ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

OPTIMAL FEATURE SELECTION FOR NETWORK INTRUSION DETECTION: A DATA MINING APPROACH

ZEWDIE MOSSIE

JUNE, 2011

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION SCIENCE

OPTIMAL FEATURE SELECTION FOR NETWORK INTRUSION DETECTION: A DATA MINING APPROACH

A Thesis Submitted to the 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

ZEWDIE MOSSIE

JUNE, 2011

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

OPTIMAL FEATURE SELECTION FOR NETWORK INTRUSION DETECTION: A DATA MINING APPROACH

By

ZEWDIE MOSSIE

Name and signature of Members of the Examining Board

Name	**Title**	**Signature**	**Date**
_____	Chairperson	_____	_____
_____	Advisor,	_____	_____
_____	Examiner,	_____	_____

Declaration

I declare that the thesis is my original work and has not been presented for a degree in any other university.

_____

Date

This thesis has been submitted for examination with my approval as university advisor.

_____

Advisor

ACKNOWLEDGEMENT

First and foremost I would like to acknowledge the tireless and prompt help of my advisor, Ato Workshet Lamenew. Ato Workshet has always allowed me complete freedom to define and explore my own directions in research. While this proved difficult and somewhat confusing to begin with, I have come to appreciate the wisdom of his way; it encouraged me to think for myself, something that is unfortunately all too easy to avoid. Thank you for your trust in me. Thank you for your patience to teach me how to do research. Thank you for your scientific and personal guidance. I learned a lot from your superior advices. Thank you for your generosity, enthusiasm and patience for being a great teacher. Thank you for your careful reading of my thesis. And thank you for your endless support in several key moments. I will always be proud to have been your student.

I am grateful to Ato Adamu Teshome for providing valuable feedback and comments on technical aspect and reading parts of this thesis.

I am also grateful to Ato Zeleke Ababaw and Ato Getinet Yilma for their contribution in reading the first draft of the thesis.

Special thanks must also go to my family, my mother Asres Mitiku and my partner Nigist Yenework, for their understanding and love. You are the motivating force behind me at all times through both the highs and lows of my time in graduate school. Thank you for everything you give me.

ABSTRACT

The traditional approach in securing computer systems against cyber threats is designing mechanisms such as firewalls, authentication tools, and virtual private networks that create a protective shield almost always with vulnerabilities. This has created Intrusion Detection Systems (IDS) to be developed that complement traditional approaches. However, with the advancement of computer technology, the behavior of intrusions has become complex that makes the work of network security experts hard to analyze and detect intrusions. In order to address these challenges, using data mining techniques have become a possible solution. However, the performance data mining algorithms are affected when no optimized features provided. This is because, complex relationships can be seen as well between the features and intrusion classes contributing to high computational costs in processing tasks, subsequently leads to delays in identifying intrusions. Feature selection is thus important to be conducted in detecting intrusions by allowing the data mining system to focus on what is really important.

Researches on data mining have focused on the induction of models with low expected error by totally ignoring the cost that could be incurred during misclassification and feature selection in skewed data distribution between classes. In reality, 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 the cost of allowing unauthorized access can be much greater than that of wrongly denying access to authorized individuals. Similarly the cost of not selecting features that contain unauthorized profiles is much more than probing profiles. Implementing cost sensitive classifiers that involve cost by modifying (direct) and without modifying (indirect) algorithm during model building and feature selection are a rising research interest to handle this problem and attempts have been made.

However, little attention has been given to evaluate the performance of direct and indirect cost sensitive classifiers using cost sensitive feature selection approach. In this research, we proposed filter approach to select important features; namely, IGR and CFS to

illustrate the significance of feature selection in classifying the NSL-KDD intrusion detection dataset. The central idea is the minority class feature sets, those which have low values, can be ranked at the top by gaining high information gain value and correlation percentages, at the same time those score low, ranked at the bottom in WEKA tool assuming some of the features can be redundant or contribute little to the detection process.

The selected features are experimented repeatedly where features added into the final selected feature set as far as no decrease in performance and then models are constructed on the two algorithms; namely, CS-CM4 (direct) and C4.5 (indirect) using TANAGRA tool. Experiments show that CFS and IGR select below half of the total (41) features with equaled or better performance in most cases. Comparatively, the approach fits more for indirect cost sensitive C4.5 than direct cost sensitive CS-CM4. Generally, the study indicated that CS-CM4 and C4.5 algorithms by far achieved better on the proposed approach with fewer features that require less storage and time to identify new attacks as well as better performance in terms of detection rate, overall classification accuracy, average misclassification cost and false positive rate.

Keywords*:* Cost sensitive feature selection, Cost insensitive feature selection, IGR, CFS.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AMC: Average Misclassification Cost

ARFF: Attribute Relation File Format

BP: Back- propagation

CART: Classification and Regression tree

CASH :Cost sensitive Attribute Selection algorithm using Histograms

CFS: Correlation Feature Selection

CS: Cost Sensitive

CSV: Comma Separated Value

DOS: Denial of Service

DR: Detection Rate

FN: False Negative

FP: False Positive

FPR: False Positive Rate

GUI: Graphical User Interface

HIDS: Host Intrusion Detection Systems

ICET: Inexpensive Classification with Expensive Testing

IDS: Intrusion Detection Systems

IG: Information Gain

IGR: Information Gain Ratio

IP: Internet Protocol

KDD: Knowledge Discovery in Databases

KDP: Knowledge Discovery Process

LAN: Local Area Network

MINDS: Minnesota Intrusion Detection System

MIT: Massachusetts Institute of Technology

NIDS: Network Intrusion Detection System

NSL: Network Simulation Language

OCA: Overall classification Accuracy

ProCASH :Progress Cost sensitive Attribute Selection algorithm using Histograms

RBF: Radial Basis Function

RFF: ReliefF

R2L: Remote to User attacks

SOM: Self-Organizing feature Map

SU: Symmetrical Uncertainty

SVM: Support Vector Machine

TCP: Transport Control Protocol

TN: True Negative

TTBM: Time Taken to Build Model

TP: True Positive

U2R: User to Root attacks

CHAPTER ONE

1. INTRODUCTION

We are living in the information age and nearly impossible to imagine our lives without the Internet and information systems. As a result, present networks based on the concept of resource sharing for collaboration, and an easy means of communication and economic growth considered as a vital tool [1]. However, the need to communicate and share resources increases the management complexity of securing computer systems, that can be depended upon to behave as it is expected. This results in vulnerabilities in software and configuration errors in networks and deployed applications. There is also the risk of malicious intrusions, such as computer viruses or the theft of data. In addition to vulnerabilities and poor management of resources, ease of access of resources can be exploited to launch attacks by intruders [2].

Following this Network Intrusion Detection (NID) is receiving considerable attention as a mechanism for shielding cyber infrastructure against attempts to compromise the confidentiality, integrity, or to bypass the security mechanisms of computer network. But, the proliferation of Internet and networking applications, coupled with the widespread availability of system hacks and viruses have increased the need for network security [1]. Network security technology has become crucial in protecting government and industry computing infrastructure. But, modern intrusion detection applications face complex requirements; they need to be reliable, extensible, easy to manage, and have low maintenance cost [3].

The most widely deployed methods for detecting cyber attacks and protecting against those attacks employ signature based detection techniques. Such methods can only detect previously known attacks that have a corresponding signature, since the signature database has to be manually revised for each new type of attack that is discovered. These limitations have led to an increasing interest in intrusion detection techniques based on

data mining [4]. Data mining based 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 ‘intrusive’ 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. Research in misuse detection has focused mainly on classification of network intrusions using various standard data mining algorithms [1]. On the other hand, anomaly detection relies on models of the ‘‘normal’’ behavior of a computer system and automatically detects any deviation from it as suspect. Anomaly detection techniques thus identify new types of intrusions as deviations from normal usage. The anomaly detection systems have the advantage of being able to detect previously unknown attacks. This advantage is paid for in terms of the large number of false positives and the difficulty of training a system with respect to a very dynamic environment [5].

In recent years, machine learning based Intrusion Detection Systems (IDS) have demonstrated high accuracy, good generalization to novel types of intrusion, and robust behavior in a changing environment. The goal is to identify occurrences of security breaches capable of compromising the integrity of resources or services [6]. It also corresponds to a suite of techniques that are used to identify attacks against computers and network infrastructures. For an intrusion detection system, it is important to detect previously known attacks with high accuracy. However, detecting previously unseen attacks is equally important in order to minimize the losses as a result of a successful intrusion [7].

Due to large volumes of data as well as the complex and dynamic properties of intrusion behaviors, identifying intrusion traffic from normal becomes difficult. Because this data becomes more useful when it is analyzed and some dependencies and correlations are detected. As a result, data mining techniques are gaining prevalence in the production of a wide range of classifiers for complex real world applications with nonuniform testing

and misclassification costs. The increasing complexity of these applications poses a real challenge to resource management during learning and classification [8].

Traditionally, data mining algorithms have focused on the induction of models with low expected error; however, in many real world applications several additional constraints should be considered [9]. The one this work emphasizes is network data must be transformed into a format that has manageable number of features that enable identify intrusions effectively.

Identifying intrusions solely based on human eyes is therefore extremely difficult. To alleviate the problem, network security experts utilize existing data mining and artificial intelligence techniques in search of possible intrusions. However, if the number of features involved in network data increases, identifying intrusions can become difficult because of the complex relationships between features [10]. Complex relationships can be seen as well between the features and intrusion classes. This contributes to high computational costs in processing tasks subsequently leads to delays in identifying intrusions. Seeing the limitations of both humans and computers, feature selection is thus important to be conducted so that the load in processing data and time consumed in detecting intrusions can be reduced [11].

A feature selection approach is the one that reduce the complexity of the overall process by allowing the data mining system to focus on what is really important. Thus, the data mining knowledge produced is found more meaningful. In general, two different approaches for feature selection can be distinguished [12]: filter and wrapper approaches. Using a filter approach, the selection of appropriate features is based on distance and information measures in the feature space and is carried out completely independent from the classifier deployed. In contrast, with a wrapper approach the selection of features is based on the classifiers accuracy. All feature selection methods have different resource requirments. The filter method has low computational cost with low reliability in classification while wrapper methods tend to have superior classification accuracy but require great computational effort. The hybrid method is a recent technique which exploit

the advantages of both filter and wrapper methods. It employs both an independent test and performance evaluation function of the feature subset [13].

