ADDIS ABABA UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF INFORMATION SCIENCE

COMPUTER NETWORK INTRUSION DETECTION: MACHINE LEARNING APPROACH

A THESIS SUBMITTED TO SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY

ADAMU TESHOME BEYENE

JULY, 2010

ADDIS ABABA, ETHIOPIA

ADDIS ABABA UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF INFORMATION SCIENCE

COMPUTER NETWORK INTRUSION DETECTION: MACHINE LEARNING APPROACH

BY

ADAMU TESHOME BEYENE

Approved by the Examining Board

_____	_____
Chairman, Examining Committee	Signature
_____	_____
Advisor	Signature
_____	_____
Examiner	Signature

ACKNOWLEDGEMENT

First and foremost, I offer my thanks to Lord Jesus, who made this work possible.

I would like to thank my advisor Workshet Lamenew for his constructive and uninterrupted comments and guidance.

Finally, my heartfelt appreciation and indebtedness goes to my family and friends.

Table of contents

Abstract

The conventional approach to securing computer systems against cyber threats is to design mechanisms such as firewalls, authentication tools, and virtual private networks that create a protective shield. However, these mechanisms almost always have vulnerabilities which are often, caused by careless design and implementation bugs. This has created the need for intrusion detection system that complements conventional security approaches by monitoring systems and identifying computer attacks.

Traditional intrusion detection methods are based on human experts' extensive knowledge of attack signatures which are character strings in a messages payload that indicate malicious content. These methods have several limitations. They cannot detect novel attacks, because someone must manually revise the signature database beforehand for each new type of intrusion discovered. Once someone discovers a new attack and develops its signature, deploying that signature in all the system is very difficult. The need for efficient detection of newly emerging malicious acts is increasingly important. The limitations in the traditional intrusion detection systems and the need for more efficient systems led to an increasing interest in intrusion detection techniques based on data mining.

The problems with the current researches on intrusion detection using data mining approach are that they try to minimize the error rate (make the classification decision to minimize the probability of error) by totally ignoring the cost that could be incurred. However, for many problem domains, the requirement is not merely to predict the most probable class label, since different types of errors carry different costs. Instances of such problems include authentication,

where the cost of allowing unauthorized access can be much greater than that of wrongly denying access to authorized individuals, and intrusion detection, where raising false alarms has a substantially lower cost than allowing an undetected intrusion. In such cases, it is preferable to make the classification decision that has minimum cost, rather than that with the lowest error rate.

For this reason, we examine how cost-sensitive classification methods can be used in Intrusion Detection systems. The performance of the approach is evaluated under different experimental conditions and different classification models in comparison with the KDD Cup 99 winner results, in terms of average misclassification cost, as well as detection accuracy and false positive rates.

Key words: Intrusion detection, Data mining, Cost sensitive learning.

List of figures

List of Tables

List of acronyms

CS: Cost Sensitive

DM: Data Mining

DOS: Denial of Service

GUI: Graphic User Interface

HIDS: Host Intrusion Detection Systems

ID: Intrusion Detection

IDS: Intrusion Detection Systems

IP: Internet Protocol

KDD: Knowledge Discovery and Data Mining

LAN: Local Area Network

NIDS: Network Intrusion Detection System

R2L: Remote to user attacks

TCP: Transport Control Protocol

U2R: User to root attacks

Chapter One

Introduction

1.1 Background

"A secure computer system is a system that can be depended upon to behave as it is expected to do" [1]. In order to achieve this, the components that make up the system must, at some point, be trusted. First, the hardware has to be trusted to behave as expected, thus minimizing the possibility of hardware failure. Second, the software installed and running on the system must be trusted to behave as expected and third, the users of the system must be trusted to behave as expected [15]. Even if all of the above is true, everyone who could gain access to the system (in the case when the computer is connected to the Internet, probably the whole world) have to be trusted to behave as expected. Since trust is a delicate virtue and often abused, a way to protect our computer systems is to detect any malicious activity and react upon the detection must be found [1, 15].

The field of intrusion detection has received increasing attention in recent years. One reason is the explosive growth of the Internet and the large number of networked systems that exist in all types of organizations [14]. The increased number of networked machines has led to a rise in unauthorized activity, not only from external attackers but also from internal sources such as disgruntled employees and people abusing their privileges for personal gain [1].

Since intrusions take advantage of vulnerabilities in computer systems or use socially engineered penetration techniques, intrusion detection (ID) is often used as another wall of protection [2].

Intrusion detection includes identifying a set of malicious actions that compromise the integrity, confidentiality, and availability of information resources.

Traditional methods for intrusion detection are based on extensive knowledge of signatures of known attacks. Monitored events are matched against the signatures to detect intrusions. These methods extract features from various audit streams, and detect intrusions by comparing the feature values to a set of attack signatures provided by human experts. The signature database has to be manually revised for each new type of intrusion that is discovered. A significant limitation of signature based methods is that they cannot detect emerging cyber threats, since by their very nature these threats are launched using previously unknown attacks. In addition, even if a new attack is discovered and its signature developed, often there is a substantial latency in its deployment across networks. The need for efficient detection of newly emerging malicious acts is increasingly important. The limitations in the traditional intrusion detection systems and the need for more efficient systems led to an increasing interest in intrusion detection techniques based on data mining [6].

There are different taxonomies for IDS have been suggested. One of these taxonomies depends on the source of audit data that will be used to detect possible intrusions. This was divided into two groups; network intrusion detection (NID) and host-based intrusion detection (HID) [5]. HID identifies the intruders by monitoring host-based traffic, while NID refers to identifying the intruders by monitoring network traffics (packets). This can be accomplished by identifying attacks using their known pattern (signature) [2].

Network Intrusion Detection Systems (NIDS) have become a standard component in security infrastructures as they allow network administrators to detect policy violations. These policy violations range the gamut from external attackers trying to gain unauthorized access (which can usually be protected against through the rest of the security infrastructure) to insiders abusing their access (which often times is not easy to protect against). Detecting such violations is a necessary step in taking corrective action, such as blocking the offender (by blocking their machine at the parameter, or freezing their account), by reporting them (to their supervisor), or taking legal action against them [2, 6].

Alternatively, detecting policy violations allows administrators to identify areas where their defenses need improvement, such as by identifying a previously unknown vulnerability, a system that was not properly patched, or a user that needs further education against social engineering attacks [6].

1.2 Statement of the Problem

Enough data exists or could be collected to allow network administrators to detect policy violations. Unfortunately, the data is so volumous, and the analysis process so time consuming that the administrators don't have the resources to go through it all and find the relevant knowledge, save for the most exceptional situations [6].

Given the nature of this problem, the possible solution is data mining approach [6].Data mining approach for intrusion detection techniques generally fall into one of two categories; misuse detection and anomaly detection. In misuse detection, each instance in a data set is labeled as 'normal' or 'intrusion' and a learning algorithm is trained over the labeled data. These

techniques are able to automatically retrain intrusion detection models on different input data that include new types of attacks, as long as they have been labeled appropriately. Unlike manual intrusion detection systems, models of misuse are created automatically, and can be more sophisticated and precise than manually created signatures. A key advantage of misuse detection techniques is their high degree of accuracy in detecting known attacks and their variations. Their obvious drawback is the inability to detect attacks whose instances have not yet been observed.

Anomaly detection, on the other hand, builds models of normal behavior, and automatically detects any deviation from it, flagging the latter as suspect. Anomaly detection techniques thus identify new types of intrusions as deviations from normal usage [7]. While an extremely powerful and novel tool, a potential draw-back of these techniques is the rate of false alarms. This can happen primarily because previously unseen (yet legitimate) system behaviors may also be recognized as anomalies, and hence flagged as potential intrusions. Hybrid IDS combine both methods and it is better one [8].

The problem with the current researches is that they try to minimize the error rate (make the classification decision to minimize the probability of error) by totally ignoring the cost that could be incurred. However, for many problem domains, the requirement is not merely to predict the most probable class label, since different types of errors carry different costs. For example of such problem in the case of computer network security include authentication, where the cost of allowing unauthorized access can be much greater than that of wrongly denying access to authorized individuals, and intrusion detection, where raising false alarms has a substantially lower cost than allowing an undetected intrusion. In such cases, it is preferable to make the

classification decision that has minimum expected cost, rather than that with the lowest error probability [33].

Another very important case is, if class imbalanced datasets occurs but this happens in many real-world applications where the class distributions of data are highly imbalanced [33]. Again, it is assumed that the minority or rare class is the positive class, and the majority class is the negative class. Often the minority class is very small, such as 1% of the dataset. If most traditional (cost insensitive) classifiers are applied on the dataset, they will likely to predict everything as negative (the majority class) [24].

The intrusion data used for this research, KDD data set, which is publicly available and most widely used data set, has class distributions of training and test datasets with different distribution and each attack types has different costs. Statistically, the attacks of U2R and R2L are of the most rare, which makes them very hard to predict. On the other hand, they are the most dangerous types. Once an attacker gains the super user right or successfully remote login, disasters of the whole system are nothing but unavoidable [19].

Comparably, attacks of Probe are not that much dangerous. Although attacks of DOS (denial of service) are massive in the whole original dataset, they impose less danger. This is because the nature of denial of service attacks lies in that they are trying to initiate as many as possible connections to consume the network traffics and server CPU time [19].

Because of the above mentioned reasons cost sensitive learning which considers cost in decision making with acceptable accuracy is better solution for computer network intrusion detection. We used cost sensitive learning algorithms, cost sensitive classification tree CS-CRT and cost

sensitive decision tree CS-MC4 algorithms. These algorithms use misclassification cost matrix to minimize the expected cost and for the detection of best prediction. Yet despite its importance, the topic is seldom addressed and researched.

1.3 Objectives

1.3.1 General Objective

The general objective of this research is to investigate the application of cost sensitive learning using data mining approach to network intrusion detection.

1.3.2 Specific Objectives

The specific objectives of this research are:

1. To review literature on the concept of intrusion detection in the area of data mining and cost sensitive learning.

2. Prepare, and preprocess KDD intrusion dataset.

3. To train and test models using CS-MC4 and CS-CRT algorithms.

4. To compare the performance of the models developed using CS-MC4 and CS-CRT algorithms.

5. Report results and make recommendations.

1.4 Scope and Limitation of the Study

Due to time constraint this research has looked at and concentrated mainly on intrusion detection methods and techniques. This work did not look at other components of an Intrusion Detection System (IDS) like data collection and response since it is very wide area of research. The NSL-KDD dataset is collected in a simulated environment so it might not be a perfect representative of the real networks.

1.5 Significance of the study

Network security is a blooming industry as more and more of the corporate workspace is converted to digital media. Because companies and home users keep sensitive information on their computers, there is a great need to protect that information from those who would exploit it. Even though conventional approach to securing computer systems against cyber threats (which use firewalls, authentication tools, and virtual private networks) create a protective shield. it almost always have vulnerabilities. One way to help keep attackers is by using an intrusion detection system (IDS), which are designed to locate and notify systems administrators to the presence of malicious traffic[6].

Detecting intrusions allows administrators to identify areas where their defenses need improvement, such as by identifying a previously unknown vulnerability, a system that was not properly patched, or a user that needs further education against social engineering attacks [6].

Detecting intrusions and other forms of malicious traffic in a large, modern corporate network is difficult and costly [33]. Something must be done in order to improve efficiency. So cost sensitive learning approach to intrusion detection can detect intrusion with acceptable rate of false positives and it also greatly cuts down costs, allowing network managers or system administrators engage on other productive endeavors.

1.6 Methodology

The procedures followed in conducting the research are described below:

1.6.1 Literature Review

The researcher conducted literature review to assess the major issues and concepts in the field of intrusion detection, data mining and cost sensitive learning. Various books, journals, articles and papers from the Internet and books were consulted to assess the importance and applications of cost sensitive learning and data mining technology in general and its application on network intrusion detection.

1.6.2 Data Collection

NSL-KDD data set which is the new version of KDD Cup 99 dataset and the only publicly available and the widely used data set for intrusion detection have been used for the experimental purpose.

1.6.3 Data Preparation

The process of data cleaning and preparation is highly dependent on the specific machine learning algorithm and software chosen and algorithms used for the machine learning task. The researcher attempted to prepare the data according to the requirements of the Tanagra which is powerful, user friendly and freely available for noncommercial purpose machine learning software and according to the requirements of CS-MC4 and CS-CRT algorithms by consulting different literatures.

After collecting the target dataset, the researcher checked the consistency of individual attribute values and types, and quantity , distribution of missing values and each attacks are categorized to five classes using Microsoft excel then the data format was changed into arff file format that can be understand by the Tanagra tool.

1.6.4 Building and Training Models

The Intrusion detection models were developed using cost sensitive classification tree CS-CRT and cost sensitive decision tree CS-MC4 algorithms on full training NSL-KDD dataset using a powerful machine learning and data mining Tanagra tool.

1.6.5 Testing models

Full testing dataset of NSL-KDD passed through the developed models to detect the intrusions and find the detection error rates, precision, false positives rate, average misclassification cost and accuracy of the detection models but for comparison of models we used mainly average misclassification cost, false positives rate and accuracy of detection.

