pp. 18-28, 1998.



2017 International Conference on Software and e-Business (ICSEB 2017)

December 28-30, 2017

Hong Kong

ICPS Published by ACM

The Effect of Sizes of the Feature Sets on Intrusion Detection Performances

Yoekleng Kuy
International College
King Mongkut's Institute of Technology Ladkrabang
10520, Bangkok, Thailand
Tel: +66 2 329 8261
yoekleng.kuy@gmail.com

Isara Anantavrasilp
International College
King Mongkut's Institute of Technology Ladkrabang
10520, Bangkok, Thailand
Tel: +66 2 329 8261
isara.an@kmitl.ac.th

ABSTRACT
Adaptive Intrusion Detection System (IDS) is a class of IDS that uses observed flows behaviors to detect malicious activities – usually with the aids of machine learning techniques. Most researches in this field focus on which features to be used or which classification methods to be employed. However, none have studied the impact of number of opted features on the accuracies of the anomaly detection or the smallest set of features that should be employed. This paper attempts to address these issues. We have applied feature selection algorithm, ReliefF [1] on NSL-KDD dataset [2] to select 10 most discriminative features out of 41 features. Then several machine learning algorithms are employed to classify normal and anomaly flows (both binary and multiple classes) using different set of features, each with different sizes. Experiment results show that >95% accuracies can be achieved with only 4-5 features and accuracy does not improve significantly after 6-7 features. We have also compared our results with other works and show that our work yields better results using the lower or the same number of features.

CCS Concepts
Security and privacy→Intrusion detection systems • Security and privacy→Artificial immune systems.

Keywords
Network traffic; NSL-KDD; intrusion detection system; machine learning; feature selection.

1. INTRODUCTION
Nowadays, computer networks, especially the Internet, play a vital role in every business and everyone's lives. All kinds of data or transactions are stored or conducted online. Protecting those data and transactions as well as network infrastructures from unauthorized persons or malicious activities is thus a very important and tedious task. One of the most widely-used tools to help detecting such malicious activities is Intrusion Detection System (IDS). Traditionally, an IDS detects malicious activities by scrutinizing network connections or "flows" that are coming in or going out from the networks. If a flow contains known or pre-programmed signatures, it will be marked as malicious or threats. Signatures could be known strings, binary sequences or behaviors.

ICSEB 2017, December 28–30, 2017, Hong Kong, Hong Kong.

ACM ISBN 978-1-4503-5488-2/17/12…$15.00
DOI: https://doi.org/10.1145/3178212.3178234

With precise signatures, the IDS can classify flows into exact services or protocols. However, the signatures must be specified and pre-programmed into the IDS beforehand. Thus, such method may not be able to detect unknown malicious flows.

To address such shortcomings, a new class of IDS is introduced [3]. Such IDSs do not rely on known, predefined signatures, but rather on observed flow behaviors. We call them Behavior-based IDS or B-IDS. B-IDS observes flow characteristics or "features" such as number of simultaneous connections from or to the same hosts, average sizes of packets, flow lengths, etc. to classify whether the observed flows are malicious. One can also classify them into normal and anomaly, where anomaly could be malicious flows or "unsure" ones. It is left to the network administrator to further scrutinize the anomaly flows whether they are threats or not. IDSs that classify flows into such classes are called anomaly-based IDS [4]. Furthermore, flows can also be classified into set of services or protocols similar to signature-based IDS as well.

Behavior-based IDS can be divided into three main categories statistical-based, knowledge-based and machine learning-based [4]. Statistical-based systems capture and profile flow characteristics using stochastic processes. Knowledge-based system employs pre-defined set of rules to analyze the flow behaviors. Machine learning-based methods employ machine learning algorithms to learn and capture characteristics of different types of flows without human intervention. This adaptive ability allows them to learn unseen behaviors and classify unknown flows. Due to their adaptiveness, various machine learning-based IDSs have been introduced [4-10]. They have also been thoroughly evaluated and compared [4, 8-10]. Numerous kinds of features have also been studied and evaluated – either by measuring the accuracies of the classifiers employing different sets of features or using feature selection algorithms to measure their discriminability [4, 8-10].

However, to the best of our knowledge, a study of relationship between number of features and classification accuracy is still missing. Without such knowledge, we might waste time computing too many features without improving any accuracy. In this regard, we would also like to know what the minimal set of features that yield desired classification performance is.

To address those questions, we select top 10 most discriminative features from the total of 41 features. Then, we further sort and group them into 10 different feature sets, each containing 1 to 10 features. Each set is used to build several flow classifiers using several machine learning algorithms, resulting in several classifiers corresponding to different feature sets and learning

algorithms. Impact of the sizes of the feature sets on the classification accuracies can be later analyzed.

Machine learning algorithms considered in this paper includes J48, a decision tree algorithm [11], a rule- induction learner [12], Naïve Bayes, a classifier based on Bayesian method [13], Backpropagation-based Artificial Neural Network (ANN) [14] and Support-Vector Machine (SVM) [15]. To evaluate the performance of the learning algorithms, we extract 41 features from the flows in NSL-KDD dataset [2]. Then, a feature selection algorithm called ReliefF [1] is employed to select 10 best features according to their discriminability. The performances of the classifiers are evaluated on binary-class scenario (normal and anomaly) and multi-class scenario (normal and four attack types). Experiment results show that >95% accuracies can be achieved with only 4-5 features and accuracies improve only slightly after 6-7 features. We have also compared our results with other adaptive IDSs that employ the same set of features and dataset.