Feature selection is an optimization method [8] which attempts to select the more discriminative features from data sets in order to improve classification quality and reduce computational complexity that can work on labeled and unlabeled data. The goal of feature selection is to avoid selecting too many or too few features than is necessary [10]. If too few features are selected, there is a good chance that the information content in this set of features is low. On the other hand, if too many (irrelevant) features are selected, the effects due to noise present in (most real world) data may overshadow the information present. Hence, this is a trade-off that must be addressed by any feature selection method.

1.1. Statement of the Problem

Intrusion Detection Systems (IDS) are recognized as one of the areas requiring extensive research to tackle existing problems of network security [14]. The reliability issue in detecting intrusions is one of the challenges faced by researchers. False alarms are likely to be generated by IDS if the intrusive activities in networks are unfamiliar to its existing knowledge [15]. Damage costs caused by intruders, are also the main concern in IDS which is overwhelmed by imbalanced data.

Data mining is the primary technique in extracting relevant knowledge from volumes of data. Similarly imbalanced class distribution and non-uniform misclassification cost problem has been improved using cost sensitive learning techniques. The cost sensitive technique considers cost factor, i.e. cost matrix when building classifiers with the aim of minimizing the total misclassification costs [16]. In a dataset with the class imbalance problem, the most obvious characteristic is the skewed data distribution between classes; because misclassification costs are often closely related with imbalanced distribution of class values in the dataset (rare classes usually being of higher interest) in this case the User to Root (U2R) and Remote to Local (R2L).

Assuming the minority 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 traditional (cost insensitive) classifiers are applied on the dataset, they will likely to predict everything as negative (the majority class). In the same case selection of features biases to majority class feature having large values of records compared to the minority class.

Feature selection has been studied extensively; however, the majority of data mining literature evaluation does not take feature selection costs into consideration although, tasks that require attribute selections which consider these costs are abundant in real-world applications [17]. An example of such a task is applying Host-based Intrusion Detection Systems (HIDS), where a continuous monitoring of many parametrs is conducted in order to detect malicious applications. This is becasue there are devices that have very limited in their resources like mobiles,the continuous monitoring of parameters must be limited in order to efficiently use HIDS on such devices. The performance of common attribute evaluation measures in problems where the class distribution is imbalanced and in problems with unequal misclassification costs becomes hot research issue [18].

In this study the focus is on the works of [19], that investigate classification should not only 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. For many problem domains, the requirement is not merely to predict the most probable class categories, but there has to be examining the major cost factors associated with an IDS damage cost due to successful intrusions since different types of errors carry different costs.

Having this problem [19] investigated two cost sensitive classifier algorithms CS-CRT and CS-MC4 which uses misclassification cost matrix to minimize the expected misclassification cost and for the detection of best prediction implemented in TANAGARA package. Experimental results have manifested that feature (attribute) selection improves the performance of IDS for the different attack types for direct cost sensitive algorithms and reduction in space of features.

However, a classification task cannot achieve high performance without being provided a good set of features, regardless of the algorithm that is being used to train the computer [20]. So feature selection is vital to result in good classifier performance. In the work of [19] Information Gain (IG) value is used to select features which works based on information theory by selecting features that reveal the most information about the classes.

Ideally, such feature selections are highly discriminative and occur in a single class. However, in text classification it is often the case that features would occur in more than one class. This consequently forces IG to select the best features amongst those that occur in multiple classes. In this situation, IG would be sufficient to select such multi-class features, provided that all pairs of classes are equally similar. However, if some pairs of classes are more similar than others (for example, when there is an ordinal relationship between them) then it matters which classes a feature is distributed across. In this case, a feature that occurs in two similar classes is relatively more discriminative than one that occurs in two dissimilar ones. As a result of this, a feature selection algorithm that does not address inter-class similarities (in particular IG) would be inadequate for selecting such multi-class features [13].

The gap in [19] is the cost insensitive feature selection Information Gain (IG) value that reveals the most information about the classes where the focus of the study has been cost sensitive learning. The cost of feature (attribute) selection is not considered that tends to be, the higher the misclassification cost, the higher the weight value. The weights can be used to incorporate the cost of not selecting a particular feature for machine learning. There are varieties of cost sensitive feature selection algorithms deployed in limited data mining tools that are not investigated. In this thesis, metacost sensitive feature selections are proposed to evaluate the performance of direct and indirect cost sensitive learning algorithms.

The other gap need empirical investigation on the work of [19] is using KDD 99 winner result [15] as a benchmark for direct comparison. The point here is that the KDD 99 winner is based on cost sensitive bagged boosting (initially every instance has an equal weight) of decision trees and very successful in binary classification, even though it can also be applied to multiple classes [16]. Comparing the performance of representatives of state-of-the-art direct cost sensitive classifier with that makes use of indirect cost information algorithm(weight is assigned to each example to reflect its importance) to check appropriateness of the proposed approach and illustrate more on the algorithms. The indirect cost information algorithm C4.5 is chosen because it is widely and commonly used and the cost insensitive variant of CS-CM4 making comparisons valid.

Fundamentally, this study aims to extend the works of [19] (information gain value feature selection) to metacost sensitive feature selection approaches and compare empirically the result with cost insensitive feature selection using C4.5 and CS-CM4 classifiers in comparison with other recent works to see the performance gap. Finally, the research tries to address the following research questions:

- Which cost sensitive information selection algorithm CS-CM4 (direct) or C4.5 (indirect) fit more for the proposed approach (cost sensitive feature selection)?
- Is it possible to increase the performance of CS-MC4 and C4.5 algorithm using metacost sensitive feature selection approach compared to cost insensitive feature selection approach?
- Is it possible to decrease the feature space usage and time taken to train in detecting new attacks by applying the metacost sensitive feature selection without degrading the performance compared to the cost insensitive feature selection?

1.2. Objectives

1.2.1. General Objectives

The general objective of this study is to investigate jointly cost sensitive learning and feature selection to advance the classification performance of algorithms that incorporate cost using data mining approach for network intrusion detection systems.

1.2.2. Specific Objectives

The specific objectives of this research are the following:

- To review different literatures on the concept of intrusion detection in the area of data mining particularly feature selection approaches.
- To preprocess NSL-KDD dataset by removing missing and incomplete values.
- To experiment appropriateness of cost sensitive feature selection for direct and indirect cost sensitive classifier algorithms.
- To construct models using CS-CM4 and C4.5 machine learning algorithms on optimal features selected using cost sensitive and insensitive approaches.
- To compare the cost sensitive and insensitive feature selection approaches.
- To compare the performance of the models built with other recent works.
- To report results and make recommendation based on findings

1.3. Scope and Limitations of the Study

Due to time constraints this research has looked and concentrated on the IDS methods and techniques mainly feature (attribute) selection. The emphasis is on selected cost sensitive classification decision tree and feature selection algorithms. This work does not cover other non-decision tree cost sensitive learning and non-filter feature selection algorithm. But, there are algorithms that need investigation on the approach which have different methods in incorporating costs and in more complex learning algorithms. However dataset may not be a perfect representative of existing real networks. This is because; data collection methods are often loosely controlled, resulting in many data quality problems such as feature redundancy and irrelevance especially in dynamic and

complex environment. This leads also not to compare analogously with other works on the same techniques and methods.

1.4. Research Methodology

The methodology that followed in conducting this research is described as follows:

1.4.1. Literature Review

A recent technical report on current IDS, Journals articles, books, conference papers Internet and non commercial software products are briefly reviewed that provide a discussion on the challenges to developing effective IDS. These includes trends on IDS and its applications, data mining and intrusion detection, cost sensitive learning and intrusion detection and cost sensitive optimal feature selection approaches.

Some experimental studies comparing different misclassification cost minimization methods have been previously reported, but most of them focus on accuracy maximization. We here describe some of these efforts. Cost sensitive machine learning and feature selection methods that assign different costs to different class in order to punish the misclassification, where minimization of total misclassification costs is the classification criterion, instead of maximization of the information gain or something else [18]. Feature selection on IDS implemented without considering cost though cost of feature selection getting prevalence to minimize processing time and CPU usage [17]. The KDD winner cost matrix [16] used for this research is used to estimate the impact of damage on domains where there is class imbalance. In situations where there is class imbalance and devices with very limited in their storage, efficient monitoring of many parameters becomes challenging [17]. Exploring algorithms that include cost to minimize damage is getting importance [18], [19]. Algorithms that incorporate cost either directly [36] or indirectly [15], [44], [61] during learning is getting importance. In many works the dataset used is simulated that make comparisons valid [21], [24] with their limitation in representing existing real network traffic. To this end data mining tools are available that enable identify normal data from intrusive by including cost in their learning algorithms [23], [51] and feature selection [23].

The data mining model followed as a methodology is Knowledge Discovery in Databases (KDD) which is, interactive and iterative, involving numerous steps with many decisions made by the user [101]. The steps in short can be expressed: understand the application domain and select the dataset under target (Data Collection), cleaning and preprocessing (Data Preparation), data reduction and transformation (Feature Selection), matching particular data mining methods to search the data using algorithms (Training and Building Model), and outputs patterns and relationships to the evaluation (testing models) [90]. The description of each component is presented below.

1.4.2. Data Collection

NSL-KDD data set [21] most widely used data set publicly available for IDS is used for the experimental purpose, which consists of the same features as KDD 99. There are many reasons to use this dataset. First, it is problematic to get high quality data for performing the evaluation due to privacy and competitive issues, since many organizations are not willing to share their data with other institutions to use real data. Second, even if real life data are available, labeling network connections as normal or intrusive requires enormous amount of time for many human experts. Third, the constant change of the network traffic can not only introduce new types of intrusions but can also change the aspects of the "normal" behavior, thus making construction of useful benchmarks even more difficult.

This dataset make easy to be shared with other researchers, allowing all kinds of techniques to be easily compared in the same baseline. Even though the dataset might have been criticized for its possible problems [22] the fact is that it is the most prevalent dataset among the few comprehensive datasets that can be shared in intrusion detection nowadays. Finally it has a reasonable number of training and test instances which make it practical to run the experiments.

1.4.3. Data Preparation

Data preprocessing, a critical initial step in data mining work, is often used to improve the quality of training data set. To do so data cleaning and preparation is the core task of

data mining which is dependent on software chosen and algorithms used [22]. In this study we attempted to prepare the data according to the requirements of the TANAGRA [51] and WEKA [23] software package which are powerful, user friendly and freely available for noncommercial purpose. The preprocessing is according to the requirements of CS-MC4 and C4.5 algorithms by consulting different literatures implemented in TANAGRA. The consistency of individual attribute values and types, and quantity and distribution of missing values is identified using Microsoft Excel. Then the data format is changed into a form that can be understood by the TANAGRA and WEKA. The metacost sensitive and insensitive attribute selection approaches are employed with WEKA tool for feature ranking and selection. The metacost sensitive feature selection implemented only in WEKA; that is why we select it for this work.