1.7 Thesis Outline

The following chapter covers literature survey; this is to review the published and unpublished works, journals and books and to gather any information relevant to the research work. It gives background knowledge about computer security and intrusion detection, and intrusion dataset used. Chapter four introduces about data mining and its application to intrusion detection, and in more detail, the theoretical foundation of cost sensitive learning will be presented. Chapter five demonstrates the performance of the models developed using the selected cost sensitive algorithms tested on a network intrusion dataset and detailed analysis of results is given in the discussion part. The last chapter is devoted for the final conclusions and recommendations based on the research findings.

Chapter Two

Literature Review

This chapter discusses about the need for securing computer systems and the role of intrusion detection in security and it briefly describes the data used for intrusion detection model. It also gives a broad overview of the field of computer security and intrusion detection as it is presented in the literature. The next chapter is about data mining approaches using cost sensitive learning for detecting intrusions with more emphasis to cost sensitive learning.

2.1 Computer Security and its Role

Definition of a secure computer system is "one that can be depended upon to behave as it is expected to" [15]. The dependence on the expected behavior being the same as exhibited behavior is referred to as trust in the security of the computer system. The level of trust indicates the confidence in the expected behavior of the computer system. The expected behavior is formalized into the security policy of the computer system and governs the goals that the system must meet. This policy may include functionality requirements if they are necessary for the effective functioning of the computer system.

A narrower definition of computer security is based on the realization of confidentiality, integrity, and availability in a computer system [14]. Confidentiality requires that information be accessible only to those authorized for it, integrity requires that information remain unaltered by accidents or malicious attempts, and availability means that the computer system remains working without degradation of access and provides resources to authorized users when they

need it. By this definition, an unreliable computer system is unsecured if availability is part of its security requirements.

A secure computer system protects its data and resources from unauthorized access, tampering, and denial of use. Confidentiality of data may be important to the commercial success or survival of a corporation, data integrity may be important to a hospital that maintains medical histories of patients and uses it to make life critical decisions, and data availability may be necessary for real-time traffic control [14].

There is a close relationship between the functional correctness of a computer system and its security. Functional correctness implies that a computer system meets its specifications. If the functionality specification includes security policy requirements, then functional correctness implies security of the computer system. However, the reverse is not true, i.e., functional error may not result in violations of the security policy, especially as it relates to confidentiality, integrity, and availability. For example, an operating system service call may not process all valid arguments to it correctly [14], yet it may not be possible to violate the security policy by taking advantage of this fact.

2.2 Attack to Security

A computer attack is any malicious activity directed at a computer system or the services provided by the system. The attacks could come in different form or even a physical attack against the computer hardware [13]. An understanding of some techniques that have been used by attackers will go a long way in helping provide a system that will protect institutions against the intrusions [10].

Attackers have employed a varied number of techniques to break into systems and these include and are not limited to:

1. Social Engineering: This is the type of attack where the attacker fools an authorized network user into divulging information that leads to access to network resources. The details may include: passwords, security policies, network or host addresses. The attacker could also impersonate genuine users, service providers or the support staff and sometimes uses this identity to deliver harmful software such as Trojan.

2. Abuse of Feature: In abuse of feature, the authorized users perform actions that are considered illegitimate. He may open illegitimate sites, send request or open up telnet sessions with into prohibited servers or hosts that could for example fill up the users' assigned quarters or storage space.

3. Configuration errors - The attacker may gain access to a network with configuration errors. These weaknesses could be allowing access without prompting for any form of authentication such as assigning users guest accounts which are left without password requirements. Users may erroneously be assigned administrative privileges. A user may also obtain privileges of other users by obtaining their passwords or bypassing control that restrict access (Root attack) [10].

4. Masquerading - Having monitored the flow of traffic, the attacker can modify a TCP packet or originate a packet but send it with a forged source address to give the impression that it is from a trusted source. The attacker can therefore send Trojan, perform a denial of service or make request for sensitive data.

5. Worms - These are destructive self-replicating programs that spread across the network. They can be sent as e-mail attachments to hamper network performance.

6. Viruses - Are programs that replicate when a user performs some actions, such as running a program. The viruses may have the ability to destroy sensitive documents.

7. Implementation Bug - Most trusted applications and programs have bugs. These bugs are exploited by hackers to gain access to the computer systems or networks. Examples of these include: buffer over flows, race conditions or even mishandled files. This is also referred to as client attack, where the attacker may exploit bugs on the client, server, network software or protocol [10].

2.3 Security Countermeasures

The prevention of attacks and frauds is the first and most important line of defense [6]. By deploying preventive countermeasures its aim is to prevent security threats by eliminating as much vulnerabilities as possible [2]. In preventing malicious attacks and unauthorized usage of computers a number of technologies and techniques have existed and many Organizations are striving to maintain confidentiality, integrity and availability of their networked resources and a number of techniques have been employed to guard against network intrusion. However, even though these measures provide a level of security, they have been found to be lacking in a number of ways [11, 13].

1. The use of firewall. A firewall is a hardware or software solutions used to enforce security policy on a private network. It is mostly used to control traffic to or from a private network.

However, these are but just a list of permit and deny rules; therefore they may not always have the ability to detect intrusions.

Firewall, user authentication, data encryption and Virtual Private Networks (VPN) provide a level of security but they are limited by the fact that they cannot give protection against malicious codes, inside attacks or unsecured modems [11]. They therefore would only be effective as one of the available lines of defense.

2. For institutions that already have intrusion prevention systems, perfectly secure system are hard to come by. There are always a number of system flaws in addition to possible administrator configuration errors. Intrusion detection systems can thus be used to supplement the already existing systems.

3. Many times intrusion prevention systems are implemented on expensive routers. These are sometimes not flexible enough to allow changes such as in network design and topology. Intrusion detection systems provide the flexibility needed to provide adequate security without necessarily acquiring specialized hardware.

4. Cryptography hides information from unauthorized users; however this method makes it hard to know whether any attack has taken place. Generally key management is not an easy task. Crypto systems may require special key management systems such as the use of a Terminal Access Controller Access System (TACACS) or Remote Authentication Dial In User Service (RADIUS) server. This could mean specialized hardware or configuration. Otherwise hackers could gain access to these keys and break into the system.

5. The provision of physical security to the network site or to servers. However these are limited by the fact that physical security may not provide a practical solution to attackers who employ telnet sessions to gain access to a network.

6. Authentication. A technique used to verify users of a network resource. The effectiveness of this is weakened by the fact that many still ”use easy to crack passwords” while some user 3are either un trustworthy or are just careless with their passwords such that many times can easily be got by unauthorized users.

7. Many organizations have also employed anti viruses, however this may not provide information as to whether there has been an intrusion or not. Anti-viruses also require frequent updates. Denning also presented four basic motivations for IDS’s. He observed that most systems have security flows susceptible to intrusion, most of the systems are not easily replaceable, difficulty to develop perfectly secure systems and available systems are vulnerable to abuse such as by inside users [12]. This shows a greater need for intrusion detection systems.

Solutions	**Firewalls**	**Anti-Viruses**	**Intrusion Detection Systems (IDSs)**
Ability to control internal attacks	No	Limited way	Yes
Ability to log intrusion attempts	No	Limited way	Yes
Ability to provide a history for attacks	No	Yes	Yes
Ability to send alerts	No	No	Yes
Ability to detect attacks	No	Limited way	Yes
Ability to reacting to attacks	No	Limited way	Yes

TABLE 2.1: The abilities of intrusion detection system.

2.4 Intrusion Detection

An intrusion is defined as "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource" [16]. And another definition with the same sense is "the potential possibility of a deliberate unauthorized attempt to access information, manipulate information, or render a system unreliable or unusable" [17].

An intrusion is a violation of the security policy of the system. The definitions above are general enough to encompass all the threats any definition of intrusion is, of necessity, imprecise, as security policy requirements do not always translate into a well-defined set of actions whereas policy defines the goals that must be satisfied in a system, detecting breaches of policy requires knowledge of steps or actions that may result in its violation.

The intruders may be an entity from outside or may be an inside user of the system trying to access unauthorized information. Based upon this observation intruders can be widely divided into two categories; external intruders and internal intruders [2].

1. External intruders are those who don't have an authorized access to the system they are dealing with.

2. Internal intruders are those who have limited authorized access to the systems and they overstep their legitimate access rights.

Intrusion detection is the process of determining an intrusion into a system by the observation of the information available about the state of the system and monitoring the user activities. Detection of break-ins or attempts by intruders to gain unauthorized access of the system is intrusion detection.

2.4.1 Classification of intrusion detection system (IDS) based on different information source:

2.4.1.1 Host-based IDS

Its data comes from the records of various activities of hosts, including audit record of operation system, system logs, application programs, information, and so on. Taking Windows NT operation system as an example, its event logs mechanism searches and collects three patterns of system events: Operation system event, safety event and application event; and examples of application program information are as follows: Database system, WWW servers, and so on. Its advantage and disadvantage are stated as follows [20]:

Advantage:

1. It can judge whether or not the host is intruded more accurately: Because its data comes form system audit records and system logs of hosts, comparing with network based intrusion detection system, it can more accurately judge network attacks or intrusion on hosts.

2. It can detect attacks under encrypted network environment: Because the data comes from system files and transmitted encrypted data in network which are decrypted in hosts, thus the data is not affected.

3. It does not need additional hardware: It just needs monitoring system installed in specified hosts, without additional hardware.

Disadvantage:

1. Higher cost: Monitoring systems must be installed in each host; and because of different hosts, the audit files and log pattern are accordingly different, thus different intrusion detection systems are required in each host.

2. It may affect system efficiency of monitored hosts: Intrusion detection system in monitoring state may occupy system sources of hosts.

2.4.1.2 Network-based IDS [21]

Its data is mainly collected network generic stream going through network segments, such as: Internet packets. And its advantage and disadvantage are stated as follows:

Advantage:

1. Low cost: Only network-based IDS can detect all attacks in a LAN, and the cost is just for the device.

2. It can detect attacks that cannot be done by host-based IDS, such as denial of service.

Disadvantage:

1. The flux is large, and some packets may be lost, and it cannot detect all packets in network.

2. In large-scale network, it requires more rapid CPU and more memory space, to analyze bulk data.

3. It cannot deal with encrypted packets, and it may not receive attack information in encrypted packets accordingly.

2.4.2 Classification of intrusion detection based on different detection method

2.4.2.1 Misuse Detection

It is also named signature-based detection, which can transform the information of attack symptom or policy disobeying into state transition-based signature or rule, and such information is stored in signature database. To judge whether or not it is attack, pre-treated case data should be first compared with the signature of signature database, and those conforming to attack signature data can be judged as attack [21]. Its advantage is high detection rate and low false alarm rate for known attacks; however, its detection capacity is low for unknown detection methods, and attack database should be renewed on a regular basis.

2.4.2.2 Anomaly Detection

It may establish a profiles for normal behavior of users, which comes from statistics data of users in the former period; when detection is performed, the profiles is compared with actual users' data, if the offset is below threshold value, user's behavior can be considered normal, and it has no intention of attacks; if the offset is above threshold value, user's behavior can be considered abnormal [21].

Anomaly detection is based on an assumption that intruder's behavior is different from normal users' behavior. Detection rate of the method is high, and it is more likely to detect unknown attacks, but misjudgment (false positive) rate is also high.

2.4.2.3 Hybrid

It is also possible to include both normal and attack data in the model. Such systems are referred to be hybrid detectors.

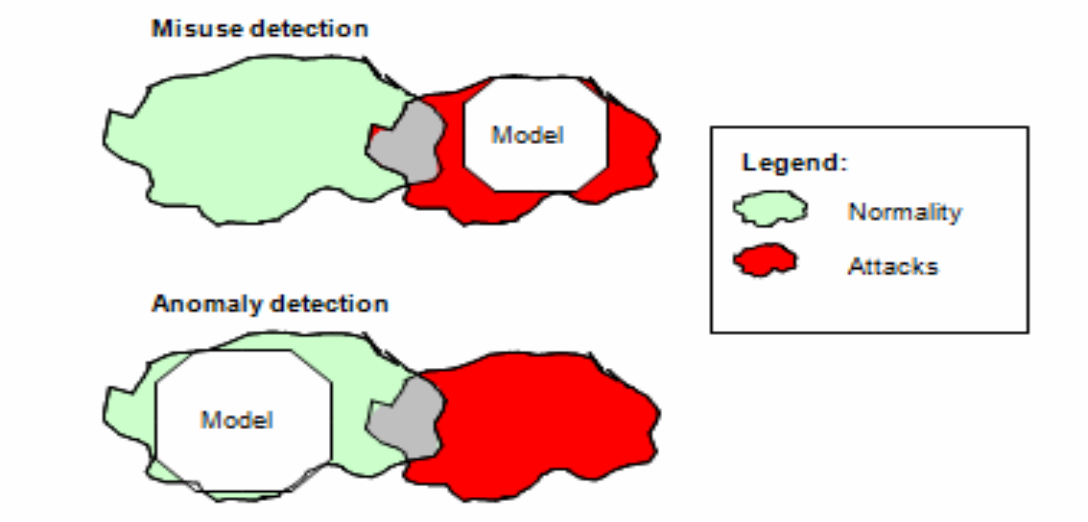

FIGURE 1: The relation between misuse detection and anomaly detection.