The remaining parts of the paper is organized as follows: In Section 2, previous and related works are described. Section 3 provides descriptions the dataset, the feature selection algorithm and the machine learning approaches. Section 4 describes experiment process and methodology, followed by results analysis in Section 5. Section 6 concludes the paper.

2. PREVIOUS WORKS

In 2003, Early et al. employs machine learning techniques to classify flows into different network services. Statistics related to TCP flags are used as features [16]. Benferhat and Tabia proposed a combination of classifier decision tree and Naïve Bayes for anomaly-based IDS and re-analyzed the output results of the hybrid approach [17]. Pardhan employed two machine learning algorithms SVM and ANN to classify network traffic into 7 classes: FTP, WWW, P2P, NetBIOS, DNS, Mail, and Telnet [7]. Pervez and Fraid presented a combination approach of intrusion classification on multi-class NSL-KDD dataset by employing feature selection and SVM. They have evaluated their method on feature sets of different sizes; 3, 36, and 41 features. Accuracies obtained from aforementioned feature sets are 91%, 99% and 99%, respectively [8]. Although they have evaluated their classifier on different feature sets, the number of features in each set is still too broad and only one machine learning algorithm is employed. Ingre and Yadav conducted a study of an adaptive IDS using ANN on NSL-KDD dataset. Their method yields 81.2% classification accuracy when using 29 features and evaluated against binary-class dataset. On multi-class dataset, their method achieved 79.9% accuracy with 41 features [9]. Yan proposed to integrate the entropy theory and SVM to detect network anomaly traffic. He employed six features including source IP, destination IP, source port, destination port, packet size, and packet type [10]. A network traffic classification method using the principle component analysis (PCA) technique together with six machine learning algorithms including Naïve Bayes, Decision Tree, 1-Nearest Neighbor (NN), Random Forest, SVM and H2O (deep learning) was proposed by Miao et al. [18]. They also analyzed its performance based on two metrics including overall accuracy and F-measure. Boger et al. employed K-mean, an unsupervised machine learning algorithm, to classify network traffic into five clusters. Their dataset was obtained from the National Collegiate Cybersecurity Defense Competition (NCCDC) [19].

Frank carried out a survey of Artificial Intelligence methods which was used in anomaly-based IDS system [5]. He also presented a use case of feature selection to improve the performance classification of network traffic. In [4], Garcia-Teodoro et al., conducted a review of the several anomaly-based IDS and discussed challenges and solutions to large-scale anomaly-based IDS deployment.

Even though various researches on behavior-based intrusion detection are conducted, most of the works attempted to find the most effective features and classification methods. None has thoroughly investigated the relationship between number of features and classification accuracy. Using as many features as possible may not always improve the classification accuracy. Computational power and time would also be wasted. To address this issue, in this paper, the effect of number of features on the classification accuracy is examined. We will show that with properly selected set of features, combining with appropriate leaning methods, high classification accuracy can be achieved.

3. PRELIMINARIES AND DATA ANALYSIS

In the following, we will show whether the number of features used to build classifiers has any effect on the flow identification performance. We will also show what the smallest feature sets that yield certain accuracies are. We will start by describing and analyzing our datasets, features, feature selection and machine learning algorithms considered in our experiments. Then we will move on to our methodology, experiment settings and results in the next section.

3.1 Dataset

The dataset used in our experiments is called NSL-KDD [2], which is stemmed from DARPA'98 [6] packet traces. The DARPA'98 is consists of 4 gigabytes of compressed raw tcpdump traces captured by MIT Lincoln Laboratories. Total capturing time is seven weeks. The traces contain 5 million flows, each with around 100 bytes. During the Third International Knowledge Discovery and Data Mining Tool competition, a new dataset is extracted from original DARPA'98 traces to be used as a benchmark dataset. This new dataset is called KDD'99. Each flow in the dataset is labeled with a class and represented by 41 features. In 2009, the dataset is revised and refined by Tavallaee et al. [2] to remove redundant instances and the instances whose features are missing. This refined dataset is called NSL-KDD. It is currently employed by many, e.g., [8] [9], as benchmark dataset. It is thus employed in our experiment as well.

NSL-KDD dataset contains 148,517 flow instances, available in both binary-class and multi-class variants. For binary-class variant, the flows are labeled as normal or anomaly. For multi-class variant, the flows are labeled as normal or different type of attacks. The attacks, are shown in Table 1, are categorized into four groups as follows:

Denial of Service (DoS): A kind of attack such that the attacker tries to exhaust the network resources or services of the target system, e.g. land, smurf, udpstorm.
User to Root (U2R): The attacker attempts to gain access to the system as a normal users, then exploit the vulnerabilities of systems to gain access as super users, e.g. bufferoverflow, sqlattack, rootkit.
Remote to Local (R2L): Such attack occurs when the attacker remotely send packets to the target machine trying to exploit vulnerabilities to gain access to the machine, e.g. ftp_write, snmpgetattack, xsnoop.

Probing (Probe): The attacker attempts to scan on the target systems and looks for weakness or vulnerabilities e.g. mscan, portsweep, nmap.

Table 1. Attacks contained in NSL-KDD taken from [6]

DoS	U2R	R2L	Probe
pod	ps	phf	nmap
land	perl	spy	mscan
back	worm	imap	saint
smurf	xterm	xlock	satan
apache2	rootkit	named	ipsweep
teardrop	sqlattack	xsnoop	portsweep
neptune	loadmodule	sendmail	
udpstorm	snmpguess	multihop	
mailbomb	bufferoverflow	ftp_write	
processtabl e		httptunnel	
		warezclient	
		warezmaster	
		guesspasswd	
		snmpgetattack	

Number of instances of each class are shown in Table 2 and 3.