1.4.4. Feature Selection

Data reduction and projection are ways of finding useful features to represent the data depending on the goal of the task [90]. With dimensionality reduction or transformation methods, the effective number of variables under consideration can be reduced, or invariant representations for the data can be found [20].

1.4.5. Training and Building Models

The intrusion detection models are developed using direct cost sensitive CS-MC4 and C4.5 as indirect cost sensitive decision tree algorithms on full training NSL-KDD dataset using TANAGRA tool and the performance of each is compared with each other and with the selected benchmarks and recent works.

1.4.6. Testing Models

The intrusion detection model is tested with appropriate proportion of test data set which is already distinguished from the beginning on the NSL-KDD dataset to evaluate the performance of the feature selection approaches and algorithms used to build the models in terms of overall classification accuracy, detection rate, false alarm rate, time taken to build and test the model, and average misclassification cost. As interest in intrusion

detection has grown, the topic of evaluation of IDS has also received great attention [24], [25], [26], [22].

Intrusion detection evaluating systems is a difficult task due to several reasons. The main one is when measuring the performance of an IDS, there is a need to measure not only detection rate (i.e. how many attacks detected correctly), but also the false alarm rate (i.e. how many of normal connections were incorrectly detected as attacks) as well as the cost of misclassification. The evaluation is further complicated by the fact that some of the attacks (e.g. denial of service (DoS), probing) may use hundreds of network packets or connections, while on the other hand attacks like U2R (User to Root) and R2L (Remote to Local) typically use only one or a few connections [24].

In summary the data mining models followed in this research can be expressed in figure (1.1) as shown below adopted from [90].

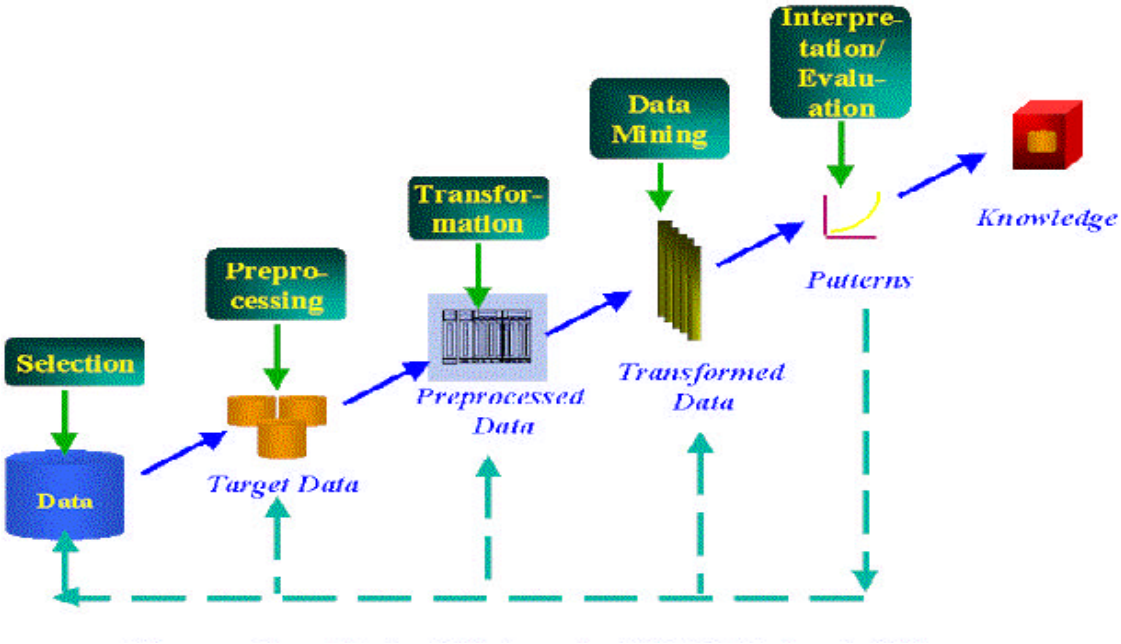


Figure 1.1: an Overview of the steps that compose the KDD Process

1.5. Significance of the Study

Network security is a growing 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. 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.

A fundamental problem in data mining is to understand the conditions for which a learning algorithm works well. Understanding an algorithm's strengths and weaknesses and being able to compare two or more algorithms with each other are necessary for designers to develop (or select) learning algorithms for a specific problem. However, most researchers and practitioners route to empirical evaluation to understand the interaction between learning algorithms and a domain. Unfortunately, most evaluation methods give very little information to the designer. For example, the most common method of empirically evaluating a classifier is to examine its error, or more generally loss, and many comparisons of algorithms use only this metric [27]. However because loss is a single number, it reveals little about the algorithm except gross performance on the domain but here the focus is on direct and indirect cost sensitive machine learning and metacost sensitive feature selection approaches with many evaluation metrics that give more information.

In addition, other researchers will use and try to learn when a classification algorithm is appropriate for a problem domain based on characteristics of the data set properties such as the number of features and number of classes to learn by inspection or through automated analysis when an algorithm is appropriate. Finally, direct and indirect cost sensitive learning approach to intrusion detection can detect intrusion with acceptable rate of false positives, high detection accuracy, low misclassification cost and time taken to build models. This may greatly cuts down costs, allowing network managers or system administrators engage on other productive activities.

The research is applicable where, a continuous monitoring of many parameters is conducted to detect malicious applications. Becasue there are devices that have very limited in their resources like mobiles,the continuous monitoring of parameters must be limited in order to efficiently use such devices. There is also domins that prevails unequal cost in slecting and not selecting a particular feature.

1.6. Thesis Outline

The rest of the paper is organized as follows. Chapter 2 defines some terms and provides an overview of concepts on computer system security from continuation of chapter one. The focus is application of data mining for in intrusion detection, data sets available for IDS and algorithms that have cost sensitive during learning. The algorithms are from the supervised family for the study even though discussions are there from unsupervised. It also reviews some common machine learning algorithms and techniques. Feature selection methods with evaluation metrics to measure the relevance of features are surveyed in relation to machine learning. Three broad categories of algorithms that involve machine learning scheme to estimate the worth of features is discussed. Techniques of evaluation measure of features, feature selection and ranking algorithms and cost sensitive feature selection are described.

Chapter 3 begins by presenting the brief discussion of the NSL-KDD benchmarked dataset, its preprocessing strategy and tools used. The chapter also describes the work proposed in literatures that use for performance evaluation metrics. Chapter 4 presents the brief discussion of experimental setup. The first half of the chapter empirically experiments the two approaches of cost sensitive and insensitive (each employing the information gain ratio and feature correlation) measures are examined. The second half of the chapter evaluates machine learning algorithms with experimental results are presented and analyzed in detail in each one of the approaches. Finally in chapter 5 the paper is concluded with their future investigation based on finding results.

CHAPTER TWO

2. LITERATURE REVIEW

In this chapter, firstly as an introduction the seriousness of the network security problem are pointed out. Afterwards, some conventional network intrusion detection methods are briefly discussed before data mining based approaches introduced. In the next section, the topic of data mining and cost sensitive learning are discussed. In the final section, the focus of the study, feature selection models, cost sensitive approaches and optimal feature selection, which link closely to the detection of network intrusions are discussed.

2.1. Computer System Security

As [28] define a secure computer system is one that can be depended upon to act as likely. The reliability that is displayed between the likely behavior and the exhibited behavior is referred to as trust in the security of the computer system. They define the level of trust as an indication of the assurance in the expected behavior of the computer system. The likely behavior is included into the security policy of the computer system and governs the goals that the system must meet.

Computer security also narrowly defined as the recognition of confidentiality, integrity, and availability in a computer system. Confidentiality to mean information is available only those authorized to access it; integrity assures that information remains unchanged by accident or malicious tampering; and availability ensures that the computer system remains working when needed without deprivation of access to approved users [29].

2.1.1. Threats to Security

A threat is the probable for a particular vulnerability to be successfully exercised. Vulnerability is a weakness that can be accidentally triggered of intentionally exploited. A threat does not present a risk when there is no vulnerability that can be exercised [30]. In the early years of computers, the number of users for every machine was usually

limited to just one. The computers were not connected to each other, and rather than implementing security features on the machine, the door to the computer room was simply locked. With the growth of interconnected machines, and later large networks, the awareness of the risks associated with networks increased [31]. The threats that network systems are confronting come from failures of hardware or software, tentative probing and malicious attacks from local or remote hackers. Classification scheme of intrusions based on intrusion type, [32] presents the following six types:

1. **Attempted break-in:** regularly detected by profiles of a typical behavior or violations of security constraints.
2. **Masquerade attack:** often detected by profiles of a typical behavior or violations of security constraints.
3. **Penetration of the security control system**: usually detected by monitoring for specific patterns of activity.
4. **Leakage**: often detected by a typical usage of Input/output (I/O) resources.
5. **Denial of service**: frequently detected by a typical usage of system resources.
6. **Malicious use**: often detected by a typical behavior profiles, violations of security constraints, or use of special privileges.

2.1.2. Detecting threats to Security

New methods and countermeasures are continuously being developed to detect treat attacks. Some of these methods are able to actually stop attacks, while some only detect breaches in security; after or during their occurrences. Some examples of this are [30]:

- **Physical controls:** Sometimes the easiest ways to enforce security are ignored as more sophisticated technical approaches are sought. Some straightforward ways are locks on doors, guard at entry points, and backup copies of important data.
- **Software controls or countermeasures:** Security is often implemented in the software in terms of internal program controls, operating system (OS) controls, development controls, and antivirus software. Recording network logs to monitor

what has been going on is one form of defense against system failures and human attacks.

- **Encryption:** To ensure confidentiality, sensitive data is often encrypted, making the data incomprehensible to an outside observer. Additionally, encryption enforces integrity to a certain degree, since data that cannot be read is generally hard to modify in a meaningful manner.

2.2. Intrusion Detection

The main worry of intrusion detection system is to identify possible abnormal behavior from computer users. Intrusion detection is the task of detecting and responding to sophisticated techniques exploiting security vulnerabilities of computer misuse, by detecting unauthorized access to a computer network [33]. Intrusion detection started in 1980's and since then a number of approaches have been introduced to build intrusion detection systems [34], [35], [36], [37]. However, intrusion detection is still at its childhood and inexperienced attackers can launch powerful attacks which can bring down an entire network [38].