2.5 Intrusion Dataset

NSL-KDD is a data set suggested to solve some of the inherent problems of the KDD Cup 99 data set [58]. It can be applied as an effective benchmark data set to help researchers compare different intrusion detection methods. Furthermore, the number of records in the NSL-KDD train and test sets are reasonable. This advantage makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research work will be consistent and comparable.

Since NSL-KDD dataset is the new version of the original KDD Cup 99 dataset, it is better to introduce the KDD Cup 99 under this section and for detail explanations about NSL-KDD dataset see chapter four.The KDD Cup 1999 Data is original from 1998 DARPA Intrusion Detection Evaluation [18]. In the 1998 DARPA intrusion detection evaluation program, an environment was set up to acquire raw TCP/IP dump data for a network by simulating a typical US Air Force LAN.

The LAN was operated like a real environment, but being blasted with multiple attacks. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted [19] and the features of the KDD dataset are listed below in table 2.2.

No	Features
1	duration
2	protocol_type
3	service
4	flag
5	src_bytes
6	dst_bytes
7	Land
8	wrong_fragment
9	urgent
10	hot
11	Num_failed_logins
12	logged_in
13	Num_compromised
14	Root_shell
15	su_attempted
16	Num_root
17	Num_file_creations
18	Num_shells
19	Num_access_files
20	Num_outbound_cmds
21	is_host_login
22	is_guest_login
23	count
24	srv_count
25	serror_rate
26	srv_serror_rate
27	rerror_rate
28	srv_rerror_rate
29	same_srv_rate
30	diff_srv_rate
31	srv_diff_host_rate
32	dst_host_count
33	dst_host_srv_count
34	dst_host_same_srv_rate
35	dst_host_diff_srv_rate
36	dst_host_same_src_port_rate
37	dst_host_srv_diff_host_rate
38	dst_host_serror_rate

39	dst_host_srv_serror_rate
40	dst_host_rerror_rate
41	dst_host_srv_rerror_rate

TABLE 2.2: Features of KDD dataset.

KDD Cup 1999 Data has 41 attributes this dataset has been used for the Third International Knowledge Discovery and Data Mining Tools Competition. The task of this competition was to build a network detector to find "bad" connections and "good" connections. For "bad" connections, the attack has categories [18], the four different categories of attack patterns are as follows [19] (see table 4.3).

Probing

Probing is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits. There are different types of probes: some of them abuse the computer's legitimate features; some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise.

Denial of service attacks (DOS)

DOS is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine There are different ways to launch DoS attacks: by abusing the computers legitimate features; by targeting the implementations bugs; or by exploiting the system's misconfigurations. DoS attacks are classified based on the services that an attacker renders unavailable to legitimate users.

User to root attacks (U2R)

User to root exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Most common exploits in this class of attacks are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions.

Remote to user attacks (R2L)

A remote to user (R2L) attack is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine's vulnerability to illegally gain local access as a user. There are different types of R2L attacks: the most common attack in this class is done using social engineering.

Chapter Three

Data mining and cost sensitive learning

3.1 Overview

Mining information and knowledge from large databases has been recognized by many researchers as a key research topic in database systems and machine learning and by many industrial companies as an important area with an opportunity of major revenues. Researchers in many different fields have shown great interest in data mining. Several emerging applications in information providing services, such as data warehousing and on-line services over the Internet, also call for various data mining techniques to better understand user behavior, to improve the service provided, to improve network security, and to increase the business opportunities[35].

Data mining, also known as Knowledge Discovery in Databases (KDD) has been recognized as a rapidly emerging research area. This research area can be defined as efficiently discovering human knowledge and interesting rules from large databases. Data mining involves the semiautomatic discovery of interesting knowledge, such as patterns, associations, changes, anomalies and significant structures from large amounts of data stored in databases and other information repositories [35].

Data mining is an information extraction activity whose goal is to discover hidden facts contained in databases. Using a combination of machine learning, statistical analysis, modeling techniques and database technology, data mining finds patterns and subtle relationships in data and infers rules that allow the prediction of future results.

For every data mining system, a data preprocessing step is one of the most important aspects. Data preprocessing consumes very great amount of time for a typical, real world data mining effort. Poor quality of data may lead to nonsensical data mining results which will subsequently have to be discarded. Data preprocessing concerns the selection, evaluation, cleaning, enrichment, and transformation of the data.

Data preprocessing involves the following aspects: [36].

1. Data cleaning is used to ensure that the data are of a high quality and contain no duplicate values. The data cleaning process involves the detection and possible elimination of incorrect and missing values.
2. Data integration. When integrating data, historic data and data referring to day-to-day operations are merged into a uniform format.
3. Data selection involves the collection and selection of appropriate data. The data are collected to cover the widest range of the problem domain.
4. Data transformation involves transforming the original data set to the data representations of the individual data mining tools.

3.2 Data Mining and Intrusion Detection

Data Mining is assisting various applications for required data analysis. Recently, data mining is becoming an important component in intrusion detection system. Different data mining approaches like classification, clustering, association rule, and outlier detection are frequently used to analyze network data to gain intrusion related knowledge. This section will elaborate on several of these data mining techniques and will describe how they are used in the context of intrusion detection.

3.2.1 Clustering

The amount of available network audit data instances is large, human labeling is time-consuming, and expensive. Clustering is the process of labeling data and assigning it into groups. Clustering algorithms can group new data instances into similar groups. These groups can be used to increase the performance of existing classifiers. High quality clusters can also assist human expert with labeling.

Clustering discovers complex intrusions occurred over extended periods of time and different spaces, correlating independent network events [49]. The sets of data belonging to the cluster are modeled according to pre-defined metrics and their common features [42]. It is used to detect hybrids of attack in the cluster. Clustering is an unsupervised machine learning mechanism for finding patterns in unlabeled data with many dimensions. K-means clustering is used to find natural groupings of similar alarm records. Records that are far from any of these clusters indicate unusual activity that may be part of a new attack. The network data available for

intrusion detection is primarily categorical with attributes having a small number of unordered values [47].

The steps involved in identifying intrusion in using clustering techniques are as follows [49]:

1. Find the largest cluster, i.e., the one with the most number of instances, and label it normal.

2. Sort the remaining clusters in an ascending order of their distances to the largest cluster.

3. Select the first K1 clusters so that the number of data instances in these clusters sum up to ¼ ´N, and label them as normal, where ´ is the percentage of normal instances.

4. Label all the other clusters as attacks.

Unlike traditional anomaly detection methods, they cluster data instances that contain both normal behaviors and attacks, using a modified incremental k-means algorithm. After clustering, heuristics are used to automatically label each cluster as either normal or attacks. The self-labeled clusters are then used to detect attacks in a separate test dataset.

3.2.2 Association rules

Association rules [51, 52] are one of many data mining techniques that describe events that tend to occur together. The concept of association rules can be understood as follows: Given a database D of transactions where each transaction $T \in \mathsf{D}$ denotes a set of items in the database, an association rule is an implication of the form $X \_ Y$, where $X \subset \mathsf{D}$, $Y \subset \mathsf{D}$ and $X \cap Y = \varnothing$. The rule X _Y holds in the transaction set D with confidence c if c% of transactions in X also contains Y. Two important concepts when dealing with association rules are rule confidence and rule support. The probability of rule confidence is defined as the conditional probability $P(Y \subseteq T \mid$

$X \subseteq T$. The rule X _Y has support s in the transaction database D if s% of transactions in D contains $X \cup Y$. Association rules have been successfully used to mine audit data to find normal patterns for anomaly detection [54, 50]. They are particularly important in the domain of anomaly detection because association rules can be used to construct a summary of anomalous connections detected by the intrusion detection system.

Association rules show attributes value conditions that occur frequently together in a given dataset. A typical and widely-used example of association rule mining is Market Basket Analysis. Association rules provide information of this type in the form of "if-then" statements [51]. These rules are computed from the data. Association rules are probabilistic in nature. In addition to the antecedent (the "if" part) and the consequent (the "then" part), an association rule has two numbers that express the degree of uncertainty about the rule. In association analysis the antecedent and consequent are sets of items that are disjoint. The first number is called the support for the rule. The support is implying the number of transactions that include all items in the antecedent and consequent parts of the rule. The other number is known as the confidence of the rule. Confidence is the ratio of the number of transactions that include all items in the consequent as well as the antecedent to the number of transactions that include all items in the antecedent [81].

Many association rule algorithms have been developed in the last decades, which can be classified into two categories [37, 46]:

1. candidate-generation-and-test approach such as Apriori and

2. pattern-growth approach.

The challenging issues of association rule algorithms are multiple scans of transaction databases and a large number of candidates. Apriori was the first scalable algorithm designed for association-rule mining algorithm [51]. The Apriori algorithm searches for large itemsets during its initial database pass and uses its result as the basis for discovering other large datasets during subsequent searches.

Basic steps for incorporating association rule for intrusion detection as follows [37].

1. First network data need to be formatted into a database table where each row is an audit record and each column is a field of the audit records.

2. There is evidence that intrusions and user activities shows frequent correlations among network data. For example, one of the reasons that "program policies", which codify the access rights of privileged programs, are concise and capable to detect known attacks is in that the intended behavior of a program, e.g., read and write files from certain directories with specific permissions is very consistent. These consistent behaviors can be captured in association rules.

3. Also rules based on network data can continuously merge the rules from a new run to the aggregate rule set of all previous runs. Thus with the association rule, it has the capability to capture behavior in association rule for correctly detecting intrusion and hence lowering the false alarm rate.

3.2.3 Classification

An intrusion detection system that classifies audit data as normal or anomalous based on a set of rules, patterns or other affiliated techniques can be broadly defined as a classification-based intrusion detection system.

Classification is similar to clustering in that it also partitions customer records into distinct segments called classes. But unlike clustering, classification analysis requires that the end-user/analyst know ahead of time how classes are defined. It is necessary that each record in the dataset used to build the classifier already have a value for the attribute used to define classes. As each record has a value for the attribute used to define the classes. The objective of a classifier is not to explore the data to discover interesting segments, but to decide how new records should be classified. Classification is used to assign examples to predefined categories. Machine learning software performs this task by extracting or learning discrimination rules from examples of correctly classified data [46].

Classification models can be built using a wide variety of algorithms. Classification categorizes the data records in a predetermined set of classes used as attribute to label each record; distinguishing elements belonging to the normal or abnormal class. This technique has been popular to detect individual attacks but has to be applied with complementary fine tuning techniques to reduce its demonstrated high false positives rate.

Data classification for intrusion detection can be achieved by the following basic steps [44].

1. In order for a machine learning program to learn the classification models of the normal and abnormal system call sequences, it needs to be supplied with a set of training data containing pre-labeled normal and abnormal sequences. Different mechanisms based on either linear discrimination, decision tree or rule based methods can be used to scan the normal network traces and create a list of unique sequences of system calls. This list is generally named as normal list.

2. Next step is to scan each of the intrusion traces. For each sequence of system calls, first look it up in the normal list. If an exact match can be found then the sequence is labeled as normal otherwise it is labeled as abnormal.

3. Then ensure that the normal traces include nearly all possible normal short sequences of system calls. An Intrusion trace contains many normal sequences in addition to the abnormal sequences since the illegal activities only occur in some places within a trace.

3.2.4 Outlier Detection

An outlier is an infrequent observation that immensely deviates from the characteristic distribution of other observations. Outlier detection has many applications, such as data cleaning, fraud detection and network intrusion [60].

The existence of outliers indicates that individuals or groups that have very different behavior from most of the individuals of the dataset. Many times, outliers are removed to improve accuracy of the estimators [42].

Most anomaly detection algorithms require a set of purely normal data to train the model. They assume that anomalies can be treated as previously unobserved patterns. Since an outlier may be defined as a data point which is very different from the rest of the data, it is possible to employ several outlier detection schemes for intrusion detection which are based on statistical measures, clustering methods and data mining methods [60].

Outlier detection is very useful in anomaly based intrusion detection. With outlier detection approach, it can detect novel attack/intrusion by identifying them as deviation from normal behavior. The basic steps in detecting intrusion based on outlier detection are as follows.

1. As outlier detection technique is used in anomaly detection, it first needs to identify normal behavior. This behavior can be data set or pattern of some events on the network.

2. Then useful set of feature need to be constructed and,

3. Similarity function needs to be defined between them.

It will need to run specific outlier detection algorithm on the set of feature. The algorithm can be based on a statistical based approach, a distance based approach, or a model based schema. All these approaches are based on finding the deviation between collected and scanned data sets [48]. In case of intrusion detection, the collected set of data set will be the set of events and their relation to intrusion. Such relation can be calculated based on normal behavior and any other behavior which significantly deviates from normal behavior. As with such deviation it can preempt attacks based on their behavioral deviation.

Outlier detection approaches can useful for detecting any unknown attacks. This is the primary reason that makes outlier detection a popular approach for intrusion detection systems. Statistical based outlier detection scheme uses a probabilistic model as representation of underlying mechanism of data generation [60]. Such probabilistic model can be useful in intrusion detection environment to decide the probability before alarming the system for intrusion.