Table 2. NSL-KDD in binary class

Class	# Instances
Normal	77,054
Anomaly	71,463
Total	**148,517**

Table 3. NSL-KDD in multi-class

Class	# Instances
Normal	77,054
DoS	53,385
U2R	252
R2L	3,749
Probe	14,077
Total	**148,517**

3.2 Features

Instances in NSL-KDD have the same 41 features as those in KDD'99. The features can be categorized into three following groups:

Basic TCP/IP features: the individual TCP/IP connection.

Content features: the content of individual connection is suggested by domain knowledge.

Traffic features: is computed using a two-second time window in a certain of traffic connection.

Table 4 provides short descriptions of all features.

Table 4. Features description taken from [6]

Features	Description
1. Basic TCP/IP features	
duration	duration of connection
protocol_type	connection protocol (e.g. tcp, upd)
service	destination service (e.g. http, ftp, ssh)
flag	flag state of connection (e.g. syn, rej)
src_bytes	bytes sent from src to dst
dst_bytes	bytes sent from dst to src
land	1 if connection from the same host/port; 0 otherwise
wrong_fragement	number of wrong framgements
urgent	number of urgent packets
2. Content features	
hot	number of "hot" indicators
num_failed_logins	number of failed logins
logged_in	1 if successfully logged in, 0 otherwise
num_compromised	number of "compromised" conditions
root_shell	1 if root shell is obtained, 0 otherwise
su_attempted	1 if "su root" command attempted, 0 otherwise
num_root	number of "root" access
num_file_creations	number of file creation operations
num_shells	number of shell prompts
num_access_files	number of operations on access control files
num_outbound_cmd s	number of outbound commands in the ftp session
is_hot_login	1 if the login belongs to the "hot" list, 0 otherwise
is_guest_login	1 if the login is a "guest" login, 0 otherwise
3. Traffic features	
count	number of connections to the same host as the current connection in the past two-second
srv_count	number of connections to the same service as the current connection in the past two-second
serror_rate	% of connections that have "syn" error
srv_serror_rate	% of connections that have "syn" error
rerror_rate	% of connections that have "rej" error
srv_rerror_rate	% of connections that have "rej" error
same_srv_rate	% of connections to the same service
diff_srv_rate	% of connections to the diff service
srv_diff_host_rate	% of connections to the diff host
dst_host_count	count of connections having same destination host
dst_host_srv_count	count of connections having same destination host and using same service
dst_host_same_srv_ rate	% of connections having the same destination host and using the same service
dst_host_diff_srv_ra te	% of different services on the current host
dst_host_same_src_ port_rate	% of connections to the current host having the same source port
dst_host_ srv_diff_host_rate	% of connections to the same service coming from different hosts
dst_host_ serror_rate	% of connections to the current host that have an "S0" error
dst_host_ srv_serror_rate	% of connections to the current host and specified service that have the "S0" error
dst_host_ rerror_rate	% of connections to the current host

Features	Description
	that have the "rst" error
dst_host_ srv_rerror_rate	% of connections to the current host and specified service that have the "rst" error

3.3 Feature Selection

Machine learning algorithms identify flow types based on their characteristics or features. However, some features are redundant, not discriminative or do not work well with other features. Thus, simply adding features into consideration might not always yield better classification accuracy. Thus, finding the smallest set of features is essential. This is the goal of feature selection algorithms. Kira and Rendell proposed a feature selection algorithm called Relief on instance-based learning to update feature weight [20]. Relief selected feature sets based on a statistical method that does not rely on heuristics search. The complexity run-time is only linear time in the number of given features and the number of training data. However, it works only binary-class problem, limiting its uses. ReliefF, an extension to Relief that is applicable on multi-class problems is later proposed by Kononenko in [1]. It was one of the most popular feature selection algorithm [21] and due to its high performance [24]. Therefore, it is used in our experiments, to select most discriminative sets of features.

The algorithm ranks the features according to their relevance score. The score of a feature is calculated from distances of instances within the same class and between different classes considering the feature.

3.4 Machine Learning Algorithms

Machine learning algorithms allow computer to learn relevant concepts without being preprogrammed. They can be divided into supervised and unsupervised approaches. On the one hand, supervised approaches predefine labels or classes to which the instances will be classified into. On the other hand, unsupervised approaches let the machine define the classes by itself. In this paper, we focus only on supervised algorithms as the classes of the flows have already been defined. Five supervised machine learning algorithms are considered in our experiments:

1) J48: A decision tree algorithm written in Java as part of WEKA machine learning framework [22]. It is a variant of C4.5 decision tree algorithm proposed by Quinlan [11]. The algorithm is based on the concept of statistical property called information gain that measures how well a given feature separates the training examples with respect to the target class [23].

2) JRip: An implementation of rule-learning algorithm called Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [12]. RIPPER is an optimized version of incremental reduced error pruning (IREP) which is proposed by William W. Cohen. The concept of the algorithm is to generate the classification rule sets and prune their conditions to reduce classification errors and improve rules coverage [22].

3) Naïve Bayes (NB): A classification algorithm that computes probability of an instance belonging to each class based on Bayes probability theorem [13]. The instance is classified into the class that is most probable. Despite its simplicity, NB provides the competitive results comparing to other complex learning algorithms [23].