An IDS does not usually take defensive measures when an attack is detected, it is a reactive rather than pro-active agent [39], [3]. It plays the role of information rather than a police officer. It is thus more important than ever before that since it seems apparent that administrators cannot prevent insurgence, they should at least try to detect it and prevent similar attacks in the future. The following keywords are used in IDS [7]:

- **Risk**: unintentional or unpredictable exposure of information, or violation of operations integrity due to the malfunction of hardware or incomplete or incorrect software design.
- **Vulnerability:** A known or suspected flaw in the hardware or software or operation of a system that exposes the system to penetration or its information to accidental disclosure.
- **Attack:** A specific formulation or execution of a plan to carry out a threat.

- **Penetration:** A successful attack, the ability to obtain unauthorized (undetected) access to files and programs or the control state of computer system.
- **Intrusion:** A set of actions aimed to compromise the security goals, namely integrity, confidentiality, or availability, of a computing and networking resource.
- **Intrusion detection:** The process of identifying and responding to intrusion activities.

2.2.1. Intrusion Detection Categorization

Generally, there are two kinds of classification methods for intrusion detection system [40]: Source of data and detection methods.

Based on the sources of data, intrusion detection systems can be divided into two major classes, host based and network based intrusion detection systems.

2.2.1.1. Host Based vs. Network Based Intrusion Detection

Host based Intrusion Detection (HID): In HID systems, the intrusion detection system is installed on the local host/terminal. By investigating the status of audit information on system's behavior, the system finds signs of intrusion and can then defend its own local machine. It identifies the intruders by monitoring host based traffic audit information [41]. The audit information can be obtained from different sources such as system logs and activities, application logs, and target monitoring.

Network Based Intrusion Detection (NID): NID compares network traffic data being observed with well defined attack patterns. Each pattern is described by a vector of network features that perform a condition part for classifiers in data mining which has been used to automatically discover these patterns that will be used to detect network intrusion according to input data from network packet [41].

Based on different detection methods, intrusion detection systems can be divided into two major classes, misuse and anomaly.

2.2.1.2. Misuse vs. Anomaly Based Intrusion Detection

Misuse Detection (Knowledge-Based): It is also named signature based detection, which can change the information of attack indication 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, pretreated case data should be first compared with the signature of signature database, and those compliant to attack signature data can be judged as attack. 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 [40].

Anomaly Detection (Behavior Based): It may establish a profiles for normal behavior of users, which comes from statistics data of users in the previous 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. 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 rate is also high [40].

Hybrid: The advantage of misuse detection is low misjudgment rate, as well as low detection capacity for unknown attacks; comparatively, anomaly detection owns the capacity of detecting unknown attacks, but with high misjudgment rate. If these said two methods are combined for detection, they can supply disadvantage of each other like Minnesota Intrusion Detection System MINDS [42].

2.3. Goals and Functions of IDS

The purpose of intrusion detection systems is to accurately detect anomalous network behaviour or misuse of property, sort out true attacks from false alarms, and advise network administrators of the activity. Many organisations now use intrusion detection

systems to help them decide if their systems have been compromised. Given the goal of IDS, the functions of IDS can be [31]:

- Monitoring users and system activity
- Auditing system configuration for vulnerabilities and miss configurations
- Assessing the integrity of critical system and data files
- Recognizing known attack patterns in system activity
- Identifying abnormal activity through statistical analysis
- Managing audit trails and highlighting user violation of policy or normal activity
- Correcting system configuration errors
- Installing and operating traps to record information about intruders

2.4. Data Mining and Intrusion Detection

Data mining is supporting various applications for required data investigation. Recently, data mining is becoming an important element in intrusion detection system. Different data mining approaches like clustering, classification, association rule, and outlier detection are commonly used to evaluate network data to develop intrusion related knowledge [43], [44].

2.4.1.1. Clustering

Clustering is the classification of similar objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait often proximity according to some defined distance measure. Machine learning typically regards data clustering as a form of unsupervised learning [45] techniques that can sense intrusions that have not been previously learned. Examples of such algorithms include K-means-Clustering and Self-Organizing feature Map (SOM) [57].

It is useful in intrusion detection as malicious activity should cluster together, separating itself from non-malicious activity. Clustering provides some significant advantages over the classification techniques; it does not require the use of a labeled data set for training.

The amount of available network audit data instances is large, human labeling is time-consuming, and expensive. It can also be 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 [45], [46]. This can be achieved by first, clustering the traffic data in to different cluster groups and then give to the classifier.

2.4.1.2. Classification

Classification is similar to clustering in that it also partitions records into distinct segments called classes. But unlike clustering, classification analysis requires that the end-user know in advance 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, and because the end-user decides on the attribute to use, classification is much less investigative than clustering [7].

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 allocate examples to pre-defined categories. Machine learning software performs this task by extracting or learning discrimination rules from examples of correctly classified data. 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 modification techniques to reduce its demonstrated high false positives rate [47]. Supervised learning methods for intrusion detection can only perceive known intrusions. In supervised learning, the input of the learning algorithm consists of examples (in the form of feature vectors) with a label assigned to them. The objective is to learn to assign correct labels to new unseen examples of the same task. In this thesis, the focus is on supervised learning algorithms, since the labels of intrusion dataset are done by supervision.

Generally in supervised learning classification models can be two types: rule-based and instance-based classifiers. A rule-based classifier learning algorithm generalizes rules for classifying the test instances, and an instance-based classifier learning algorithm stores the training instances and predicts the class of the stored instances which are nearest (according to some distance measure) to the test instance. There is also eager versus lazy learning algorithms. Eager learning algorithms devote most of their effort in the learning phase; lazy learners defer the decision of how to take a broad view beyond the training data until each new query instance is encountered. Hybrids are mixtures of the eager and lazy learners. The reason for constructing hybrids is the contrast between memory-based learning and eager learning. This leads to the hope that this double effort will be repaid with improved performance [57]. In this section, a subset of well developed classifier learning algorithms is reviewed and discussed.

Decision Trees: Decision trees use simple knowledge representation to classify examples into a finite number of classes. In a typical setting, the tree nodes represent the attributes, the edges represent the possible values for a particular attribute, and the leaves are assigned with class labels. Classifying a test sample is straightforward once a decision tree has been constructed. An instance is classified by following paths from the root node through the tree, taking the edges corresponding to the values of attributes, until the splitting completed or no overfitt. Some of the popular tree algorithms include ID3 [48], C4.5 [49] and CART [50].

A decision tree classifier is modelled in two phases: Tree Building and Tree Pruning. In tree building, the decision tree model is built by recursively splitting the training data set based on a locally optimal standard until all or most of the records belonging to each of the partitions bear the same class label. After building the decision tree, a tree pruning step is performed to reduce the size of the decision tree. Decision trees that are too large are prone to overfitting. Pruning attempts to get better the generalization competence of a decision tree by orderly the branches of the initial tree. The tree pruning approach is fault based: start from the base of the tree and inspect each non-leaf subtree. If replacement of

this subtree with a leaf, or with its most frequently used branch, would lead to a lower predicted fault rate, then prune the tree accordingly [48].

Neural Networks: Neural networks have the topology of a directed graph and loosely simulate the structure of biological neural networks in human brains. They are composed of processing nodes that convey actions to each other using connections. This one way inter unit connections grasp the processing ability of the network through weights obtained by learning from a set of training data. Each node evaluates the input values, calculates a total for the joint input values, compares the total with a threshold value, and determines what its own output will be. A neural network's learning is defined as changes in the memory weight matrix. There is a variety of strategies to train the network, including applications of numerical and statistical methods such as Back- Propagation (BP) errors, Radial Basis Function (RBF) of differential equations, least squares fitting and others [54].

Bayesian Classification: Bayesian classification is based on the inferences of probabilistic graphic models which indicate the probabilistic dependencies essential to a particular model using a graph structure [53]. In its simplest form, a probabilistic graphical model is a graph in which nodes stand for random variables, and the arcs represent conditional dependence assumptions. Hence it provides a compressed representation of combined probability distributions.

Support Vector Machines: Support vector machines are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. SVMs attempt to separate data into multiple classes (two in the basic case) through the use of a hyper-plane. It is one of the binary classifiers based on maximum margin strategy introduced by [54].

2.4.1.3. Outlier Detection

An outlier is unusual observation that to the highest degree deviates from the characteristic distribution of other observations. Outlier detection has many applications, such as data cleaning, fraud detection and network intrusion. The truth of outliers

indicates that individuals or groups that have very different manners from most of the individuals of the dataset. Many times, outliers are removed to improve accuracy of the estimators [44].

2.4.1.4. Association Rule

The Association rule is particularly designed for use in data analyses. It considers each attribute/value pair as an item. An item set is an arrangement of items in a single network request. The algorithm scans through the dataset trying to come across item sets that are inclined to appear in many network data. The objective behind using association rule based data mining is to derive multi-feature (attribute) correlations from a database table. Association rule mining finds associations and/or correlation relationships among large set of data items [44], [55].

2.4.2. Intrusion Dataset

The KDD cup 99 intrusion detection dataset [56] is based on the 1998 DARPA project, which contributes designers of Intrusion Detection Systems (IDS) with a standard on which to evaluate different methodologies. It was used in the 3rd International Knowledge Discovery and Data Mining Tools Competition for building a network intrusion detector, a systematic model capable of characterizing between intrusions and normal connections.

In 1998, DARPA intrusion detection evaluation program, a simulated environment was set up to obtain raw TCP/IP dump data for a Local Area Network (LAN) by the Massachusetts Institute of Technology (MIT) Lincoln Lab to evaluate the performance of various intrusion detection methods. It was operated like a real environment, but being darned with several intrusion attacks and received much attention in the research community of adaptive intrusion detection [24]. The KDD 99 dataset competition uses a version of DARPA 98 dataset. In KDD 99 dataset, each example corresponds to attribute values of a class in the network data flow, and each class is labeled either normal or attack.

The work of [22] shows that there are some likely problems in the KDD 99 data set. The first deficiency in the KDD 99 data set is the huge number of redundant records. Analyzing KDD 99 train and test sets, they found that about 78% and 75% of the records are duplicated in the train and test set, respectively. This large amount of redundant records in the train set make learning algorithms incline towards the more frequent records, and thus bring to a halt it from learning unfrequent records which are usually more damaging to networks such as R2L and U2R attacks. The reality of these repeated records in the test set, on the other hand, will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records.