Distance based outlier detection approaches such as Nearest Neighbor approach are engaged in finding outlier that do not have enough neighbors per define density of local neighborhood. Such outlier detection is very useful in anomaly based intrusion detection systems that are involved in detecting abnormal behavior or deviating patterns [45]. Such density based and distance based approaches can help to identify abnormal behavior from the set of normal behavior and enable to detect any unknown intrusions [60].

3.3 Cost sensitive learning and intrusion detection

3.3.1 Why cost sensitive learning

Standard data mining algorithms (cost insensitive learning) try to make the classification decision to minimize the probability of error. However, for many problem domains, the requirement is not merely to predict the most probable class label, since different types of errors carry different costs.

For instance, diagnosing disease for a healthy person does not produce the same consequences as to predict health for an ill person and another example of such problem in the case of computer network security include authentication, where the cost of allowing unauthorized access can be much greater than that of wrongly denying access to authorized individuals, and intrusion detection, where raising false alarms has a substantially lower cost than allowing an undetected intrusion. In such cases, it is preferable to make the classification decision that has minimum expected cost, rather than that with the lowest error probability.

Another case where cost sensitive learning is very important is if class imbalanced datasets occurs but this happens in many real-world applications where the class distributions of data are highly imbalanced [33]. Again, it is assumed that the minority or rare class is the positive class, and the majority class is the negative class. Often the minority class is very small, such as 2% of the dataset. If most traditional (cost insensitive) classifiers are applied on the dataset, they will likely to predict everything as negative (the majority class). This was often regarded as a problem in learning from highly imbalanced datasets.

However, two fundamental assumptions are often made in the traditional cost-insensitive classifiers [33]. The first is that the goal of the classifiers is to maximize the accuracy (or minimize the error rate); the second is that the class distribution of the training and test datasets is the same. Under these two assumptions, predicting everything as negative for a highly imbalanced dataset is often the right thing to do. It is usually very difficult to outperform this simple classifier in this situation [24].

Thus, the imbalanced class problem becomes meaningful only if one or both of the two assumptions above are not true; that is, if the cost of different types of error (false positive and false negative in the binary classification) is not the same, or if the class distribution in the test data is different from that of the training data.

In the case when the misclassification cost is not equal, it is usually more expensive to misclassify a minority (positive) example into the majority (negative) class, than a majority example into the minority class (otherwise it is more plausible to predict everything as negative). In case the class distributions of training and test datasets are different (for example, if the training data is highly imbalanced but the test data is more balanced), an obvious approach is to sample the training data such that its class distribution is the same as the test data (by over sampling the minority class and/or under sampling the majority class) [33].

Note that sometimes the number of examples of the minority class is too small for classifiers to learn adequately. This is the problem of insufficient (small) training data, different from that of the imbalanced datasets.

In this case as it has been mentioned in the previous chapter the intrusion data , KDD data set , used for this research has different class distributions of training and test datasets and each attack type has different cost , so cost sensitive learning is better solution for computer network intrusion detection.

3.3.2 Cost-sensitive learning

Cost-Sensitive Learning is a type of learning in data mining that takes the misclassification costs into consideration. The goal of this type of learning is to minimize the total cost. The key difference between cost-sensitive learning and cost-insensitive learning is that cost-sensitive learning treats the different misclassifications differently. Cost-insensitive learning does not take the misclassification costs into consideration.

The goal of this type of learning is to pursue a high accuracy of classifying examples into a set of known classes. The class imbalanced datasets occurs in many real-world applications where the class distributions of data are highly imbalanced. Cost-sensitive learning is a common approach to solve this problem [26].

Classification is the most important task in inductive learning and machine learning. A classifier can be trained from a set of training examples with class labels, and can be used to predict the class labels of new examples. The class label is usually discrete and finite. Many effective classification algorithms have been developed, such as naïve Bayes, decision trees, neural networks, and so on. However, most original classification algorithms pursue to minimize the error rate: the percentage of the incorrect prediction of class labels. They ignore the difference between types of misclassification errors [26]. In particular, they implicitly assume that all

misclassification errors cost equally. In many real-world applications, this assumption is not true. The differences between different misclassification errors can be quite large.

For example, in medical diagnosis of a certain cancer, if the cancer is regarded as the positive class, and non-cancer (healthy) as negative, then missing a cancer (the patient is actually positive but is classified as negative; thus it is also called "false negative") is much more serious (thus expensive) than the false-positive error. The patient could lose his/her life because of the delay in the correct diagnosis and treatment" [30].

Cost-sensitive learning takes costs, such as the misclassification cost, into consideration. It is one of the most active and important research areas in machine learning, and it plays an important role in real-world data mining applications. [30] provides a comprehensive survey of a large variety of different types of costs in data mining and machine learning, including misclassification costs, data acquisition cost (instance costs and attribute costs), active learning costs, computation cost, human-computer interaction cost, and so on. The misclassification cost is singled out as the most important cost, and it has also been mostly studied in recent years [26].

3.3.2.1 Theoretical backgrounds

The following is the theoretical backgrounds of cost sensitive learning summarized from different literature [25, 31].

One of the most common practical approaches to cost sensitive classification is to change the class distribution of the training data with respect to the cost function and then present an error based learning algorithm with those modified data (misclassification cost). The hope is that the bias introduced on the training data would be able to output a hypothesis that minimizes the overall costs of the decisions for unknown future examples.

A simple approach is so called rebalancing (or stratification), i.e. changing the frequency of classes in the training data in proportion to their cost [25, 31]. For simplicity, let us take a two-class classification problem as an example to illustrate the method as shown in table 3.1.

In cost-sensitive learning, the costs of false positive (actual negative but predicted as positive; denoted as FP), false negative (FN), true positive (TP) and true negative (TN) can be given in a cost matrix, as shown in Table 3.1. In the table, it also used the notation C(i, j) to represent the misclassification cost of classifying an instance from its actual class j into the predicted class i. (it is used 1 for positive, and 0 for negative) .

These misclassification cost values can be given by domain experts, or learned via other approaches. In cost-sensitive learning, it is usually assume that such a cost matrix is given and known. For multiple classes, the cost matrix can be easily extended by adding more rows and more columns [31].

Parameter	Actual negative	Actual positive
Predict negative	C(0,0), or TN	C(0,1), or FN
Predict positive	C(1,0), or FP	C(1,1), or TP

TABLE 3.1: An example of cost matrix for binary classification.

Note that C(i, i) (TP and TN) is usually regarded as the "benefit" (i.e., negated cost) when an instance is predicted correctly. In addition, cost-sensitive learning is often used to deal with datasets with very imbalanced class distribution [26]. Usually the minority or rare class is regarded as the positive class, and it is often more expensive to misclassify an actual positive example into negative, than an actual negative example into positive. That is, the value of FN or C(0,1) is usually larger than that of FP or C(1,0). This is true for the HIV example, HIV patients are usually rare in the population, but predicting an actual HIV patient as negative is usually very costly [25, 31].

Given the cost matrix, an example should be classified into the class that has the minimum expected cost [31]. This is the minimum expected cost principle. The expected cost R(i|x) of classifying an instance x into class i (by a classifier) can be expressed as:

$$R(i \mid x) = \sum_{j} P(j \mid x) C(i, j),$$

Where P(j|x) is the probability estimation of classifying an instance into class j. That is, the classifier will classify an instance x into positive class if and only if:

$$P(0|x)C(1,0) + P(1|x)C(1,1) \leq P(0|x)C(0,0) + P(1|x)C(0,1)$$

This is equivalent to:

$$P(0|x)(C(1,0) - C(0,0)) \leq P(1|x)(C(0,1) - C(1,1))$$

Thus, the decision (of classifying an example into positive) will not be changed if a constant is added into a column of the original cost matrix. Thus, the original cost matrix can always be converted to a simpler one by subtracting C(0,0) to the first column, and C(1,1) to the second column. After such conversion, the simpler cost matrix is shown in Table 3.2.

Thus, any given cost matrix can be converted to one with C(0,0) = C(1,1) = 0. It assume that C(0,0) = C(1,1) = 0. Under this assumption, the classifier will classify an instance x into positive class if and only if:

$$P(0|x)C(1,0) \leq P(1|x)C(0,1)$$

Parameter	**True negative**	**True positive**
Predict negative	0	C(0,1) – C(1,1)
Predict positive	C(1,0) – C(0,0)	0

TABLE 3.2: A simpler cost matrix with an equivalent optimal classification.

As P(0|x)=1 – P(1|x), it can be obtained a threshold p* for the classifier to classify an instance x into positive if P(1|x) ≥ p*, where

$$p^* = \frac{C(1,0)}{C(1,0)+C(0,1)} = \frac{FP}{FP+FN}.$$

Thus, if a cost-insensitive classifier can produce a posterior probability estimation p(1|x) for test examples x, this can make it cost-sensitive by simply choosing the classification threshold according to the above formula, and classify any example to be positive whenever P(1|x) ≥ p.

3.3.2.2 Direct method versus Meta learning

Cost-sensitive learning can be categorized into two categories. The first one is to design classifiers that are cost-sensitive in themselves. They are called the direct method. Examples of direct cost-sensitive learning are ICET [29] and cost-sensitive decision tree [24, 27]. The other category is to design a wrapper that converts any existing cost-insensitive classifiers into cost-sensitive ones. The wrapper method is also called cost-sensitive meta-learning method, and it can be further categorized into thresholding and sampling. Since direct methods takes expected cost directly for decision making they are better than Meta learning (Note that direct cost sensitive learning directly takes costs into model building; they can also take easily attribute costs directly into consideration, while Meta cost sensitive learning algorithms generally cannot) [29].

Here is the summery of some of the research papers on the cost-sensitive learning and typical methods used.

Cost-sensitive learning

- Direct methods
    - ICET [29]
    - Cost-senstive decision trees [31]
- Meta-learning

- MetaCost [23]
- Cost-sensitive naïve Bayes [22]
- Empirical Thresholding [28]
- Costing [32]

3.3.2.3 Direct Cost-sensitive Learning

The main idea of building a direct cost-sensitive learning algorithm is to directly introduce and utilize misclassification costs into the learning algorithms. Note that direct cost sensitive learning directly takes costs into model building; they can also take easily attribute costs directly into consideration, while Meta cost sensitive learning algorithms generally cannot [29].

Related works on direct method

There are several works on direct cost-sensitive learning algorithms, such as ICET [29] and cost-sensitive decision trees [31].

Turney [29] incorporates misclassification costs in the fitness function of genetic algorithms ICET. On the other hand, cost-sensitive decision tree by Ling et al. [31], called CSTree here, uses the misclassification costs directly in its tree building process. That is, instead of minimizing entropy in attribute selection as in IDS, CSTree selects the best attribute by the expected total cost reduction. That is, an attribute is selected as a root of the (sub)tree if it minimizes the total misclassification cost.

Note that as both ICET and CSTree directly take costs into model building, they can also take easily attribute costs (and perhaps other costs) directly into consideration, while meta cost-sensitive learning algorithms generally cannot.

Drummond and Holte [31] investigate the cost-sensitivity of the four commonly used attribute selection criteria of decision tree learning: accuracy, Gini, entropy, and DKM. They claim that the sensitivity of cost is highest with the accuracy, followed by Gini, entropy, and DKM.

The cost sensitive learning algorithms that are used for this research are direct cost sensitive learning algorithms i.e. Cost sensitive classification tree CS-CRT and cost sensitive decision tree CS-MC4 algorithms which are not explored before.

CS-MC4 is the cost sensitive version of C4.5 that use misclassification cost matrix to minimize the expected cost and for the detection of best prediction and CS-CRT is cost sensitive version of CART. The basic ideas of cost sensitive are discussed in the previous pages and for further details about C4.5 and CART refer other literatures since C4.5 and CART are commonly used and widely known algorithms.

3.3.2.4 Cost-Sensitive Meta-Learning

Cost-sensitive meta-learning converts existing cost-insensitive classifiers into cost-sensitive ones without modifying them. Thus, it can be regarded as a middleware component that pre-processes the training data, or post-processes the output, from the cost-insensitive learning algorithms.

Cost-sensitive meta-learning can be further classified into two main categories: thresholding and sampling.

3.3.2.4.1 Thresholding

It uses as a threshold to classify examples into positive or negative if the cost insensitive classifiers can produce probability estimations. MetaCost is a thresholding method [23]. It first uses bagging on decision trees to obtain reliable probability stimations of training examples, relabels the classes of training examples according to and then uses the relabeled training instances to build a cost-insensitive classifier.

To predict the class of test instances it uses a cost-insensitive algorithm to obtain the probability estimations P(j|x) of each test instance. Then it uses to predict the class label of the test examples. All thresholding-based meta-learning methods replies on accurate probability estimations of p(1|x) for the test example x. To achieve this, several methods can be used to improve the calibration of probability estimates [23].

3.3.2.4.2 Sampling

On the other hand, sampling first modifies the class distribution of training data according to, and then applies cost-insensitive classifiers on the sampled data directly. There is no need for the classifiers to produce probability estimations, as long as it can classify positive or negative examples accurately. [32] Proportional sampling with replacement produces duplicated cases in the training, which in turn produces over fitting in model building.