4) Artificial Neural Network (ANN): A computational learning method that replicated the nature of human brain known as neurons network [14]. Typical ANN is a graph of nodes, consists of three common elements [23]:

a. Activation function that transforms neuron's net input signal into a single output signal. The function resides in a computation "node".

b. Network topology that dictates the structure of neurons network, such as number of input nodes, hidden (middle) layers, output nodes and how the nodes are connected.

c. Training algorithm specifies how weight of the input signal is set and updated.

5) Support Vector Machine (SVM): A machine learning algorithm for classification problem which is based on statistical learning theory [15]. The background concept of SVM is to find a hyper plane with maximum margin to separate the data. In linear separable data, the hyper plane is a line. In, non-linear separable data hyper plane is a plane. Kernel trick [23] is applied to map data into higher dimensional. Well-known kernels are:

a. Linear kernel
b. Polynomial kernel
c. Sigmoid kernel
d. Gaussian RBF kernel

4. EXPERIMENT SETTINGS

The goal of this study is to find out whether only a few features are enough to identify which flows are anomaly or threats. We also investigate what is the smallest set of features that yield desired performance.

As mentioned earlier, the dataset used in our experiments, NSL-KDD, labels the instances in the dataset in two schemes: binary class and multi-class. To keep our experiment results comparable to other works, we will keep the original flow labels provided in NSL-KDD. Our experiments are conducted as follows. First, we will employ ReliefF to select the 10 best features out of 41 features provided in NSL-KDD. Then we assign features into 10 sets, each set consists of 1 best features, 2 best features, to all 10 best features. Table 5 shows the best 10 features sorted by their ranks and the feature sets in which they are assigned to (marked with X).

Table 5. List of features selected by ReliefF

Rank and selected features		Feature sets									
Rank	**Feature**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1st	service	X	X	X	X	X	X	X	X	X	X
2nd	same_srv_rate		X	X	X	X	X	X	X	X	X
3rd	dst_host_ serror_rate			X	X	X	X	X	X	X	X
4th	dst_host_count				X	X	X	X	X	X	X
5th	count					X	X	X	X	X	X
6th	flag						X	X	X	X	X
7th	dst_host_srv_count							X	X	X	X
8th	dst_host_diff_srv_rate								X	X	X
9th	dst_host_ rerror_rate									X	X
10th	dst_host_ srv_serror_rate										X

Then we evaluate all learning algorithms mentioned in Section 3 using all feature sets to see how the feature set sizes affect the performance of the classifiers. To evaluate each algorithm, we divide 50% of our dataset and use as training set while keeping the other half as test sets. Each learning algorithm is training using the training set. The learner yielded by the learning algorithm is then used to classify the instances on the test set. Performance of the classifier is determined by the ratio of correctly classified

instances against all instances. The evaluations are performed in both binary and multi-class scenarios.

Our experiments, including feature selection, learning and classification operations, are conducted on WEKA, a machine learning framework and toolset, version 3.8.1 [23]. The underlying hardware is Apple iMac Intel Core i5 with 4GB of RAM running MacOS 10.12 64-bit as operating system.

5. RESULTS AND DISSCUSSION

In this section, results of both binary and multi-class experiments are reported and analyzed. The comparisons between results of our proposed method and those of the previous works are also presented.

5.1 Binary Class

As shown in Table 6, J48, Naïve Bayes and ANN achieve 90% accuracy using only one feature, with SVM follows close behind at 89.9%. Accuracies of almost all classifiers, except Naïve Bayes, increase proportional to number of employed features. However, starting from 7 features, the accuracies of all classifiers do not increase significantly anymore. No algorithms can gain any improvements after 9 features. ANN's accuracy even drops after 9 features. Figure 1 provides a chart showing accuracies of each algorithm against number of features used. The highest performance each classifier can achieve along with corresponding number of features are listed below:

- J48: 98.7% using 9 features.
- JRip: 98.2% using 9 features.
- NB: 92.9% using 3 features.
- ANN: 96.6% using 9 features.
- SVM: 94.3% using 9 features.

Although J48 shows the best performance in our experiments, the performances other learning algorithms are not far behind. Both J48 and JRip show more than 98% correctness while ANN achieve more than 96%. It is also interesting to note that accuracies of Naïve Bayes are getting worse after 3 features. We believe this is because the selected features are not correlating to each other, resulting in inconsistent probability values.

Table 6. Accuracy rate in binary class. The highest performance of each learning algorithm is marked in bold

	Classifiers				
Feature set	**J48**	**JRip**	**NB**	**ANN**	**SVM**
1	90.0%	74.8%	90.0%	90.0%	89.9%
2	93.7%	92.8%	92.7%	93.6%	92.5%
3	94.3%	93.2%	**92.9%**	94.0%	93.0%
4	95.4%	94.6%	91.8%	94.8%	93.0%
5	96.6%	95.6%	91.6%	95.3%	93.5%
6	97.3%	96.4%	90.2%	92.9%	94.2%
7	98.0%	97.2%	90.3%	95.7%	94.2%
8	98.2%	97.9%	89.6%	95.9%	94.2%
9	**98.7%**	**98.2%**	89.8%	**96.6%**	**94.3%**
10	**98.7%**	**98.2%**	89.6%	96.5%	**94.3%**

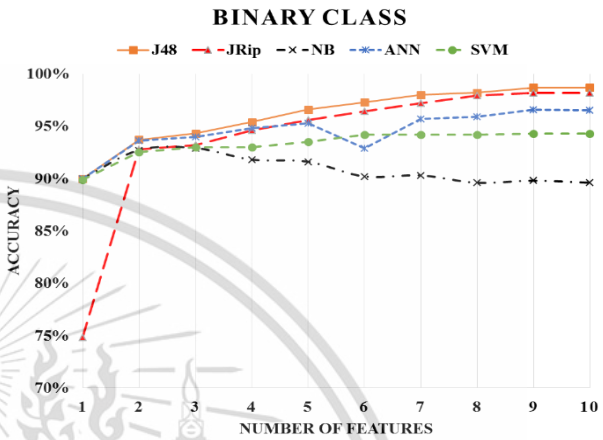


Figure 1. Accuracy rates line graph in binary class.