To solve these issues [22] have proposed a new data set, NSL-KDD [21], which consists of selected records of the complete KDD 99 data set. This data set is publicly available for researchers and has the following advantages over the original KDD data set:

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
- There are no duplicate records in the proposed test sets; therefore, the performance of the learners is not biased by the methods which have better detection rates on the frequent records.
- The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of dissimilar data mining methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
- The numbers of records in the train and test sets are reasonable, which makes it reasonable to run the experiments on the complete set without the need to randomly select a small portion. As a result, assessment results of different research works will be reliable and analogous.

2.4.2.1. Attack Category

The training and test dataset of NSL-KDD similar to KDD99 consists both quantitative and qualitative 41 features and is labeled as either normal or an attack, with exactly one specific attack type and the simulated attacks fall in one of the following five main classes for intrusion analysis [22]: (one normal class and four main intrusion classes: Probe, DOS, U2R, and R2L).

a. **Normal connections (Normal)** are produced by pretend daily user behavior such as downloading files, and visiting web pages.
b. **Denial of Service (DOS)** attack causes the computing power or memory of a victim machine too busy or too full to handle legal requests. DOS attacks are classified based on the services that an attacker renders unavailable to genuine users like apache2, land, mail bomb, back, etc.
c. **Remote to User (R2L)** is an attack that a remote user gains access of a local user/account by sending packets to a machine over a network communication, which include send-mail, and Xlock.
d. **User to Root (U2R)** is attacks that an intruder begins with the access of a normal user account and then becomes a root-user by make use of various vulnerabilities of the system. Most common exploits of U2R attacks are regular Buffer-overflows, load-module, Fd-format, and Ffb-config.
e. **Probing (Probe)** is an attack that scans a network to gather information or find known vulnerabilities. An intruder with a map of machines and services that are available on a network can use the information to look for exploits.

2.4.2.2. Intrusion Features Category

According to [24] the 41 features that are found in NSL-KDD and KDD 99 dataset can be classified into three groups:

1. **Basic features**: this group encapsulates all the attributes that can be extracted from a TCP/IP connection. Most of these features leading to an inherent impediment in detection.
2. **Traffic features**: this category includes features that are computed with respect to a window interval and is divided into two groups:
    a. **Same host features**: examine only the connections in the past 2 seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc.
    b. **Same service features**: examine only the connections in the past 2 seconds that have the same service as the current connection. The two abovementioned types of traffic features are called time-based. However, there are several slow probing attacks that scan the hosts (or ports) using a much larger time interval than 2 seconds, for example, one in every minute. As a result, these attacks do not produce intrusion patterns with a time window of 2 seconds. To solve this problem, the “same host” and “same service” features are re-calculated but based on the connection window of 100 connections rather than a time window of 2 seconds. These features are called connection-based traffic features.
3. **Content features**: unlike most of the DoS and Probing attacks, the R2L and U2R attacks don’t have any intrusion frequent sequential patterns. This is because the DoS and Probing attacks involve many connections to some host(s) in a very short period of time; however the R2L and U2R attacks are embedded in the data portions of the packets, and normally involves only a single connection. To detect these kinds of attacks, there has to be some features that able to look for suspicious behavior in the data portion, e.g., number of failed login attempts. These features are called content features. The details are shown in (appendix A).

2.5. Cost Sensitive Learning and Intrusion Detection

2.5.1. Machine Learning

Machine learning is the study of computer algorithms that improve automatically through experience. Applications range from data mining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests. In contrast to statistical techniques, machine learning techniques are well suited to learning patterns with no a priori knowledge of what those patterns may be. Clustering and classification are probably the two most popular machine learning problems. Techniques that address both of these problems have been applied to IDSs.

Machine learning [44] is a quickly increasing new technology for mining knowledge from data. In data mining, the data is stored electronically and the search is automated by computer. A major focus of machine learning research is to automatically learn to be familiar with complex patterns and make intelligent decisions based on data. Its difficulty lies in the fact that the set of all possible behaviors are difficult to describe [31]. Machine learning algorithms can be divided in two major categories: supervised and unsupervised [47], which is discussed in classification and clustering respectively.

2.5.2. Cost Sensitive Learning

Costs are central to statistical decision theory though cost sensitive learning received only modest attention before [15]. Cost sensitive learning is a type of learning in data mining that takes the misclassification costs (and possibly other types of cost) 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 practice a high accuracy of classifying examples into a set of known classes [16].

Cost sensitive learning is an extensively used practice in data mining, which assigns different levels of misclassification penalty to each class. Cost sensitive technique has been incorporated into classification algorithms by taking into account the cost information and trying to optimize the overall cost during the learning process [58]. Generally cost sensitive learning techniques fall into two types [59], i.e. the direct and indirect methods.

2.5.2.1. Direct Cost Sensitive Method

The main idea of building a direct cost sensitive learning algorithm is to directly introduce and make use of misclassification costs into the learning algorithms such as ICET, EG2 and CS-ID3 [36]. The direct method incorporate misclassification costs and test costs in the fitness function of genetic algorithm, but the definition of fitness function is very different from each other. The cost sensitive learning algorithms that are used for this research are direct cost sensitive learning algorithms i.e. cost sensitive decision tree CS-MC4 used by [19]; the cost sensitive version of C4.5 [49] that use misclassification cost matrix to minimize the expected cost and for the detection of best prediction.

Most classifiers assume that the misclassification costs (false negative and false positive cost) are the same. In most real-world applications, this assumption is not true. For example, in customer relationship management, the cost of mailing to non-buyers is less than the cost of not mailing to the buyers [16]; or the cost of misclassifying a non-terrorist as terrorist is much lower than the cost of misclassifying an actual terrorist who carries a bomb to a flight. Another example is cancer diagnosis: misclassifying a cancer is much more serious than the false alarm since the patients could lose their life because of a late diagnosis and treatment [60].Cost is not necessarily monetary, for examples, it can be a waste of time or even the severity of an illness [15].

2.5.2.2. Indirect Cost Sensitive Method

The indirect method is to design a “wrapper” that changes any existing cost insensitive classifiers into cost sensitive ones, which can be further divided into three categories: Relabeling, Reweighting, and Re-sampling [59]. This method is also called cost sensitive

meta-learning, such as MetaCost [15], Costing [61], and CostSensitiveClassifier [44]. In this approach the cost insensitive C4.5 decision tree is used by making it as MetaCost [15].

Generally both direct and indirect method 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. [58] 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.

2.5.3. Costs in Intrusion Detection Induction

The works of [74] proposed to use cost sensitive machine learning techniques that can automatically build discovery models optimized for the overall cost metrics instead of relying solely on statistical accuracy.

2.5.3.1. Damage costs

The damage cost differentiates the highest amount of harm inflicted by an attack when intrusion detection is unavailable or not wholly effective. This cost is important but difficult to define since it is probable a function of the risks that need to be analyzed by a site that seeks to protect itself. The distinct cost function per attack (or attack type) and per service (e.g., http, ftp, and email) should be used to compute the cost of damage. Rather than simply measuring false negatives (FN) as a rate of missed intrusions, measure the total damage loss of all missed attacks, because different intrusions cost differently, and the goal of IDS is to minimize damage to the business. [16] FN cost is the cost of not detecting an attack, and represents the most dangerous case. It is incurred by most systems today that do not field IDSs. Here, the IDS falsely decide that a connection is not an attack and does not respond to the attack. This indicates that the attack will

succeed and some service or data might be lost. The FN cost is therefore defined as the damage cost associated with the particular type of attack [62].

2.5.3.2. Response Costs

The response cost is the cost to drive alarm that indicates a potential intrusion. For IDS, one might consider reducing or suspending an uncertain connection and attempting to check, by analyzing the service request, if any system data was compromised, or if any system resources were badly treated by attackers or blocked from other legal users. Other costs can be included, such as the time spent by personnel gathering evidence for trial purposes if the intruder can be traced. These costs can be predictable, as a first approximation, by the amount of CPU and disk resources needed to respond to a doubtful connection. For simplicity, instead of estimating the response cost for each intrusive connection, they assign a typical, averaged, response cost for each specific type of attack [16]. TP cost is the cost of detecting an attack and doing something about it (i.e., challenging it). Here, one hopes to stop an attack from damaging the service in the case of a correctly classified attack [62].

2.5.3.3. Operational Costs

The operational cost is the cost of running IDS. The main cost here is the amount of resources needed to extract and test features from raw traffic data. Some features cost more to gather than others. However, costlier features are often more enlightening for detecting intrusions. For example, to notice accurately some denial of service attacks, IDS needs to compute and examine some temporal and statistical features of active connections. Computing such features requires the storage and lookup of potentially many active connections, and is more costly than just examining features of a single connection. IDS should detect an ongoing attack and generate an alarm as quickly as possible so that damage could be minimized. Slower IDS which uses features with higher computational costs should therefore be penalized [62]

2.5.4. Cost Models in Cost Sensitive Machine Learning

The cost model of IDS devises the total expected cost of the IDS. The cost model depends on the detection performance of the IDS. Misclassification costs false positive (FP, the cost of misclassifying negative instance into positive) and false negative (FN, the cost of misclassifying a positive instance into negative), and the cost of correct classification is zero. They simply assign FP as the weight to each negative instance, and assign FN as the weight to each positive instance [16].

The cost of misclassifying an instance of class i as class j is C(i, j) and is assumed to be equal to 1 unless specified otherwise; C(i, j) = 0 for all i . Any classification tree can have a total cost computed for its terminal node assignments by summing costs over all misclassifications. The issue in cost sensitive learning is to induce a tree that takes the costs into account during its growing and pruning phases [15]. These misclassification cost values can be given by domain experts, or learned via other approaches. In cost sensitive learning, it is usually assumed 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 as shown in table 2.1.

	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 2.1: An example of cost matrix for binary classification

When optimizing sampling levels to improve overall cost, a logical evaluation criterion is the cost itself. Therefore, once classification occurs in the cross-validation phase, the wrapper or filter calculates the validation cost and uses this information to select optimal feature selection levels. This approach is dependent on a priori knowledge of the cost relationship between classes.

2.6. Cost Sensitive Optimal Feature Selection Methods for IDS

The majority of machine learning literature evaluation does not take feature grouping, misclassification and test costs in the attribute selection process into consideration. However,tasks that require attribute selection which consider these costs are abundant in real-world applications. To cope with the cost problem, [17] developed a new cost sensitive fitness function based on histogram comparison which integrates with a genetic search method to form a new attribute selection algorithm termed CASH (Cost sensitive Attribute Selection algorithm using Histograms). Moreover,several data mining tasks are faced with resource constraints.