Chapter Four

Experimentations and Discussions

We used data mining software tool known as Tanagra version 1.4.34 available freely at [18]. The software is a GUI based software and easy to use. Tanagra is capable of classifying large volumes of data within a second depending on the speed and specification of computer processor.

All experiments were performed using an Intel Core 2 Duo Processor 2.16 GHz processor with 4 GB of RAM and implemented on a Vista Windows operating system.

4.1 NSL-KDD Intrusion Detection Dataset

We have used NSL-KDD intrusion dataset which is available at [56] and it is a dataset suggested to solve some of the inherent problems of the original KDD Cup 99 dataset [58,59]. The NSL-KDD dataset has the following advantages over the original KDD Cup 99 dataset (see chapter three under title intrusion dataset for more detail about KDD Cup 99 dataset).

1. It does not include redundant records in the training set, so the classifiers will not be biased towards more frequent records.

2. There are no duplicate records in the proposed test sets, therefore the performance of the learners are not biased by the methods which have better detection rates on the frequent records.

3. The number of selected records from each difficulty level group is inversely proportional to the percentage of the records in the original KDD dataset. As a result the classification rates of

distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.

4. The number of records in the training and testing sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion.

5. Statistical observations, one of the most important deficiencies in the KDD dataset is the huge number of redundant records, which causes the learning algorithms to be biased towards the frequent records and thus prevent them from learning unfrequent records which are usually more harmful to networks such as U2R and R2L attacks.

Table 4.1 and Table 4.2 shows the statistics of the redundant records in the KDD Cup 99 training and testing datasets.

Original	**Records**	**Distinct Records**	**Reduction Rate**
Attacks	3,925,650	262,178	93.32%
Normal	972,781	812,814	16.44%
Total	4,898,431	1,074,992	78.05%

TABLE 4.1: Statistics of redundant records in the KDD training dataset.

Original	Records	Distinct Records	Reduction Rate
Attacks	250,436	29,378	88.26%
Normal	60,591	47,911	20.92%
Total	311,027	77,289	75.15%

TABLE 4.2: Statistics of redundant records in the KDD testing dataset

4.2 Data Pre-Processing

Data preprocessing can be described as any type of processing performed on the raw data to prepare it for any other processing procedure. Data preprocessing is an important step in data mining. It transforms the data into a format that is acceptable to the actual data processing algorithm.

Features in the NSL-KDD dataset have mixed data types (i.e., continuous, discrete and nominal) with a highly skewed distribution of values (see Appendix C). The data was pre-processed as follows:

1. The requirement for CS-MC4 and CS-CRT is that the input values can have discrete or continuous value, so no farther processing is done in this regard to change the value of the inputs.

2. Each connection record in the data is labeled either as normal or attacks. There is a total of thirty-nine intrusion types including both training and test datasets (see appendix D). All of these types can be classified into four major categories [57].

All attack in the training and the test data set were mapped to their respective categories as shown below on the table 4.3(see appendix D for detail).

Category	**Attack Types**
Probe	ipsweep, mscan, nmap, portsweep, satan, saint
DOS	apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm
U2R	buffer_overfow, httptunnel, loadmodule, perl, ps, rootkit, sqlattack, xterm
R2L	ftp write, guess passwd, imap, multihop, named, phf, sendmail, nmpgetattack, snmpguess, spy, warezclient, warezmaster, worm, xlock, xsnoop

TABLE 4.3: Attack types in NSL_KDD data and their categorization.

3. The NSL-KDD dataset is available in text format to change into arff format which can be read by the Tanagra tool, header information is added in both the training and test datasets at the top of each files (see appendix A).

4. Table 4.4 and 4.5 show the number of instances in the full NSL-KDD training datasets as well as the test dataset along with their respective percentages. All attack types are mapped to their respective categories while the non-intrusive instances are labeled as normal.

Category	Instances	Percentage
Normal	67,343	53.5%
DOS	45,927	36.5%
Probe	11,656	9.3%
R2L	995	0.8%
U2R	52	0.04%
TOTAL	125,973	100%

TABLE 4.4: Number of instances in the full NSL-KDD Training data set.

Category	Instances	Percentage
Normal	9,711	43.1%
DOS	7,458	33.1%
Probe	2,421	10.7%
R2L	2,754	12.2%
U2R	200	0.9%
TOTAL	22,544	100%

TABLE 4.5: Number of instances in the full NSL-KDD test dataset.

4.3 Evaluation Metrics

We have used the cost matrix defined for the KDD Cup 99 Dataset [55] which is shown in Table 4.6.

Category	**Normal**	**Probe**	**DOS**	**U2R**	**R2L**
Normal	0	1	2	2	2
Probe	1	0	2	2	2
DOS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

TABLE 4.6: Cost matrix defined for the KDD 1999 Dataset.

4.4 Performance measure

The most commonly used performance measure used to judge the accuracy of a network intrusion detection system are as follows [54]. The network IDS estimates parameters shown in Table 4.7 and figure 2.

Parameter	Definition
True Positive Rate (TP)	Attack occur and alarm raised
False Positive Rate (FP)	No attack but alarm raised
True Negative Rate (TN)	No attack and no alarm
False Negative Rate (FN)	Attack occur but no alarm

TABLE 4.7: Parameters used for performance measure.

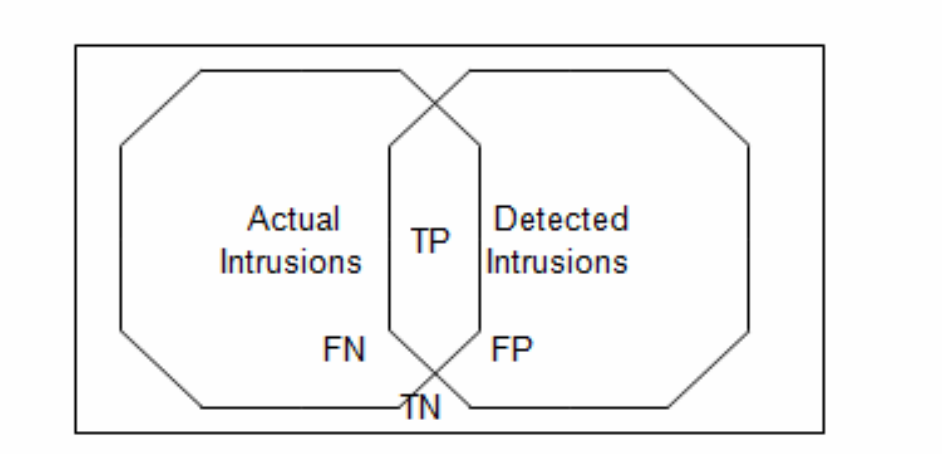

FIGURE 2: Parameters used for performance measure.

Precision stands for that the attack has been occurred and the IDS detects correctly. This formula use TP to divide TP adds FP to find precision rate.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall stands for an attack is happened and IDS detects attacks from really attacks. This formula use TP divide TP adds FN to find recall value shown in formula below.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Overall accuracy formula shows when the attack has been occurred, IDS detects attacks correctly. This formula use TP adds TN to divide TP adds FP adds FN adds TN to estimate overall accuracy which is shown in formula below.

$$\text{Overall} = \frac{TP + TN}{TP + FP + FN + TN}$$

The false alarm (false positives rate) is defined as an intrusion has been occurred but IDS cannot detect correctly or intrusion happens. The formula uses FP adds FN to divide TP adds FP adds FN adds TN to calculate false alarm rate which is shown below.

$$\text{False Alarm} = \frac{FP + FN}{TP + FP + FN + TN}$$

The average misclassification cost assigned to each classification model using a cost matrix (given in Table 4.6) based on the rarity of the classes. The average misclassification cost is computed by multiplying the cost matrix and confusion matrices and dividing the sum of the resultant matrix by the total number of test instances. A lower average misclassification cost means better classification model.

To compare the performance of the classifiers, three important formulas are used to evaluate methods. These are over all accuracy, false positive rate and average misclassification cost.

4.5 KDD1999 winner results

The winner of KDD1999 [23] uses cost-sensitive bagged boosting of decision trees with the results in table 4.8. We used these results to compare our approaches.

Actual	**Predicted**					
	0	1	2	3	4	%correct
0	60262	243	78	4	6	99.5%
1	511	3471	184	0	0	83.3%
2	5299	1328	223226	0	0	97.1%
3	168	20	0	30	10	13.2%
4	14527	294	0	8	1360	8.4%
%correct	74.6%	64.8%	99.9%	71.4%	98.8%	

TABLE 4.8: Results of KDD99 winner.

Symbol	**Definition**
0	Normal
1	Probe
2	denial of service (DOS)
3	user-to-root (U2R)
4	remote-to-local (R2L)

TABLE 4.9: The categories of network attacks.

Other Performance Measures for KDD99 winner

Overall Accuracy = 92.71%

False alarm rate = 0.55%

Average misclassification cost = 0.2331

4.6 Experimentations

The experimentations have two major parts; the first one is experimentation on CS-MC4 and CS-CRT using all the 41 attributes and the second is experimentation on CS-MC4 and CS-CRT using feature (attribute) selection method. Comparative discussions of each approaches used with the KDD 99 winner results are given in the discussion part.

In all these experimentations we have used the default parameters as it is in Tanagra tool for both CS-MC4 and CS-CRT algorithms in order to compare and measure the performances of the algorithms without any modification as shown below in the tables.

Cost sensitive C4.5 parameters	
Min size of leaves	5
Lambda for laplacian	3.00

TABLE 4.10: Parameters used for CS-MC4.

Classification tree (C-RT) parameters	
Size before split	10
Pruning set size (%)	33
x-SE rule	1.00
Random generator	1
Show all tree seq (even if > 15)	1

TABLE 4.11: Parameters used for CS-CRT.

4.6.1 Experimentation I: Using all the Attributes

These experimentations have two phases namely, learning and classifying training data and then classifying the testing data. In the first phase, full training dataset of NSL-KDD are used to construct a detection model using CS-MC4 and CS-CRT algorithms. In the second phase, the full testing dataset of NSL-KDD passed through the trained model to detect the intrusions and find the detection error rates, precision, false positive, average misclassification cost and accuracy of the detection model.

We have taken full training dataset of NSL-KDD as input to the system which contains about 125,973 connection records to train the system and a full test dataset which contains 22,544 connection records to test the system. The detection rate in each phase and the classification performance of CS-MC4 and CS-CRT on the NSL-KDD test dataset using all the 41 attributes is presented below.

4.6.1.1 CS-MC4 Experiments

TABLE 4.12: CS-MC4 performance using all the attributes in the training and testing phase.

Error rate	0.0080
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
Normal	0.9896	0.0025
DOS	0.9990	0.0012
R2L	0.9488	0.0439
Probe	0.9945	0.0490
U2R	0.7783	0.1538

	Normal	DOS	R2L	Probe	U2R	Sum
normal	64648	49	131	482	20	65330
DOS	10	45247	0	36	0	45293
R2L	76	1	2986	79	5	3147
Probe	54	5	2	11925	5	11991
U2R	25	1	4	17	165	212
Sum	64813	45303	3123	12539	195	125973

(a) Training phase

Error rate	0.0110
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
normal	0.9871	0.0050
DOS	0.9985	0.0023
R2L	0.9169	0.0861
Probe	0.9880	0.0515
U2R	0.7250	0.2162

	normal	DOS	R2L	Probe	U2R	Sum
normal	11573	16	50	80	5	11724
DOS	3	8080	1	8	0	8092
R2L	27	0	552	20	3	602
Probe	22	2	1	2061	0	2086
U2R	6	1	0	4	29	40
Sum	11631	8099	604	2173	37	22544

(b) Testing phase

Other Performance Measures

Overall Accuracy = 98.9%

False alarm rate = 1.3%

Average misclassification cost = 0.0199

4.6.1.2 CS-CRT Experiments

TABLE 4.13: CS-CRT performance using all the attributes in the training and testing phase.

Error rate	0.0577
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
normal	0.9276	0.0148
DOS	0.9735	0.0016
R2L	0.6457	0.1383
Probe	0.9957	0.3232
U2R	0.2170	0.8472

	normal	DOS	R2L	Probe	U2R	Sum
normal	60600	64	310	4115	241	65330
DOS	291	44091	0	911	0	45293
R2L	533	0	2032	568	14	3147
Probe	33	6	12	11940	0	11991
U2R	54	0	4	108	46	212
Sum	61511	44161	2358	17642	301	125973

(a) Training phase

Error rate	0.0581
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
normal	0.9291	0.0177
DOS	0.9744	0.0016
R2L	0.6146	0.1395
Probe	0.9966	0.3206
U2R	0.1750	0.8955

	normal	DOS	R2L	Probe	U2R	Sum
normal	10893	13	59	705	54	11724
DOS	52	7885	0	155	0	8092
R2L	125	0	370	101	6	602
Probe	6	0	1	2079	0	2086
U2R	13	0	0	20	7	40
Sum	11089	7898	430	3060	67	22544

(b) Testing phase

Other Performance Measures

Overall Accuracy = 94.2%

False alarm rate = 7.1%

Average misclassification cost = 0.0895

4.6.2 Experimentation II: Feature Selection from Dataset

Effective feature (attribute) selection from intrusion detection datasets is one of the important research challenges for constructing high performance IDS. Irrelevant and redundant attributes of intrusion detection dataset may lead to complex intrusion detection model as well as reduce detection accuracy. This problem has been studied during the early work [5] research on KDD99 benchmark intrusion detection dataset, where 41 attributes were constructed for each network connection.