5.2 Multi-Class

Similar to binary class scenario, all algorithms except Naïve Bayes show performance gain when using more features. However, in multi-class case, only J48 gains more than 90% with two features followed by JRip and ANN with three features. After 6 features, no algorithm shows significant improvement. Interestingly, Naïve Bayes manages to hold around 88-90% after just two features. J48: subset features 9 and 10 provide 98.6%. The following summarizes the highest accuracy of each learning algorithm and corresponding number of features. Detailed classification accuracies are shown in Table 7. Figure 2 displays a chart showing accuracies of each algorithm against number of features used.

- J48: 98.6% using 9 features.
- JRip: 98.4% using 9 features.
- NB: 90.8% using 6 features.
- ANN: 97.1% using 8 features.
- SVM: 93.7% using 10 features.

Our experiments show that certain accuracy can be achieved only small set of features. In case of binary class, 90% correctness can be obtained using only one feature. In the multi-class case, where four kinds of attacks are classified separately, 90% correctness can be obtained with only two features. More importantly, in both cases, adding more features into consideration would yield better performances up to a certain point. Thus, it is safe to say that by selecting a proper set of features (that could be very small) and learning algorithm, one can achieve desired classification performance without wasting computational effort to process non-discriminative features. In some cases, adding more features might even lead to inferior results.

Table 7. Accuracy rate in multi-class. The highest performance of each learning algorithm is marked in bold

	Classifiers				
Feature set	**J48**	**JRip**	**NB**	**ANN**	**SVM**
1	84.1%	83.3%	84.1%	84.1%	84.1%
2	90.5%	88.9%	89.3%	89.9%	88.7%
3	92.1%	90.2%	88.1%	91.5%	89.3%
4	93.6%	91.8%	90.0%	92.6%	89.4%
5	95.6%	94.7%	90.3%	94.4%	89.9%
6	96.8%	95.7%	**90.8%**	95.3%	91.7%
7	97.6%	97.2%	89.2%	96.2%	92.3%
8	98.0%	97.7%	88.9%	**97.1%**	93.4%
9	**98.6%**	**98.4%**	89.0%	**97.1%**	93.6%
10	**98.6%**	**98.4%**	89.0%	96.9%	**93.7%**

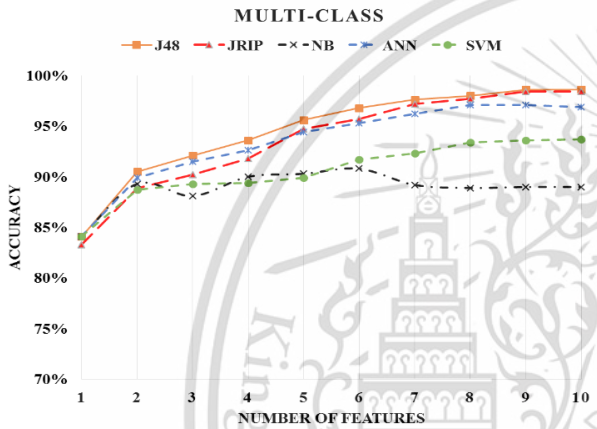


Figure 2. Accuracy rates line graph in multi-class.

This also holds true when comparing our results with previous works. As shown in Table 8, if J48 is employed as the learning algorithm, our method outperforms other works using only 3 features. It is also important to note that [8] and [9] used ~80% of the data as training set and only 20% as test set (as specified by [2]) while we use only 50% as training set. Their methodologies should yield better results than ours as they use more data to train the classifier. Yet, it is not the case.

Table 8. Accuracies of our method compared to others

	Employed Classifier	**Number of features**	**Accuracy (binary class)**	**Accuracy (multi-class)**
Pervez and Fraid [8]	SVM	3	N/A	91.01%
Ingre and Yadav [9]	ANN	29	81.2%	79.9%
Our Method	J48	3	94.3%	92.1%

6. CONCLUSION

In this paper, we employ ReliefF, a feature selection algorithm, to select 10 most discriminative features from total of 41 features. Then, several machine learning algorithms, namely J48, JRip, NB, ANN, and SVM, are used to build classifiers using different sets of features each with different sizes. Experiments conducted on NSL-KDD dataset show that 90% classification accuracies can be achieved using only one and two features in binary and multi-class scenarios, respectively. Performances of all classifiers, except Naïve Bayes, improve when proportional to the numbers of features used. However, in both binary class and multi-class cases, the accuracies do not significantly improve after 6-7 features and stop improving entirely after 9 features. Thus, we believe that large number of features are not necessary, only a few features with high discriminability are sufficient.

To the best of our knowledge, investigations on the effect of the sizes of feature sets and classification accuracies similar to ours have never been carried out before in the literature. We believe that our works would lead to more efficient intrusion detection system and be beneficial in related areas especially behavior-based IDS, malicious flows detection or traffic classification. In any case, despite promising results, there are still rooms for improvements. Our future work will include the impact of the sizes of feature sets on the unsupervised machine learning approaches. We will extend our analysis on other datasets as well.