An example of such a task is applying Host-based Intrusion Detection Systems (HIDS),where a continuous monitoring of many parameters is conducted in order to detect malicious applications. Becasue there are devices that have very limited in their resources like mobiles,the continuous monitoring of parameters must be limited in order to efficiently use HIDS on such devices. Moreover,in order to cope with the resource constraint problem they also developed another attribute selection algorithm, ProCASH (Progress Cost sensitive Attribute Selection algorithm using Histograms) which incorporates a new search method and fitness function. CASH and ProCASH are new cost sensitive feature selection algorithms [17] sensitive to resource consumption, misclassification costs and feature grouping. As far as our knowledge concerns the two algorithms are not implemented in the existing data mining tools. In this work the existing cost insensitive feature selection algorithms are used using the indirect cost sensitive learning methods which are discussed in the next section.

The performance of common attribute evaluation measures in problems where the class distribution is imbalanced and in problems with unequal misclassification costs was investigated. The empirical results show that there is difference in different number of classes. ReliefF, which use probability estimates in the update formula reliably exploit information from cost matrix and effective in its ability to detect highly dependent attributes however its processing time is large. Cost sensitive adaptation of gain ratio fails

to detect all important attributes in two class problem and also recommended feature selection and weighting in the cost sensitive context [18].

Feature reduction (feature selection, variable selection) methods have been widely used in many fields. Feature lessening methods have the applications in the fields of text classification, protein classification and intrusion detection [63]. According to the availability of class labels, there are feature selection methods for supervised learning [64]. The feature lessening methods can be broadly divided into three classes called filters, wrappers and embedded methods [65], [66].

2.6.1.1. Filter Feature Selection Methods

The supervised filter based feature selection algorithms can be divided into a number of categories from the different points of views. For this work the following algorithms which are available in WEKA [23] are discussed. The filter based feature reduction methods identify the most hopeful feature subset from original set of features. The methods work autonomous of learning algorithm and are more computationally efficient than other methods [67]. In literature, the important filter methods proposed are Relief and its extension [63], Information Gain (IG) [13], Information Gain Ratio (IGR) [48], Symmetrical Uncertainty (SU) [48], [68] and Correlation Based feature selection (CFS) [68]. Filter feature reduction methods differ in the assessment process of relevance of features. The assessment process may consist of subset selection or feature ranking. Subset selection process selects the relevant features and rejects the irrelevant feature. Feature ranking process ranks the feature in definite order of degree of relevance [65], [66]. Feature ranking process determines the significance of individual features and it neglects probable relations of features. The reduction methods are among the simplest and fastest approaches based upon measurement of relevance of the features [69].

Filter methods can be additionally divided into ranker and non-rankers. Rankers are methods that employ some measure to score each feature and make available a ranking. From this ordering, several feature subsets can be chosen, by manually setting a threshold. A non-ranker method provides only a selected subset of the features without providing any ranking. The process can be expressed figuratively as shown in figure 2.1.

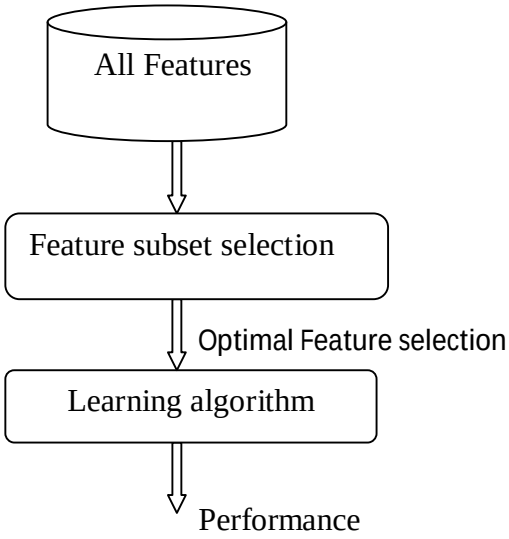


Figure 2.1: The feature filters approach independently of the learning algorithm

2.6.1.2. Wraper feature selection Methods

The wrapper methods select the features by using the forecast performance of learning algorithm. The method is reliant upon the learning algorithm. The wrapper methods are less generative and computationally inefficient than other methods. However, a wrapper-based approach is the best largely attribute selection scheme in terms of accuracy if speed of execution was not considered [70].

The search space organization is such that each state represents a feature subset. For **n** features, there are **n** bits in each state, and each bit indicates whether a feature is present (1) or absent (0). Operators decide the connectivity between the states, and choose to use operators that add or delete a single feature from a state, analogous to the search space commonly used in stepwise methods in statistics. The size of the search space for **n** features is $2^{\mathbf{n}}$, so it is impractical to search the whole space exhaustively, unless **n** is small [71].

Wrapper methods search through the space of feature subsets and calculate the estimated accuracy of a single learning algorithm for each feature that can be added to or removed from the feature subset. The feature space can be searched with various strategies, e. g.,

forwards (i. e., by adding attributes to an initially empty set of attributes) or backwards (i. e., by starting with the full set and deleting attributes one at a time). Usually an exhaustive search is too expensive, and thus non-exhaustive, heuristic search techniques like genetic algorithms, greedy stepwise, best first or random search are often used [71]. The process can be expressed figuratively as shown below in figure 2.2.

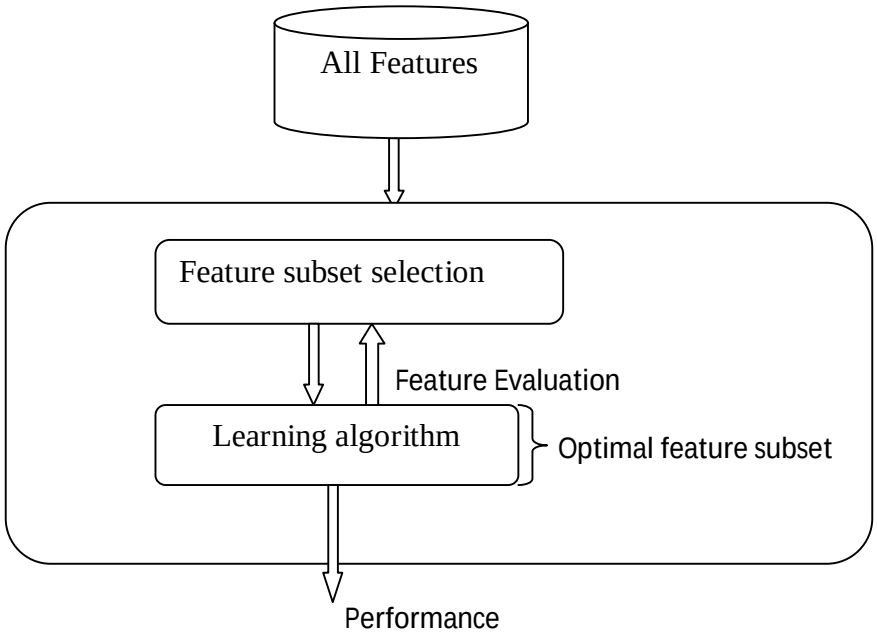


Figure 2.2: The feature wrapper approach wrapped with learning algorithm

2.6.1.3. Embedded Feature Selection Methods

The embedded methods involve the integration of feature reduction techniques with the learning process for a given learning algorithm. In other words the method embeds the feature selection procedure into the learning process. The example for embedded method is C4.5 [48]. To take advantage of the above two models and avoid the pre-specification of a stopping criterion, the hybrid model is recently proposed to handle large data sets makes use of both an independent measure and a mining algorithm to evaluate feature subsets.

The hybrid method uses the independent measure to decide the best subsets for a given cardinality and uses the mining algorithm to select the final best subset among the best subsets across different cardinalities. Basically, it starts the search from a given subset S_0 (usually an empty set in sequential forward selection) and iterates to find the best subsets at each increasing cardinality. In each round for a best subset with cardinality c, it searches through all possible subsets of cardinality c + 1 by adding one feature from the remaining features. Each newly generated subset S with cardinality c + 1 is evaluated by an independent measure M and compared with the previous best one. If S is better, it becomes the current best subset S_{best} at level c + 1. At the end of each iteration, a mining algorithm A is applied on S_{best} at level c+1 and the quality of the mined result is compared with that from the best subset at level c. If S_{best} is better, the algorithm continues to find the best subset at the next level; otherwise, it stops and outputs the current best subset as the final best subset. The quality of results from a mining algorithm provides a natural stopping criterion in the hybrid model [64].

Generally the aim of feature reduction methods set is many folds [69]. Firstly, it avoids the over fitting problem and helps to improve the performance of model. Secondly, it helps to develop fast and cost effective models. Finally, it helps better understanding of the processes that generate the data. These benefits are achieved at the cost of additional complexity in the modeling process. Given the input data D tabled as N number of instances and M number of features X {x, i1... M} i and target class variable T. The feature reduction problem is defined as to find a subset of M features that optimally characterize class variable T.

The optimal description of class variable depends upon two factors. First factor is the searching algorithms that search the best subset of features meeting the optimal characterization condition. In spite of exhaustive searching, many other techniques like forward search, backward elimination, sequential forward floating search, etc. have been proposed in literature [72], [73]. Second factor is the situation that defines the optimal characterization. Generally, the condition is minimal classification error rate or maximal of dependency of class variable on the subset of features.

2.6.2. Evaluation Measure of Features

In literature, many metrics have been recommended to determine the importance of features. [62] Have divided the evaluation measures into five classes: 1) distance, 2) information (or uncertainty), 3) dependence, 4) consistency, and 5) classifier error rate.

1. **Distance Measure:** It is also known as separability, divergence, or discrimination measure. For a two class problem, a feature fi is chosen to another feature fj. If fi induces a greater difference between the two-class conditional probabilities than fj; if the difference is zero then fi and fj are impossible to differentiate. An example of this is Euclidian distance. Distance measure is employed in [75], [76].

2. **Information Measure:** These measures normally establish the information gain from a feature. The information gain from a feature fi is defined as the difference between the prior uncertainty and expected posterior uncertainty using fi. Feature fi is chosen to feature fj if the information gain from feature fi is greater than that from feature fj. An example of this type is entropy. Information measure is employed in [77], [78].

3. **Dependence Measure:** Dependence measures or correlation measures enumerate the ability to predict the value of one variable from the value of another variable. Correlation coefficient is a traditional dependence measure and can be used to find the correlation between a feature and a class variable. If the correlation of feature fi with class variable C is higher than the correlation of feature fj with C, then feature fi is preferred to fj. An insignificant variation of this is to determine the dependence of a feature on other features; this value indicates the degree of redundancy of the feature. All evaluation functions based on dependence measures can be classified as distance and information measures. But, these are still kept as a separate category because, conceptually, they represent a different viewpoint. An example of this is CFS. Dependence measure is employed in [79].