The feature selection methods of data mining algorithms identify some of the important attributes for detecting anomalous network connections. Attributes selection in intrusion detection using data mining algorithms involves the selection of a subset of attributes "d" from a total of "D" original attributes of dataset, based on a given optimization principle by finding a useful attribute subset is a form of search.

Attribute selection and ranking [16, 17] is an important issue in intrusion detection for selecting attribute from large number of features that can be monitored for intrusion detection purpose, which are truly useful, which are less significant, and which may be useless.

This methods are commonly used for cost insensitive learning and found to be very helpful this is because of the elimination of useless features enhances the accuracy of detection while speeding up the computation, thus improving the overall performance of an IDS [17]. The attributes selection in KDD99 dataset has been widely used as a standard method for network based intrusion detection learning, and it was found that all 41 attributes of KDD99 dataset are

not the best ones for intrusion detection learning. Therefore the performance of IDS may be further improved by studying attribute selection methods [18].

The main aim of these experimentations were to explore feature (attribute) selection method called information gain which is implemented in WAKA data mining tool in enhancing of the accuracy of detection while decreasing false positive rate and average misclassification cost for cost sensitive learning algorithms CS-MC4 and CS-CRT. The feature selection here is based on supervised learning method Information Gain. Appendix B shows the feature selection result using information gain value.

These experimentations have two phases namely learning and classifying training data and then classifying the testing data. In the first phase, important attributes from full training data of NSL-KDD are selected by maximum information gain values are used to construct a detection model using the selected attributes. In the second phase, the full testing data of NSL-KDD passed through the trained model to detect the intrusions and find the detection error rates, precision, false positive, average misclassification cost and accuracy of the detection model.

We have taken 24, 30, 37 attributes at different information gain value (greater than or equal to 0.11, greater than or equal to 0.011, and greater than 0 respectively) for three different experimentations using the full training dataset of NSL-KDD as input to the system which contains about 125,973 connection records to train the system and a full test dataset which contains 22,544 connection records to test the system. The detection rate in each phase and the classification performance of CS-MC4 and CS-CRT on the NSL-KDD test dataset is presented as follows.

4.6.2.1 Experiments: Using 24 Attributes

In the first phase important 24 attributes from training data of NSL-KDD dataset are selected by maximum information gain value greater than and equal to 0.11 by assuming that the rest are less important attributes and it is used to construct detection model using CS-MC4 and CS-CRT on the selected attributes. In the second phase, the testing data of NSL-KDD passed through the trained model to detect the intrusions and measure the performances of detection model.

4.6.2.1.1 CS-MC4 Experiments

TABLE 4.14: CS-MC4 performance using 24 attributes in the training and testing phase.

Error rate	0.0088
Values prediction	**Confusion matrix**

Value	**Recall**	**1-Precision**
normal	0.9888	0.0030
DOS	0.9986	0.0014
R2L	0.9492	0.0601
Probe	0.9942	0.0511
U2R	0.6274	0.1074

	Normal	**DOS**	**R2L**	**Probe**	**U2R**	**Sum**
normal	64600	53	180	488	9	65330
DOS	9	45228	5	51	0	45293
R2L	80	2	2987	75	3	3147
Probe	56	7	2	11922	4	11991
U2R	47	0	4	28	133	212
Sum	64792	45290	3178	12564	149	125973

(a) Training phase

Error rate	0.0124
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
normal	0.9865	0.0059
DOS	0.9981	0.0023
R2L	0.8953	0.0926
Probe	0.9885	0.0597
U2R	0.5250	0.1923

	normal	DOS	R2L	Probe	U2R	Sum
normal	11566	15	48	91	4	11724
DOS	1	8077	3	11	0	8092
R2L	38	1	539	24	0	602
Probe	19	3	1	2062	1	2086
U2R	11	0	3	5	21	40
Sum	11635	8096	594	2193	26	22544

(b) Testing phase

Other Performance Measures

Overall Accuracy = 98.8%

False alarm rate = 1.4%

Average misclassification cost = 0.0232

4.6.2.1.2 CS-CRT Experiments

TABLE 4.15: CS-CRT performance using 24 attributes in the training and testing phase.

Error rate	0.0590
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
normal	0.9324	0.0227
DOS	0.9793	0.0058
R2L	0.4817	0.0375
Probe	0.9790	0.3264
U2R	0.0660	0.4400

	normal	DOS	R2L	Probe	U2R	Sum
normal	60913	254	53	4102	8	65330
DOS	26	44356	0	911	0	45293
R2L	1062	0	1516	568	1	3147
Probe	241	6	3	11739	2	11991
U2R	87	0	3	108	14	212
Sum	62329	44616	1575	17428	25	125973

(a) Training phase

Error rate	0.0586
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
normal	0.9356	0.0257
DOS	0.9801	0.0058
R2L	0.4568	0.0283
Probe	0.9794	0.3233
U2R	0.1000	0.4286

	normal	DOS	R2L	Probe	U2R	Sum
normal	10969	46	7	700	2	11724
DOS	6	7931	0	155	0	8092
R2L	225	0	275	101	1	602
Probe	42	0	1	2043	0	2086
U2R	16	0	0	20	4	40
Sum	11258	7977	283	3019	7	22544

(b) Testing phase

Other Performance Measures

Overall Accuracy = 94.1%

False alarm rate = 6.4%

Average misclassification cost = 0.0982

4.6.2.2 Experiments: using 30 attributes

In the first phase important 30 attributes from training data of NSL-KDD dataset are selected by maximum information gain value greater than and equal to 0.011 by assuming that the rest are less important attributes and it is used to construct a detection model using CS-MC4 and CS-CRT on the selected attributes. In the second phase, the testing data of NSL-KDD passed through the trained model to detect the intrusions and measure the performances of detection model.

4.6.2.2.1 CS-MC4 Experiment

TABLE 4.16: CS-MC4 performance using 30 attributes in the training and testing phase.

Error rate	0.0083

Values prediction

Value	Recall	1-Precision
Normal	0.9894	0.0026
DOS	0.9985	0.0011
R2L	0.9450	0.0344
Probe	0.9945	0.0551
U2R	0.7877	0.1436

Confusion matrix

	normal	DOS	R2L	Probe	U2R	Sum
normal	64635	45	101	530	19	65330
DOS	11	45224	1	57	0	45293
R2L	74	1	2974	94	4	3147
Probe	53	5	3	11925	5	11991
U2R	29	1	1	14	167	212
Sum	64802	45276	3080	12620	195	125973

(a) Training phase

Error rate	0.0117

Values prediction

Value	Recall	1-Precision
normal	0.9861	0.0049
DOS	0.9981	0.0014
R2L	0.9120	0.0773
Probe	0.9885	0.0640
U2R	0.7750	0.2250

Confusion matrix

	normal	DOS	R2L	Probe	U2R	Sum
normal	11561	9	44	105	5	11724
DOS	3	8077	1	11	0	8092
R2L	28	0	549	21	4	602
Probe	21	2	1	2062	0	2086
U2R	5	0	0	4	31	40
Sum	11618	8088	595	2203	40	22544

(b) Testing phase

Other Performance Measures

Overall Accuracy = 98.8%

False alarm rate = 1.4%

Average misclassification cost = 0.0201

4.6.2.2.2 CS-CRT Experiment

TABLE 4.17: CS-CRT performance using 30 attributes in the training and testing phase.

Error rate	0.0130
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
normal	0.9848	0.0056
DOS	0.9969	0.0056
R2L	0.9231	0.1735
Probe	0.9932	0.0323
U2R	0.1415	0.3750

	normal	DOS	R2L	Probe	U2R	Sum
Normal	64338	242	563	173	14	65330
DOS	32	45153	0	108	0	45293
R2L	219	0	2905	21	2	3147
Probe	61	11	8	11909	2	11991
U2R	48	0	39	95	30	212
Sum	64698	45406	3515	12306	48	125973

(a) Training phase

Error rate	0.0157
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
normal	0.9825	0.0078
DOS	0.9964	0.0059
R2L	0.8887	0.1906
Probe	0.9904	0.0382
U2R	0.2000	0.4667

	normal	DOS	R2L	Probe	U2R	Sum
normal	11519	47	119	35	4	11724
DOS	5	8063	0	24	0	8092
R2L	58	0	535	6	3	602
Probe	18	1	1	2066	0	2086
U2R	9	0	6	17	8	40
Sum	11609	8111	661	2148	15	22544

(b) Testing phase

Other Performance Measures

Overall Accuracy = 98.4%

False alarm rate = 1.7%

Average misclassification cost = 0.335

4.6.2.3 Experiments: using 37 attributes

In the first phase important 37 attributes from training data of NSL-KDD dataset are selected by maximum information gain value greater than 0 by assuming that the rest are less important attributes and it is used to construct a detection model using CS-MC4 and CS-CRT on the selected attributes. In the second phase, the testing data of NSL-KDD passed through the trained model to detect the intrusions and measure the performances of detection model.

4.6.2.3.1 CS-CM4 Experiment

TABLE 4.18: CS-MC4 Performance using 37 attributes in the training and testing phase.

Error rate	0.0077
Values prediction	**Confusion matrix**

Value	**Recall**	**1-Precision**
normal	0.9903	0.0027
DOS	0.9985	0.0011
R2L	0.9469	0.0331
Probe	0.9945	0.0498
U2R	0.8019	0.1237

	normal	**DOS**	**R2L**	**Probe**	**U2R**	**Sum**
normal	64697	42	97	481	13	65330
DOS	9	45226	0	58	0	45293
R2L	79	1	2980	81	6	3147
Probe	54	5	2	11925	5	11991
U2R	33	1	3	5	170	212
Sum	64872	45275	3082	12550	194	125973

(a) Training phase

Error rate	0.0114

Values prediction

Value	Recall	1-Precision
Normal	0.9870	0.0054
DOS	0.9981	0.0015
R2L	0.9169	0.0691
Probe	0.9880	0.0606
U2R	0.6500	0.2121

Confusion matrix

	normal	DOS	R2L	Probe	U2R	Sum
normal	11572	9	38	100	5	11724
DOS	2	8077	2	11	0	8092
R2L	30	0	552	18	2	602
Probe	22	2	1	2061	0	2086
U2R	9	1	0	4	26	40
Sum	11635	8089	593	2194	33	22544

(b) Testing phase

Other Performance Measures

Overall Accuracy = 98.9%

False alarm rate = 1.3%

Average misclassification cost = 0.0199

4.6.2.4 CS-CRT Experiment

TABLE 4.19: CS-CRT performance using 37 attributes in the training and testing phase.

Error rate	0.0578

Values prediction

Value	Recall	1-Precision
Normal	0.9275	0.0151
DOS	0.9735	0.0055
R2L	0.6435	0.0707
Probe	0.9957	0.3232
U2R	0.1981	0.8576

Confusion matrix

	normal	DOS	R2L	Probe	U2R	Sum
normal	60594	240	140	4115	241	65330
DOS	291	44091	0	911	0	45293
R2L	542	0	2025	568	12	3147
Probe	34	6	11	11940	0	11991
U2R	59	0	3	108	42	212
Sum	61520	44337	2179	17642	295	125973

(a) Training

Error rate	0.0586

Values prediction

Value	Recall	1-Precision
Normal	0.9283	0.0178
DOS	0.9744	0.0057
R2L	0.6146	0.0931
Probe	0.9966	0.3206
U2R	0.1750	0.8939

Confusion matrix

	normal	DOS	R2L	Probe	U2R	Sum
normal	10883	45	37	705	54	11724
DOS	52	7885	0	155	0	8092
R2L	126	0	370	101	5	602
Probe	6	0	1	2079	0	2086
U2R	13	0	0	20	7	40
Sum	11080	7930	408	3060	66	22544

(b) Testing phase

Other Performance Measures

Overall Accuracy = 94.1%

False alarm rate = 7.2%

Average misclassification cost = 0.0887

4.7 Discussions

Table 4.20 summarizes comparison results of detection accuracy (which refers to the proportion that a type of data is correctly classified) of normal and 4 different attacks (i.e. Probe, Dos, U2R, R2L) based on CS-MC4 and CS-CRT classifiers using 24, 30, 37, and all the attributes in comparison with KDD winner results.

	Normal		**DOS**		**R2L**		**Probe**		**U2R**	
	Cs-mc4	**Cs-crt**	**Cs-mc4**	**Cs-crt**	**Cs-mc4**	**Cs-crt**	**Cs-mc4**	**Cs-crt**	**Cs-mc4**	**Cs-crt**
All attributes	98.71	92.91	99.85	97.44	91.69	61.46	98.80	99.66	72.50	17.50
24 attributes	98.65	93.56	99.81	98.01	89.53	45.68	98.85	97.94	52.50	10.00
30 attributes	98.61	98.25	99.81	99.64	91.20	88.87	98.85	99.04	77.50	20.00
37 attributes	98.70	92.83	99.81	97.44	91.69	61.46	98.80	99.66	65.00	17.50
KDD winner	99.5		97.1		8.4		83.3		13.2	

TABLE 4.20: Performance comparison of testing for five-class classifications.