7. ACKNOWLEDGMENTS

This study was supported by International College of King Mongkut's Institute of Technology Ladkrabang, Thailand. We would like thanks to our college and colleague who greatly assisted this research work.

8. REFERENCES

[1] Kononenko, I. Estimating attributes: Analysis and extensions of RELIEF. Springer Berlin Heidelberg, City, 1994.

[2] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A. A. A detailed analysis of the KDD CUP 99 data set. City, 2009.

[3] Anantavrasilp, I. and Scholer, T. Automatic flow classification using machine learning. City, 2007.

[4] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G. and Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28, 1 (2009/02/01/ 2009), 18-28.

[5] Frank, J. Artificial Intelligence and Intrusion Detection: Current and Future Directions, 1995.

[6] Stolfo, S. J., Wei, F., Wenke, L., Prodromidis, A. and Chan, P. K. Cost-based modeling for fraud and intrusion detection: results from the JAM project. City, 2000.

[7] Pradhan, A. Network Traffic Classification using Support Vector Machine and Artificial Neural Network, 2011.

[8] Pervez, M. S. and Farid, D. M. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. City, 2014.

[9] Ingre, B. and Yadav, A. Performance analysis of NSL-KDD dataset using ANN. City, 2015.

[10] Yan, G. Network Anomaly Traffic Detection Method Based on Support Vector Machine. City, 2016.

[11] Quinlan, J. R. C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., 1993.

[12] Cohen, W. W. Fast effective rule induction. City, 1995.

[13] John, G. H. and Langley, P. Estimating continuous distributions in Bayesian classifiers. Morgan Kaufmann Publishers Inc., City, 1995.

[14] Lippmann, R. An introduction to computing with neural nets. IEEE ASSP Magazine, 4, 2 (1987), 4-22.

[15] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J. and Scholkopf, B. Support vector machines. IEEE Intelligent Systems and their Applications, 13, 4 (1998), 18-28.

[16] Early, J. P., Brodley, C. E. and Rosenberg, C. Behavioral authentication of server flows. City, 2003.

[17] Benferhat, S. and Tabia, K. On the combination of naive Bayes and decision trees for intrusion detection. City, 2005.

[18] Miao, Y., Ruan, Z., Pan, L., Zhang, J., Xiang, Y. and Wang, Y. Comprehensive Analysis of Network Traffic Data. City, 2016.

[19] Boger, M., Liu, T., Ratliff, J., Nick, W., Yuan, X. and Esterline, A. Network traffic classification for security analysis. City, 2016.

[20] Kira, K. and Rendell, L. A. The feature selection problem: traditional methods and a new algorithm. In Proceedings of the Proceedings of the tenth national conference on Artificial intelligence (San Jose, California, 1992). AAAI Press, [insert City of Publication],[insert 1992 of Publication].

[21] Howcroft, J. Evaluation of Wearable Sensors as an Older Adult Fall Risk Assessment Tool. UWSpace, 2016.

[22] Witten, I. H., Frank, E., Hall, M. A. and Pal, C. J. Data mining: practical machine learning tools and techniques (2017).

[23] Lesmeister, C. Mastering machine learning with R: master machine learning techniques with R to deliver insights for complex projects (2015).

[24] Zhang, M. and Sawchuk, A. A. A feature selection-based framework for human activity recognition using wearable multimodal sensors. In Proceedings of the Proceedings of the 6th International Conference on Body Area Networks (Beijing, China, 2011).

King Mongkut's Institute of Technology
Ladkrabang,
International College

A Study on the Effects of Minimal Sizes of Feature Sets on Intrusion Detection Performances

Master Thesis

Presented by: Yoekleng Kuy (59610057)

Supervisor: Dr. Isara Anantavrasilp

July 17, 2018

Outline

- Introduction
- Related Works
- Methodology
- Experimental Setup
- Results
- Conclusion

Introduction

- Background
- Problem Statement
- Objective

Background

- Computer networks or the Internet are quite important for all organizations i.e. university
- There are large amounts of transaction data are generated across the networks
- Thus, safeguarding computer networks is a vital task
- The transaction data are known as “connection flows” or “flows”
- A “flow” can be classified into:
  - normal: is a normal user activity
  - anomaly: is a malicious or an unsure flow

Background

- Intrusion detection systems (IDS) are used to monitor on host or network for malicious flows
- Detection Methods:
  - Signature-based: uses pre-define rules or signatures
  - Behavior-based: observes flow characteristics or “features” by employing machine learning algorithms

Problem Statement

- While, behavior-based IDS could use machine learning to classify malicious flows or threats based on “features”
- However, the mechanism of malicious threats are sophisticated to evade themselves from the detection mechanism
- And, searching for a proper set of features and robust classifiers are still the concerning problems for detection performance of behavior-based IDS