4. **Consistency Measure:** This type of evaluation measures is characteristically different from other measures because of their heavy reliance on the training

dataset and use of Min-Features bias in selecting a subset of features. Min-Features bias prefers consistent hypotheses definable over as few features as possible. These measures find out the minimal size subset that satisfies the acceptable inconsistency rate that is usually set by the user. Consistency measure is employed in [80]. The above types of evaluation measures are known as "filter" methods because of their independence from any particular classifier that may use the selected features output by the feature selection method.

5. **Classifier Error Rate Measure:** In contrast to the above filter methods, classifier error rate measures are called "wrapper methods", i.e. a classifier is used for evaluating feature subsets [71]. As the features are selected using the classifier that later uses these selected features in predicting the class labels of unseen instances, the accuracy level is very high although computational cost is rather high compared to other measures. Wrappers often give better results (in terms of the final predictive accuracy of a learning algorithm) than filters, because feature selection is optimized for the particular learning algorithm used. However, since a learning algorithm is employed to evaluate each and every set of features considered, wrappers are prohibitively expensive to run, and can be intractable for large databases containing many features. Furthermore, since the feature selection process is tightly coupled with a learning algorithm, wrappers are less general than filters and must be re-run when switching from one learning algorithm to another.

These metrics are used to measure the correlation between two variables. The correlation can be measured by using correlation coefficient, least square regression error and maximal information compression index. Information theory based metrics are used most of the time to measure the non linear correlation between the features. The most important measure is the concept of entropy that measures the uncertainty between variables. In this work information measure (IG) and dependency measure (CFS) are used.

2.6.3. Characteristics of Feature Selection Algorithms

Ideally, feature selection methods should choose the optimal feature subset from a candidate set to describe the target conceptions of a learning system. The following aspects must be considered in the process of feature selection [81].

2.6.3.1. Starting Point

From the set of complete features, first one must determine the starting point in feature space; this in turn influences the direction of search. The search for feature subsets can start with no features or, it can start with all (full) features. The first approach is called forward selection, and second is known as backward elimination [81]. One can also start the search from not only the two points explained above. A set with only half the number of the complete feature set or a set with a random number of features good, as well.

2.6.3.2. Search Strategy

Feature selection methods as used in various studies require a search algorithm to generate candidate subsets from the features space. The following common search techniques are used: Greedy search [82], Best First search [83], Genetic search [84], [85] and Ranker search [74]

Greedy search: Greedy search considers changes local to the current subset through the addition or removal of features. For a given 'parent' set, greedy search examines all possible 'child' subsets through either the addition or removal of features. The child subset that shows the highest goodness measure then replaces the parent subset, and the process is repeated. The process terminates when no more improvement can be made [82].

Best First search: Best First search is similar to greedy search in that it creates new subsets based on addition or removal of features to the current subset. However, it has the ability to backtrack along the subset selection path to explore different possibilities when the current path no longer shows improvement. To prevent backtracking through all

possibilities in the feature space, a limit is placed on the number of non-improving subsets that are considered [83].

Genetic search: A Genetic search attempts to find an optimal solution using evolutionary concepts. An initial population of individuals (solutions) is generated at random or heuristically [85]. Genetic algorithms are adaptive search techniques based on the principles of natural selection in biology. They employ a population of competing solutions evolved over time to converge to an optimal solution [84].

Ranker search: Ranker search on attribute selection techniques are used to produce ranked lists of attributes. These methods are not only useful for improving the performance of learning algorithms, the rankings they produce can also provide the data miner with insight into their data by clearly demonstrating the relative merit of individual attributes [74]. Rankers are methods that employ criterion to score each feature and provide a ranking. From this ordering, several feature subsets can be chosen manually by setting a threshold. Where as a non-ranker methods provide only a selected subset of the features without providing any ranking.

2.6.3.3. Stopping Criteria

Finally, one must make a decision the criteria for halting (stopping) the search [81]. For example, one can stop adding or removing features when none of the alternatives improves the approximation of classification accuracy, or we can stop when the number of selected features reaches a pre-determined threshold [66]. It is possible then to choose the best subset among the candidates one has encountered during the search. A stopping criterion determines when the feature selection process should stop. Some frequently used stopping criteria are the search completes, some given bound is reached, where a bound can be a specified number (minimum number of features or maximum number of iterations) and a sufficiently good subset is selected (e.g., a subset may be sufficiently good if its classification error rate is less than the allowable error rate for a given task) [66].

2.6.4. Feature Ranking and Selection Algorithms

Feature selection algorithm differ in the search strategy, the feature selection criteria, the way they add or delete an individual feature or a subset of features at each round of feature selection and the overall model for feature selection. The inclusion of the particular classifier into the feature selection process makes the wrapper approach more computationally expensive and the resulting feature subset will only be appropriate for the used classifier while the filter approach is classifier independent. However, both models require a search and evaluating strategy that should come close to optimal. Various search strategies have been developed in order to reduce the computation time as discussed in the above topics. Here the feature selection algorithms are elaborated as follows.

In this study, the focus is on filter based feature ranking and correlation percentage techniques (i.e., ranking and correlating features independently without involving any learning algorithm). The method of feature ranking is to score each feature according to a particular method, allowing the selection of the best set of features. Diverse feature ranking and feature selection techniques have been proposed in the machine learning literature, Such as [44]: Information Gain (IG) [44], [13], Information Gain Ratio (IGR) [68], Symmetrical Uncertainty (SU) [68], Relief and its Extension [63] and Correlation-Based Feature Selection (CFS) [76].

Information Gain: Information Gain (IG) is based on information theory using the concept of entropy, which measures the impurity of a data items. The value of entropy is small when the class distribution is uneven, that is when all the data items belong to one class. The entropy value is higher when the class distribution is more even, that is when the data items have more classes. Information gain is a measure on the utility of each attribute in classifying the data items. It is measured using the entropy value. Information gain [99] measures the decrease of the weighted average impurity (entropy) of the attributes compared with the impurity of the complete set of data items. Therefore, the

attributes with the largest information gain are considered as the most useful for classifying the data items.

Given a training data $D = \{t_1,...t_n\}$ where $t_i = \{t_{i1},...t_{ih}\}$ and the training data D contains the following attributes $\{A_1, A_2,...A_n\}$ and each attribute A_i contains the following attribute values $\{A_{i1}, A_{i2...,} A_{in}\}$. The attribute values can be discrete or continuous. Also the training data D contains a set of classes $C = \{C_1, C_2...C_m\}$ each example in the training data D has a particular class C_j. The algorithm calculates the information gain for each attributes $\{A_1, A_2...A_n\}$ from the training data D.

$$\inf(D) = \sum_{j=1}^{m} \frac{freq(C_j, D)}{|D|} \log_2 \left(\frac{freq(C_j, D}{|D|} \right) \qquad (1)$$

$$\inf(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \inf(T_i) \qquad (2)$$

From (1) and (2) $IG(A_i) = Inf(D) - \inf(T)$

Then the algorithm chooses one of the best attributes Ai among the attributes $\{A_1, A_2...A_n\}$ from the training data D with highest information gain value, and split the training data D into sub-datasets $D_i = \{D_1, D_2...D_n\}$ depending on the chosen attribute values of Ai. The algorithm then estimates the prior and conditional probabilities for each sub-dataset $D_i = \{D_1, D_2...D_n\}$ and classifies examples of sub-dataset Di using their respective probabilities. The prior probability $P(C_j)$ for each class is predictable by counting how often each class occurs in the dataset. For each attribute Ai the number of occurrences of each attribute value A_{ij} can be counted to determine $P(A_j)$.

Information Gain Ratio: The notion of information gain introduced earlier tends to favor attributes that have a large number of values. The splitting criterion is very important in the process of building the tree, because it determines whether a node or a

leaf as next element in the tree. The idea is to partition the training set in such a way that the information needed to classify a given example is reduced as much as possible. The IGR metric is good in performance results in the filter approach, as well as its low computational cost. The information gain ratio is a quantitative measure used to grade the relevance of the features based on the values of such features in the dataset [48].

The IGR measure is based on the Information Gain (IG) measure, which is based on measuring the relative entropy reduction. This method requires discretization to be applied to the continuous data in advance. It is good as a measure to determine the relevance of each feature, because IG biased towards the features with a large number of distinct values [66]. This metric can be computed as shown in the following formula which measures attributes in each record [48].

$IGR(D,A) = \frac{IG(D,A)}{SplitInformation(D,A)}$ Where, "D" training data with "N" features and "A" set of features in the dataset.

Symmetrical Uncertainty: Another strategy to solve the problem of IG bias toward attributes with more values, doing so by dividing by the sum of the entropies of values of X and Y is to use the Symmetrical Uncertainty (SU). A probabilistic model of a nominal valued feature Y can be formed by estimating the individual probabilities of the values where $y \sum Y$ from the training data. If this model is used to estimate the value of Y for a novel sample (drawn from the same distribution as the training data), then the entropy of the model (and hence of the attribute) is the number of bits it would take, on average, to correct the output of the model. Entropy is a measure of the uncertainty in a system. The entropy of Y is given by: $H(Y) = -\sum_{y \sum Y} p(y) \log_2 (p(y)$

If the observed values of Y in the training data are partitioned according to the values of a second feature X, and the entropy of Y with respect to the partitions induced by X is

less than the entropy of Y prior to partitioning, then there is a relationship between features Y and X [68].

$$H(Y \mid X) = -\sum_{x \sum X} p(x) \sum_{y \sum Y} (p(y \mid x) \log_2 (p(y \mid x))$$

SU compensates for information gain's bias toward attributes with more values and normalizes its value to the range [0, 1] using the following Symmetrical uncertainty coefficient= $2 \times [\frac{gain}{H(Y) + H(X)}]$.

Relief and its Extensions: Relief is an instance based feature ranking technique introduced by [75]. ReliefF is an extension of the Relief algorithm that can handle noise and multiclass data sets, and is implemented in the WEKA tool [23] when the WeightByDistance (weight nearest neighbors by their distance) parameter is set as default (false), the algorithm is referred to as RF; when the parameter is set to true, the algorithm is referred to as RFW.

The Relief algorithm [75] is a filter method based on the feature weighting approach that estimates attributes according to their performance in distinguish near instances. Relief searches the two nearest neighbors for each instance; one from the same class ("nearest hit") and another from the miss class ("nearest miss"), defining weights through the expression: U[A]=P(different value of A nearest instance from different class P (different value of A nearest instance from same class). These weights indicate the significance of each feature A to the target concept. The problem of the original Relief algorithm is that it cannot deal with incomplete data and multiclass datasets. To overcome the disadvantages of original algorithm, the extension of ReliefF algorithm was suggested by [76]. This extension algorithm is stronger, meaning it can handle incomplete and noisy data as well as multi-class dataset.