It is evident from this Table that:

1. For Normal: CS-MC4 classifiers outperform their CS-CRT counterparts in all cases but when only 30 attributes are used, the difference between the accuracy of CS-MC4 and CS-CRT became minimal i.e. 0.36%. For this type of class, in all the cases the accuracy of the KDD winner result is better than both CS-MC4 and CS-CRT classifiers.

2. For Probe attack: the accuracy of CS-MC4 is better than that of CS-CRT when only 24 attributes are used, but in other circumstances CS-CRT outperforms CS-MC4. For this type of

attack, in all the cases the accuracy of both CS-MC4 and CS-CRT is better than the KDD winner result.

3. For Dos attack: CS-MC4 classifiers outperform their CS-CRT counterparts in all cases but when only 30 attributes are used, the difference between the accuracy of CS-MC4 and CS-CRT became minimal, i.e. 0.17%. For this type of attack, in all the cases the accuracy of both CS-MC4 and CS-CRT is better than the KDD winner result.

4. For U2R attack: the accuracy of CS-MC4 classifiers is better than their CS-CRT counterparts in all cases. For this type of attack, in all the cases the accuracy of both CS-MC4 and CS-CRT is better than the KDD winner result except when only 24 attributes are used for CS-CRT classifier.

5. For R2L attack: accuracy of CS-MC4 classifiers is better than their CS-CRT counterparts in all cases. For this type of attack, in all the cases the accuracy of both CS-MC4 and CS-CRT are by far better than the KDD Cup 99 winner result.

In general, it is evident from the above table that in terms of accuracy, CS-MC4 outperformed the CS-CRT. As it can be seen from the result, the feature selection method used (i.e. information gain value) did not increase the accuracy of CS-MC4 classifier. Even though CS-MC4 classifier achieved the same result using all the attributes (41 attributes) and 37 attributes (at information gain value greater than 0), the reduction of the attributes from 41 to 37 could increase the speed of the classifier and reduce the space required. CS-CRT achieved a better result when only 30 attributes (information gain value greater than and equal to 0.011) are used relative to the other cases.

So, feature selection using information again value achieved better result for CS-CRT classifier from 41 attributes to 30 attributes but for CS-MC4 classifier from 41 attributes to 37 attributes. Results which are achieved in this work are compared with the results obtained by KDD winner results as shown in table 4.20. As it can be seen, the accuracy of KDD Winner is only better in normal, but it is far worse than CS-MC4 and CS-CRT in U2R and R2L attacks; this might be because of the data used for this research is NSL-KDD intrusion dataset (which is new version of the original KDD cup data and it is better than the original dataset in that it has no redundant data) but for the KDD winner the original KDD cup dataset is used.

Table 4.21 summarizes the efficiency of information gain value for feature selection and the performance comparison of CS-MC4, CS-CRT classifier and KDD winner result based on overall accuracy, false positives rate and average misclassification cost using 24, 30, 37 and all the attributes for 5 attack classes on NSL-KDD dataset.

	Overall accuracy (%)		**False positives rate (%)**		**Average Misclassification Cost**	
	CS-MC4	**CS-CRT**	**CS-MC4**	**CS-CRT**	**CS-MC4**	**CS-CRT**
All attributes	98.9	94.2	1.3	7.1	0.0199	0.0895
24 attributes	98.8	94.1	1.4	6.4	0.0232	0.0982
30 attributes	98.8	98.4	1.4	1.7	0.0201	0.0335
37 attributes	98.9	94.1	1.3	7.2	0.0199	0.0887
KDD winner	92.7		0.55		0.2331	

TABLE 4.21: Performance comparison of different methods.

It is evident from above table that:

1. Overall accuracy of CS-MC4 classifiers is better than their CS-CRT counterparts in all cases. And in all the cases, the overall accuracy of both CS-MC4 and CS-CRT is better than the KDD winner result.

2. False positives rate of CS-MC4 classifiers are better than their CS-CRT counterparts in all cases. And in all the cases, the false positive rate of the KDD winner result is better than both CS-MC4 and CS-CRT classifiers.

3. Average misclassification cost of CS-MC4 classifiers is better than their CS-CRT counterparts in all cases. And in all the cases, the Average misclassification cost of both CS-MC4 and CS-CRT is better than the KDD winner result.

4. Attribute reduction using information again value achieved better result for CS-CRT classifier from 41 attributes to 30 attributes, from 94.2% to 98.4% accuracy, from 7.1% to 1.7% false positive rate, and average misclassification cost from 0.0895 to 0.0335; but for CS-MC4 classifier it only reduce the attribute from 41 attributes to 37 attributes.

Chapter Five

Conclusions and Recommendations

5.1 Conclusions

This research focuses on using cost sensitive learning techniques to existing data mining algorithms to deal with cost of different types of attacks in intrusion detection while at the same time reducing the false positives rate. The cost issue is widely ignored in the intrusion detection research. As a result, most of the research projects tried to minimize the false positives rate which may not reflect real-world scenario of dealing with ever increasing different types of attacks with different costs; and an important consideration is the fact that raising false alarms carries a significantly lower cost than not detecting attacks.

For comparison results of CS-MC4, CS-CRT and KDD winner result, it was found that CS-MC4 is superior to CS-CRT and another result retrieved by a similar research [61] in terms of accuracy, false positives rate and average misclassification costs. CS-CRT is superior to KDD winner result in accuracy and average misclassification costs but in false positives rate KDD winner result is better than both CS-MC4 and CS-CRT classifiers.

This research proposed learning approach for network intrusion detection that performs feature selection method by selecting important subset of attributes. The performance of the proposed approach on the NSL-KDD intrusion detection dataset achieved a better accuracy from 94.2% to 98.4% for the CS-CRT classifier. It also reduced the false positives for the CS-CRT algorithm from 7.1% to 1.7%, and the average misclassification costs from 0.0895 to 0.0335; but for the CS-CM4 algorithm, it only reduced the attribute from 41 to 37 attributes.

Even though feature selection method could not increase accuracy or reduce false positive rate and average misclassification cost for CS-MC4, it could increase the speed of the classifier and reduce the computation space required. The experimental results have manifested that feature (attribute) selection improves the performance of IDS.

5.2 Recommendations

Future work is needed on all of the three problems addressed in this thesis. Further improvements are needed regarding the accuracy of the intrusion detection system, reduce the false positives rate and average misclassification cost while keeping a high detection rate at the same time.

The algorithms are tested on NSL-KDD intrusion detection dataset. They have to be tested on real dataset to make the necessary modifications and make it usable. Similar researches need to be conducted with a different cost sensitive learning techniques and compare the results. The research used information gain value for feature selection but cost sensitive feature selection methods can be explored to further improve the feature selection as it is helpful to achieve an enhanced result.

Bibliography

[1] Richard Power, Computer crime and security survey. Computer Security Journal, Volume XV (2), 1999.

[2] S. Wafa, Al-Sharafat , Reyadh Sh.Naoum, Adaptive Framework for Network Intrusion Detection by Using Genetic-Based Machine Learning Algorithm , IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.4, April 2009.

[3] E. Bloedorn, Data Mining for Network Intrusion Detection: How to Get Started, MITRE Technical Report, August 2001.

[4] J. Luo, Integrating Fuzzy Logic With Data Mining Methods for Intrusion Detection, Master's thesis, Department of Computer Science, Mississippi State University, 1999.

[5] S.Selvakani, R.S. Rajesh, Genetic Algorithm for framing rules for intrusion Detection, IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.11, November 2007.

[6] Sterry brugger, Data Mining Methods for Network Intrusion Detection, University of California, June 2004.

[7] D.E. Denning, An Intrusion Detection Model, IEEE Trans-actions on Software Engineering, SE-13:222-232, 1987.

[8] Paul Dokas, Levent Ertoz, Vipin Kumar, Aleksandar Lazarevic, Jaideep Srivastava, Pang-Ning Tan, Data Mining for Network Intrusion Detection ,University of Minnesota, 2001.

[10] Joseph, S and Rod, A, Intrusion detection: methods and systems. Information Management and Computer Security 11(5):222-229, 2003.

[11] A. Ajith , G. Crina, C. Yuehui. Cyber Security and the Evolution of Intrusion Detection Systems, Information Management and Computer Security 9(4): 175-182, 2001.

[12] D. Denning. An intrusion detection model. IEEE Transactions on Software Engineering 13(2): 222-32, 1967.

[13] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The Architecture of a Network Level Intrusion Detection System, Technical report, Department of Computer Science, University of New Mexico, August 1990.

[14]. L. T. Heberlein, K. N. Levitt, and B. Mukherjee. A Method To Detect Intrusive Activity in a Networked Environment, In Proceedings of the 14th National Computer Security Conference, pages 362{371, october 1991.

[15]. Simson Garfinkel and Gene Spafford. Practical Security, O'Reilly and Associates, Sebastopol, California, 1991.

[16]. R. Heady, G. Luger, A. Maccabe, and M. Servilla. The Architecture of a Network Level Intrusion Detection System, Technical report, Department of Computer Science, University of New Mexico, August 1990.

[17]. J. P. Anderson, Computer Security Threat Monitoring and Surveillance, Technicalreport, JamesP Anderson Co., Fort Washington, Pennsylvania, April 1980.

[18]. http://eric.univlyon2.fr/~ricco/tnagra.html).

[19]. Lee W, Stolfo SJ. A Framework for constructing features and models for intrusion detection systems, ACM TransInf Syst Security ,2000.

[20]. L. Ertoz, E. Eilerson and A. Lazareviv, The MINDS –Minnesota intrusion detection system, next generation data mining, MIT Press, 2004.

[21]. A. Abraham, R. Jain and J. Thomas, D-SCIDS:Distributed soft computing intrusion detection system, Journal of Network and Computer Applications, vol. 30,pp. 81-98, 2007.

[22]. Chai, X., Deng, L., Yang, Q., and Ling,C.X. , Test-Cost Sensitive Naïve Bayesian Classification, In Proceedings of the Fourth IEEE International Conference on Data Mining. Brighton, UK : IEEE Computer Society Press, 2004.

[23]. Domingos, P, MetaCost: A general method for making classifiers cost-sensitive, In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, 155-164, ACM Press, 1999.

[24]. Zadrozny, B. and Elkan, C. , Learning and Making Decisions When Costs and Probabilities are Both Unknown, In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, 204-213, 2001.

[25]. Elkan, C. The Foundations of Cost-Sensitive Learning, In Proceedings of the Seventeenth International Joint Conference of Artificial Intelligence, 973-978. Seattle, Washington, 2001.

[26]. Japkowicz, N., and Stephen, S., The Class Imbalance Problem: A Systematic Study, Intelligent Data Analysis, 6(5): 429-450, 2002.

[27]. Ling, C.X., Yang, Q., Wang, J., and Zhang, S., Decision Trees with Minimal Costs, In Proceedings of 2004 International Conference on Machine Learning (ICML'2004), 2004.

[28]. Sheng, V.S. and Ling, C.X. , Thresholding for Making Classifiers Cost-sensitive, In Proceedings of the 21 National Conference on Artificial Intelligence, 476-481, Boston, Massachusetts, 2006.

[29]. Turney, P.D., Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm, Journal of Artificial Intelligence Research 2:369- 409, 1995.

[30]. Turney, P.D. Types of cost in inductive concept learning, In Proceedings of the Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning, Stanford University, California, 2000.

[31]. Drummond, C., and Holte, R., Exploiting the cost (in)sensitivity of decision tree splitting the criteria. In Proceedings of the 17 International Conference on Machine Learning, 239- 246, 2000.

[32]. Zadrozny, B., Langford, J., and Abe, N., Cost-sensitive learning by Cost-Proportionate instance Weighting, In Proceedings of the 3th International Conference on Data Mining, 2003.

[33]. Provost, F., Machine learning from imbalanced data sets 101, In Proceedings of the AAAI'2000 Workshop on Imbalanced Data, 2000.

[34]. Ling, C.X., and Li, C., Data Mining for Direct Marketing – Specific Problems and Solutions, Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), 1998.

[35] Mohammadian, M., Intelligent Agents for Data Mining and Information Retrieval, Hershey, PA Idea Group Publishing, 2004.

[36] Wang, J., Data mining: Opportunities and challenges, Idea Group Publishing, September, 2003.

[37] Agrawal R., Mannila H., Srikant R., Toivonen H., and Verkamo A., Fast Discovery of Association Rules, Advances in Knowledge Discovery and Data Mining,MIT, 1996.

[38] Ambareen Siraj, Rayford B. Vaughn, and S. M. Bridges, Intrusion Sensor Data Fusion in an Intelligent Intrusion Detection System Architecture, Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.

[39] KDD CUP 1999 Data. The UCI KDD Archive Information and Computer Science, University of California, Irvine. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[40] Eric Bloedorn, Alan D. Christiansen, William Hill,Clement Skorupka, Lisa M. Talbot, and Jonathan Tivel, Data Mining for Network Intrusion Detection: How to Get Started, 2003.