Objective

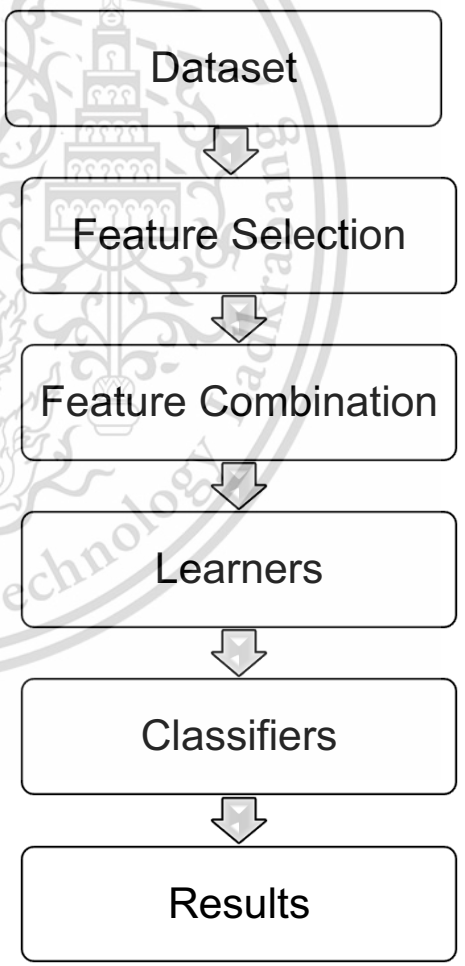
- The goal of our work is to find out what is the smallest set of features that should be employed to classify flows and robust classifier are revealed as well

Related Works

Previous works	Proposed Methods	Classes
Early et al., 2003	Decision tree (C5.0)	HTTP, FTP, SMTP, SSH and Telnet
Benferhat and Tabia, 2005	Decision tree and Naïve Bayes	Normal, DoS, U2R, R2L and Probe
Pradhan, 2011	SVM and Back-propagation ANN	FTP, WWW, P2P, NetBIOS, DNS, Mail and Telnet
Pervez and Fraid, 2014	Feature selection and SVM	Normal, DoS, U2R, R2L and Probe
Ingre and Yadav, 2015	Back-propagation ANN	Normal, DoS, U2R, R2L and Probe

Methodology

- The components of our proposed method consists of:
  - Dataset
  - Feature Selection
  - Feature Combination
  - Learners
  - Classifiers
  - Results



Dataset

- Dataset: KDD [Tavallaee, 2009] is the benchmark intrusion detection dataset which are derived from DAPAR'98 packet traces
- KDD contains 41 features and labeled with five classes

Class	# Instances
Normal	77,054
DoS	53,385
U2R	252
R2L	3,749
Probe	14,077
Total	148,517

Feature Selection

- ReliefF [Kononenko,1994], feature selection ranks and elects the best 10 features out of 41 features in KDD dataset
- ReliefF algorithm attempts to compute and rank the weights of features according to their relevance score
- The score of a feature is calculated from the distances of instances within the same class and between different classes on considering feature

Feature Combination

- After, feature selection is used to elect the 10 most relevant features
- Then, we apply the feature combination function which is represented by this formula:

$$C(n,k) = \frac{n!}{k!(n-k)!} \qquad (1)$$

- Where,
  - $n\ is\ the\ total\ number\ of\ feature\ n = 10$
  - $k\ is\ combination\ feature\ set\ 1 \leq k \leq n$

Machine Learning Algorithms

- Five supervised machine learning algorithms are employed namely:
  - Decision tree (J48) [Quinlan,1993]
  - Sequential rule covering (JRip) [Cohen, 1995]
  - Naïve Bayes (NB) [John and Langley,1995]
  - Back-propagation artificial neural network (ANN) [Lippmann, 1987]
  - Support vector machines (SVM) [Hearst et al., 1998]

Performance Measurement

- The performance of all classifiers are examined based on classification accuracy rate
- The accuracy can compute by this equation:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (2)$$

Experimental Setup

- First, we employed ReliefF to select 10 best features out of 41 features in KDD dataset
- Then, we process to sort the most discriminative 10 features along with classes and save it into new dataset
- Subsequently, that new dataset is divided into the other 1023 subsets by applying the feature combination function

Experimental Setup

Feature	Description
1. service	destination service i.e. http, ftp
2. same_srv_rate	% of connections to the same service
3. dst_host_serror_rate	% of connections to the destination host that have an “S0” error
4. dst_host_count	count of connections having same destination host
5. count	# of connections to the same host as current connection in the past two-second

Experimental Setup

Feature	Description
6. flag	Flag state of connection i.e. S0, RST
7. dst_host_srv_count	count of connections having same destination host and using same service
8. dst_host_diff_srv_rate	% of different services on the current host
9. dst_host_rerror_rate	% of connections to the destination host that have the “RST” error
10. dst_host_srv_serror_rate	% of connections to the destination host and specified service that have the “S0” error

Experimental Setup

- After that, decision tree (J48), sequential rule covering (JRip), Naïve Bayes (NB), back-propagation artificial neural network (ANN), and support vector machines (SVM) are employed
- We divide 50% of each subset and use as training set while keeping the other half as test set
- All classifiers built the classification models on the training set and evaluate on the test set.

Experimental Setup

- Each classifier performance is measured based on classification accuracy rate
- Last, the experimental results are collected and analyzed.
- Our experiments were conducted on WEKA experimenter version 3.8.2 [Witten et al., 2017]

Results

- In the experimental results,
  - First, we tried to identify the **most discriminative set of features** at each combination.
  - Then, we arrange each **most discriminative set of features** along with each accuracy rate for all classifiers into one table
  - Last, we analyze and compare to find the best performance classifier

Results

- The combination set of one feature, the most discriminative feature is **{service}**