[41] Ertoz L., MINDS - Minnesota Intrusion Detection System, Next Generation Data Mining, 2004.

[42] S. Guha, Rastogi R., and Shim K., ROCK: A Robust Clustering Algorithm for Categorical Attributes, Proceedings of the 15th Int. Conference On Data Eng, Sydney, Australia, 1999.

[45] Helmer, Wong, Honavar, and Miller, Automated discovery of concise predictive rules for intrusion detection., Technical Report TR 99-01, Department of Computer Science, Iowa State University, Ames, IA.,2001.

[46] Henery R. J., Classification, Machine Learning Neural and Statistical Classification, 1994.

[47] Knorr and R. T. Ng, Algorithms for Mining Distance Based Outliers in Large Datasets, Very Large DataBases Proceedings of the 24th Int. Conference on Very Large Databases, Aug 24-27, 1998, New York City, NY, pp. 392-403., 1998.

[48] Kohavi, Beckeer, and Sommer, Improving simple bayes, In European Conference on Machine Learning,Prague, Czech Republic., 1997.

[49] Portnoy L., E. Eskin, and Stolfo S., Intrusion detection with unlabeled data using clustering, In ACM Workshop on Data Mining Applied to Security., 2000.

[50] Ramaswarny, R. R. S., and K. Shim, Efficient Algorithms for Mining Outliers from Large Data Sets, Proceedings of the ACM Sigmod 2000 Int. Conference on Management of Data, Dallas, TX., 2000.

[51] R. Agrawal, T. Imielinski, and A. Swami, Mining Association Rules between Sets of Items in Large Databases , ACM SIGMOD Conference on Management of Data, Washington, D.C., 1993.

[52] J. Hipp, U. Güntzer, and G. Nakhaeizadeh, Algorithms for Association Rule Mining - A General Survey and Comparison, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA,USA, 2000.

[53] W. Lee, S. J. Stolfo, and K. W. Mok, A Data Mining Framework for Building Intrusion Detection Models, IEEE Symposium on Security and Privacy, Oakland, CA, 1999.

[54] W. Lee and S. J. Stolfo, Data mining approaches for intrusion detection, 7th USENIX Security Symposium (SECURITY-98), Berkeley, CA, USA, 1998.

[55] Elkan C Results of the KDD'99 Classifier Learning Contest, 1999. Available from < http://www-cse.ucsd.edu/users/elkan/clresults.html>

[56] "Nsl-kdd data set for network-based intrusion detection Systems."Available on: http://nsl.cs.unb.ca/NSL-KDD/, March, 2010.

[57] Kendall, K., A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachusetts Institute of Technology, 1999.

[58] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, A Detailed Analysis of the KDD CUP 99 Data Set, IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[59] J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory, ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, 2000.

[60] Breunig, Kriegel, R. T. Ng, and J. Sander, LOF: Identifying Density-Based Local Outliers, Proceedings of the ACM Sigmod 2000 Intl, Conference On Management of Data, Dallas, TX, 2001.

[61] V.P.Kshirsagar,and Dharmaraj R.Patil, Application of Variant of AdaBoost-Based Machine Learning Algorithm in Network Intrusion Detection, International Journal of Computer Science and Security (IJCSS), Volume (4): Issue :(2), May 03, 2010.

Appendix A

@relation 'NSL-KDD'

@attribute 'duration' real

@attribute 'protocol_type' {'tcp','udp', 'icmp'}

@attribute 'service' {'aol', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'daytime', 'discard', 'domain', 'domain_u', 'echo', 'eco_i', 'ecr_i', 'efs', 'exec', 'finger', 'ftp', 'ftp_data', 'gopher', 'harvest', 'hostnames', 'http', 'http_2784', 'http_443', 'http_8001', 'imap4', 'IRC', 'iso_tsap', 'klogin', 'kshell', 'ldap', 'link', 'login', 'mtp', 'name', 'netbios_dgm', 'netbios_ns', 'netbios_ssn', 'netstat', 'nnsp', 'nntp', 'ntp_u', 'other', 'pm_dump', 'pop_2', 'pop_3', 'printer', 'private', 'red_i', 'remote_job', 'rje', 'shell', 'smtp', 'sql_net', 'ssh', 'sunrpc', 'supdup', 'systat', 'telnet', 'tftp_u', 'tim_i', 'time', 'urh_i', 'urp_i', 'uucp', 'uucp_path', 'vmnet', 'whois', 'X11', 'Z39_50'}

@attribute 'flag' { 'OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH' }

@attribute 'src_bytes' real

@attribute 'dst_bytes' real

@attribute 'land' {'0', '1'}

@attribute 'wrong_fragment' real

@attribute 'urgent' real

@attribute 'hot' real

@attribute 'num_failed_logins' real

@attribute 'logged_in' {'0', '1'}

@attribute 'num_compromised' real

@attribute 'root_shell' real

@attribute 'su_attempted' real

@attribute 'num_root' real

@attribute 'num_file_creations' real

@attribute 'num_shells' real

@attribute 'num_access_files' real

@attribute 'num_outbound_cmds' real

@attribute 'is_host_login' {'0', '1'}

@attribute 'is_guest_login' {'0', '1'}

@attribute 'count' real

@attribute 'srv_count' real

@attribute 'serror_rate' real

@attribute 'srv_serror_rate' real

@attribute 'rerror_rate' real

@attribute 'srv_rerror_rate' real

@attribute 'same_srv_rate' real

@attribute 'diff_srv_rate' real

@attribute 'srv_diff_host_rate' real

@attribute 'dst_host_count' real

@attribute 'dst_host_srv_count' real

@attribute 'dst_host_same_srv_rate' real

@attribute 'dst_host_diff_srv_rate' real

@attribute 'dst_host_same_src_port_rate' real

@attribute 'dst_host_srv_diff_host_rate' real

@attribute 'dst_host_serror_rate' real

@attribute 'dst_host_srv_serror_rate' real

@attribute 'dst_host_rerror_rate' real

@attribute 'dst_host_srv_rerror_rate' real

@attribute 'class' {'normal', 'DOS' , 'Probe' , 'R2L' , 'U2R'}

@data

Appendix B

=== Run information ===

Evaluator: weka.attributeSelection.InfoGainAttributeEval

Search: weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1

Relation: KDDTrain

Instances: 69883

Attributes: 42

duration

protocol_type

service

flag

src_bytes

dst_bytes

land

wrong_fragment

urgent

hot

num_failed_logins

logged_in

num_compromised

root_shell

su_attempted

num_root

num_file_creations

num_shells

num_access_files

num_outbound_cmds

is_host_login

is_guest_login

count

srv_count

serror_rate

srv_serror_rate

rerror_rate

srv_rerror_rate

same_srv_rate

diff_srv_rate

srv_diff_host_rate

dst_host_count

dst_host_srv_count

dst_host_same_srv_rate

dst_host_diff_srv_rate

dst_host_same_src_port_rate

dst_host_srv_diff_host_rate

dst_host_serror_rate

dst_host_srv_serror_rate

dst_host_rerror_rate

dst_host_srv_rerror_rate

class

Evaluation mode: 10-fold cross-validation

=== Attribute selection 10 fold cross-validation (stratified), seed: 1 ===

average merit	average rank	attribute
1.038 +- 0.001	1 +- 0	5 src_bytes
0.861 +- 0.001	2 +- 0	3 service
0.732 +- 0.002	3 +- 0	30 diff_srv_rate
0.706 +- 0.001	4 +- 0	4 flag
0.661 +- 0.001	5.4 +- 0.49	6 dst_bytes
0.661 +- 0.002	5.6 +- 0.49	29 same_srv_rate
0.653 +- 0.001	7 +- 0	35 dst_host_diff_srv_rate
0.599 +- 0.001	8.2 +- 0.4	23 count
0.598 +- 0.001	8.8 +- 0.4	33 dst_host_srv_count

0.577 +- 0.001 10 +- 0 34 dst_host_same_srv_rate

0.573 +- 0.002 11 +- 0 38 dst_host_serror_rate

0.551 +- 0.002 12 +- 0 25 serror_rate

0.54 +- 0.002 13 +- 0 39 dst_host_srv_serror_rate

0.516 +- 0.002 14 +- 0 26 srv_serror_rate

0.441 +- 0.001 15 +- 0 12 logged_in

0.378 +- 0.001 16 +- 0 37 dst_host_srv_diff_host_rate

0.341 +- 0.001 17 +- 0 36 dst_host_same_src_port_rate

0.3 +- 0.001 18 +- 0 32 dst_host_count

0.238 +- 0.001 19 +- 0 24 srv_count

0.211 +- 0.001 20 +- 0 31 srv_diff_host_rate

0.14 +- 0.001 21 +- 0 40 dst_host_rerror_rate

0.122 +- 0.001 22 +- 0 2 protocol_type

0.12 +- 0.001 23 +- 0 41 dst_host_srv_rerror_rate

0.11 +- 0.001 24 +- 0 27 rerror_rate

0.079 +- 0.001 25.2 +- 0.4 1 duration

0.078 +- 0.001 25.8 +- 0.4 28 srv_rerror_rate

0.033 +- 0 27 +- 0 10 hot

0.016 +- 0 28 +- 0 22 is_guest_login

0.012 +- 0 29 +- 0 8 wrong_fragment

0.011 +- 0 30 +- 0 13 num_compromised

0.005 +- 0 31 +- 0 16 num_root

0.003 +- 0 32.4 +- 0.49 17 num_file_creations

0.003 +- 0 32.6 +- 0.49 14 root_shell

0.003 +- 0 34.1 +- 0.3 19 num_access_files

0.003 +- 0 34.9 +- 0.3 11 num_failed_logins

0.001 +- 0 36.3 +- 0.46 18 num_shells

0.001 +- 0 36.7 +- 0.46 15 su_attempted

0 +- 0 38.1 +- 0.3 7 land

0 +- 0 39.1 +- 0.3 21 is_host_login

0 +- 0 39.8 +- 0.6 9 urgent

0 +- 0 41 +- 0 20 num_outbound_cmds

Appendix C

Features of NSL- KDD dataset.

Attribute	Category	Informations
Duration	Continue	-
protocol_type	Discrete	3 values
Service	Discrete	70 values
Flag	Discrete	11 values
src_bytes	Continue	-
dst_bytes	Continue	-
Land	Discrete	2 values
wrong_fragment	Continue	-
Urgent	Continue	-
Hot	Continue	-
num_failed_logins	Continue	-
logged_in	Discrete	2 values
num_compromised	Continue	-
root_shell	Continue	-
su_attempted	Continue	-
num_root	Continue	-
num_file_creations	Continue	-

num_shells	Continue	-
num_access_files	Continue	-
num_outbound_cmds	Continue	-
is_host_login	Discrete	2 values
is_guest_login	Discrete	2 values
Count	Continue	-
srv_count	Continue	-
serror_rate	Continue	-
srv_serror_rate	Continue	-
rerror_rate	Continue	-
srv_rerror_rate	Continue	-
same_srv_rate	Continue	-
diff_srv_rate	Continue	-
srv_diff_host_rate	Continue	-
dst_host_count	Continue	-
dst_host_srv_count	Continue	-
dst_host_same_srv_rate	Continue	-
dst_host_diff_srv_rate	Continue	-
dst_host_same_src_port_rate	Continue	-

dst_host_srv_diff_host_rate	Continue	-
dst_host_serror_rate	Continue	-
dst_host_srv_serror_rate	Continue	-
dst_host_rerror_rate	Continue	-
dst_host_srv_rerror_rate	Continue	-
Class	Discrete	5 values

Appendix D

Table shows attacks, their category and availability of attacks in the NSL-KDD dataset.

Attack	Category	Training dataset	Test dataset
Apache2	Dos		X
Back	Dos	X	X
Buffer-overflow	U2r	X	X
Ftp_Write.	R21	X	X
Guess-passwd.	R21	X	X
Httptunnel.	R21		X
Imap.	R21	X	X
Ipsweep.	Probe	X	X
Land.	Dos	X	X
Loadmodule	U2r	X	X
Mailbomb.	Dos		X
Mscan.	Probe		X
Multihop.	R21	X	X
Named	R21		X
Neptune.	Dos	X	X
Nmap.	Probe	X	X
Normal.	Normal	X	X
Perl.	U2r	X	X
phf.	R21	X	X

Pod.	Dos	X	X
Portsweep.	Probe	X	X
Processtable.	Dos		X
Ps.	U2r		X
Rootkit	U2r	X	X
Saint.	Probe		X
Satan.	Probe	X	X
Sendmail.	R21		X
Smurf.	Dos	X	X
Snmpgetattack.	R21		X
Snmpguess.	R21		X
Spy	R21	X	X
Sqlattack	. U2r		X
Teardrop.	Dos	X	X
Udpstom.	Dos		X
Warezclient.	R21	X	X
Warezmaster.	R21	X	X
Worm.	R21		X
Xlock.	R21		X
Xsnoop.	R21		X
Xterm.	U2r		X

X- Represents the availability of attack in the indicated category