Set of Feature	J48	JRip	NB	ANN	SVM
{service}	**84.09**	**83.10**	**84.09**	**84.05**	**84.09**
{flag}	82.58	82.27	82.58	82.54	82.58
{count}	78.02	77.26	77.92	77.57	76.81
{same_srv_rate}	82.02	81.96	82.02	81.60	80.31
{dst_host_count}	65.76	65.76	65.76	65.31	62.70
{dst_host_srv_count}	77.73	76.61	77.67	77.27	73.38
{dst_host_diff_srv_rate}	80.35	80.32	80.17	79.27	79.00
{dst_host_serror_rate}	76.74	76.61	76.72	76.67	75.62
{dst_host_srv_serror_rate}	76.11	76.11	76.09	76.10	76.03
{dst_host_rerror_rate}	60.05	59.27	59.91	59.05	57.14

Results

- The combination set of two features, the most discriminative feature set is **{service, flag}**

Set of Feature	J48	JRip	NB	ANN	SVM
{service, flag}	91.28	89.09	**89.86**	**91.12**	**90.97**
{service, dst_host_diff_srv_rate}	**91.63**	**89.93**	87.09	90.50	87.14

- The combination set of three features, the most discriminative feature set is **{service, flag, dst_host_diff_srv_rate}**

Set of Feature	J48	JRip	NB	ANN	SVM
{service, flag, dst_host_diff_srv_rate}	94.99	**94.45**	89.96	**94.50**	**92.50**
{service, count, dst_host_diff_srv_rate}	**95.02**	93.66	89.09	92.96	88.89
{service, dst_host_srv_serror_rate, dst_host_rerror_rate}	93.35	92.27	**90.91**	91.81	89.53

Results

- The combination set of four features

Set of Feature	J48	JRip	NB	ANN	SVM
{service, flag, dst_host_diff_srv_rate, dst_host_count}	96.33	95.52	90.52	95.65	92.72

- The combination set of five features

Set of Feature	J48	JRip	NB	ANN	SVM
{service, flag, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate}	97.23	96.75	88.71	96.40	93.19

Results

- The combination set of six features

Set of Feature	J48	JRip	NB	ANN	SVM
{service, flag, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}	98.19	97.75	88.91	96.82	93.50

- The combination set of seven features

Set of Feature	J48	JRip	NB	ANN	SVM
{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate}	98.52	98.11	90.13	96.74	93.41

Results

- The combination set of eight features

Set of Feature	J48	JRip	NB	ANN	SVM
{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	98.56	98.35	89.63	96.75	93.53

Results

- The combination set of nine features

Set of Feature	J48	JRip	NB	ANN	SVM
{service, flag, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_rerror_rate}	98.64	98.44	88.91	97.11	93.62

Results

- The combination set of ten features

Set of Feature	J48	JRip	NB	ANN	SVM
{service, flag, count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate}	98.58	98.34	88.93	97.44	93.65

Results

- The arrangement of each most discriminative set of features

The Best Set Features	J48	JRip	NB	ANN	SVM
1	84.09	83.10	84.09	84.05	84.09
2	91.28	89.09	89.86	91.12	90.97
3	94.99	94.45	89.96	94.50	92.50
4	96.33	95.52	**90.52**	95.65	92.72
5	97.23	96.75	88.71	96.40	93.19
6	98.19	97.75	88.91	96.82	93.50
7	98.52	98.11	90.13	96.74	93.41
8	98.56	98.35	89.63	96.75	93.53
9	**98.64**	**98.44**	88.91	97.11	93.62
10	98.58	98.34	88.93	**97.44**	**93.65**

Results

- In consideration of feature sets, to the best of our knowledge 7-{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate} is the minimal sets that should be employed.
- Considering the overall performance J48 is the best classifier.

Conclusion

- This thesis presents a study on the impact of minimal sizes of feature sets for intrusion detection performances
- Mainly, our aim is to figure out the robust classifier and the most discriminative feature set which is the smallest set that should be employed for classifying malicious threats
- KDD dataset, ReliefF feature selection algorithm, feature combination function, machine learning such as J48, JRip, NB, ANN, and SVM were employed in our work.

Conclusion

- Experimental results show that 90% classification accuracies can be achieved using only 2 features
- Accuracies of almost all classifiers, except NB, increase proportional to number of employed features
- However, the accuracies do not significantly improve after 7 features and stop improving entirely after 9 features

Conclusion

- In consideration of feature sets, 7-{service, flag, count, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_rerror_rate} is the minimal sets that should be employed. Considering the overall performance J48 is the best classifier
- It is safe to say that by choosing a proper set of features (that could be minimal sets) and classifier, one can achieve desired classification performance without wasting computational effort to process non-discriminative features

Conclusion

- More importantly, adding more features into consideration would yield better performances up to a certain point
- In some cases, adding more features might even lead to inferior results
- We believe that our works would lead to more efficient intrusion detection system and be beneficial in related areas especially behavior-based IDS, as well as malicious flows detection

Conclusion

- Potential direction for future work should be investigate the impact of elected feature sets by employing unsupervised machine learning approaches and extend the analysis to another intrusion detection datasets

Q & A

Thank for you attention

AUTHOR BIOGRAPHY

Author :	Mr. Yoekleng Kuy
Degree :	Master of Engineering
Date of Graduation :	July 17^{th}, 2018
Date of Birth :	February 12^{th}, 1991
Place of Birth :	Banteay Meanchey, Cambodia

Undergraduate and Graduate Education:

Master of Engineering in Computing in Engineering Systems,
King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand, 2018

Bachelor degree in Computer Science,
University of Puthisastra, Phnom Penh, Cambodia, 2013

Major: Computing in Engineering Systems

Presentations and Publications:

- Paper's name *:* **"The Effect of Sizes of the Feature Sets on Intrusion Detection Performances"** 28^{th}-30^{th}, December 2017, ICSEB 2017, Hong Kong, Hong Kong.