Correlation-Based Feature Selection: One of the most important filter methods Selection (CFS) measure proposed by [68]. The CFS measure considers correlation between a feature and a class and inter-correlation between features at the same time. The

CFS measure evaluates subsets of features on the basis of the following hypothesis:"Good feature subsets hold features highly interrelated with the class, yet uncorrelated to each other". This hypothesis gives rise to two concepts. One is the feature-classification (rcfi) correlation and another is the feature-feature (rfifj) correlation. There exist broadly two measures of the correlation between two random variables: the classical linear correlation and the correlation which is based on information. The feature-classification correlation indicates how much a feature is correlated to a specific class, while the feature-feature correlation is, as the very name says, the correlation between two features. The following equation from [68] gives the merit of a feature subset S consisting of k features:

$$Merits(K) = \frac{rcf1 + rcf2 + ...rcfk}{\sqrt{k + 2(rf1f2 + rf1f3 + ... + rfk1)}}$$

In feature selection, the CFS measure is combined with some search strategies, such as brute force, best first search or genetic algorithm, in order to find the most relevant subset of features. However, the brute force method can only be practical when the number of features is small. When the number of features is large, this method needs massive computational resources. For example, with 50 features the brute force method needs to scan all 2^{50} possible subsets of features [76] that is, unfeasible in general. With best first search, one can deal with high dimensional data sets, but these methods usually give us locally optimal solutions. It is advantageous to get globally optimal subset of relevant features by means of the CFS measure with the expectation of removing more redundant features and still maintaining classification accuracies or even getting better performances. CFS [88] is also highly sensitive to outliers as it uses correlations between features. The time complexity of CFS is quite low. It requires **m** $((n^2-n)/2)$ operations for computing the pairwise feature correlation matrix, where **m** is the number of instances and **n** is the initial number of features [68].

2.6.5. Optimal Feature selection

An optimal subset is always comparative to a certain evaluation function (i.e., an optimal subset chosen using one evaluation function may not be the same as that which uses another evaluation function). Normally, an evaluation function tries to compute the selective ability of a feature or a subset to distinguish the different class labels. Given an inducer, and a dataset D with features $y_1, y_2, ...y_n$, from a distribution D over the labeled instance space, an optimal feature subset, Y_{opt}, is a subset of the features such that the exactness of the induced classifier $C = Z(D)$ is maximal. An optimal feature subset require not be unique because it may be possible to attain the same accuracy using different sets of features (e.g., when two features are perfectly correlated, one can be replaced by the other). By definition, to get the uppermost likely accuracy, the best subset that a feature subset selection algorithm can select is an optimal feature subset. The central problem with using this definition in realistic learning scenarios is that one does not have access to the original distribution and must approximate the classifier's accuracy from the data [71].

The high dimensionality of data poses challenges to learning tasks due to the curse of dimensionality. In the presence of many irrelevant features, learning models tend to overfitt and become less understandable. Feature selection is one valuable means to recognize relevant features for dimensionality reduction [65], [66]. Different studies show that features can be removed with no performance decline. The training data can be labeled, unlabeled or partial labeled, leading to the development of supervised [86], [87], unsupervised and semi-supervised feature selection algorithms.

Many factors affect the success of machine learning on a particular task. The representation and quality of the example data is first and foremost. Theoretically, having more features should result in more discriminating power [88]. However, realistic experience with machine learning algorithms has revealed that this is not always the case. Many learning algorithms can be viewed as making an inclined estimate of the probability of the class label given a set of features. This is a difficult, high dimensional

distribution. Unluckily, learning is often performed on incomplete data. This makes estimating the many probabilistic parameters difficult. In order to avoid over fitting the training data, many algorithms employ the Occam's razor [89] bias to build a simple model that still achieves some acceptable level of performance on the training data. This bias often leads an algorithm to prefer a small number of predictive attributes over a large number of features that, if used in the proper combination, are fully predictive of the class label. If there is too much irrelevant and redundant information present or the data is noisy and unreliable, then learning during the training phase is more difficult.

Feature subset selection is the method of identifying and removing as much irrelevant and redundant information as possible. This reduces the dimensionality of the data and may allow learning algorithms to operate faster and more effectively. In some cases, accuracy on future classification can be improved; in others, the result is a more compact, easily interpreted representation of the intention concept. Recent research has shown common machine learning algorithms to be negatively affected by irrelevant and redundant training information. For example the simple nearest neighbor algorithm is sensitive to irrelevant attributes. Its sample complexity (number of training examples needed to reach a given accuracy level) grows exponentially with the number of irrelevant attributes [54]. Generally, the optimal subset will be the smallest subset of features that can identify instances of a class as consistently as the complete feature set (before feature selection).

CHAPTER THREE

3. DATA COLLECTION AND PREPARATION

The efforts made to establish a knowledge discovery process (KDP) model were initiated in academia when the data mining field was being shaped as methodology to facilitate machine learning approach. Researchers started defining multistep procedures to guide users of data mining tools in the complex knowledge discovery world. The main emphasis is to provide a sequence of activities that would help to execute a KDP in an arbitrary domain [90]. Based on this methodology the chapter examines the dataset used, steps of preprocess employed, algorithms used from data mining tools, evaluation metrics and performance measure of the algorithms with regard to the feature selection and processes to be carried out.

3.1. NSL-KDD Intrusion Detection Dataset

The dataset to be used in this experiment is the NSL-KDD dataset [21] which is a new dataset for the evaluation of researches in network intrusion detection system. It consists of selected records of the complete KDD 99 dataset. NSL-KDD dataset solve the issues of KDD 99 benchmark and connection record contains 41 features. The NSL-KDD training set contains a total of 22 training attack types; with an additional 17 types in the testing set only. Dataset have been criticized for its possible problems [22], but the fact is that it is the most prevalent dataset that is used by many researchers and it is among the few comprehensive datasets that can be shared in intrusion detection nowadays with other researchers by allowing all kinds of techniques to be easily compared in the same baseline. The dataset contains one type of normal data and 39 different types of attacks that are broadly categorized in four groups of DoS, PROBE, U2R and R2L as shown in table 3.1 and 3.2 [22].

3.2. Data Preprocessing

Before data is fed into an algorithm, it must be collected, inspected, cleaned and selected. Since even the best predictor will fail on bad data, data quality and preparation is crucial. Also, since a predictor can exploit only certain data features, it is important to detect which data preprocessing works best [91]. For this study preprocessing of NSL-KDD dataset contains five processes:

i. Assigning attack names to one of the five classes NORMAL, PROBE, DoS (Denial of Service), U2R (User to Root) and R2L (Remote to Local). To identify and label each attack different literatures are consulted and Microsoft Excel helps to filter and name easily using fill handle.

ii. There are records which don't have attributes and removed from the dataset and there is also a mismatch in the KDD 99 winner cost matrix and the confusion matrix; as a result arrangements are made to match the cost matrix and confusion matrix.

iii. The NSL-KDD dataset is available in text format; so to be read by TANAGRA and WEKA tool it has to be changed into ARFF format, but both can support other formats too that is TANAGRA .xls and CSV and WEKA CSV and data bases.

iv. Finally, rank and select feature from the 41 using WEKA as preprocessing to identify important ones for final use to build models.

3.3. Tools used for the Experiment

3.3.1. Microsoft Excel

It is currently the most widely used spreadsheet for Microsoft Windows and other operating systems integrated as part of Microsoft Office that allows manipulating numerical and alphanumeric data. Because of the versatility of modern spreadsheets, they are used sometimes to make smaller databases, reports, and other uses with extension ".xls" which can be supported by TANAGRA. In our work we use for labeling the attack types in to four categories and also identify the missing values both in NSL-KDD training and testing dataset by importing the records which are comma separated text format. The

dataset in a spreadsheet (Excel) format save a lot of time in preprocess. Rows are records of connection; columns are attributes of records (duration, protocol type, service, src_bytes, dst_bytes, etc) and each cell should include one value only [92].

Classification of Attack	**Attack Name**
Probing	Port-sweep, IP-sweep, Nmap, Satan
Denial of Service (DoS)	Neptune, Smurf, Pod, Teardrop, Land, Back
User to Root (U2R)	Buffer-overflow, Load-module, Perl, Rootkit, spy
Remote to Local (R2L)	Guess-password, Ftp-write, Imap, Phf, Multihop, Warezmaster, Warezclient

Table 3.1: Attack types in NSL-KDD training dataset and their categorization

Classification of	**Attack Name**
Probing	Port-sweep, IP-sweep, Nmap, Satan, **Saint, Mscan**
Denial of Service (DoS)	Neptune, Smurf, Pod, Teardrop, Land, Back, Apache2, **Udpstorm, Processtable, Mail-bomb**
User to Root (U2R)	Buffer-overflow, Load-module, Perl, Rootkit, **Xterm, Ps, Http-tunnel, Sqlattack, Worm, Snmp-guess**
Remote to Local (R2L)	Guess-password, Ftp-write, Imap, Phf, Multihop, Spy, Warezmaster, Warezclient **,Snmpgetattack, Named, Xlock, Xsnoop, Send-mail**

Table 3.2: Attack types in NSL-KDD Testing dataset and their categorization

NB: attack names in the above table with bold exists only in the testing dataset

Table 3.3 shows number of instances in the full NSL-KDD training datasets as well as the test dataset along with their respective number of attacks and normal. All attack types are mapped to their respective categories while the non-intrusive instances are labeled as normal.

Dataset Label	normal	probe	DOS	U2R	R2L	Total Attack	Total Normal	Total
Training data	67,343	11,656	45,927	52	9,95	58,630	67,343	**125,973**
Testing data	9,711	2,421	7,458	2,00	2,754	12,831	9,711	**22,544**
Total of total								**148,517**

Table 3.3: NSL-KDD Training and Testing data Connection Distributions

From table 3.3 the proportion of signatures in DoS and Probe attacks in the testing set are very similar to those present in the provided training set compared to types of attack of U2R and R2L which differ significantly between the training and the testing sets. In the testing set, over 80% U2R attacks and 30% R2L attacks are new to the training set.

3.3.2. TANAGRA Data Mining Tool

TANAGRA was written as a support to education and research on data mining, which intended to be a free, open-source, user-friendly piece of software for students and researchers to mine their data. It can import text files with whitespace delimited fields. Each record or transaction appears on its own line [51]. There is also a conversion tool available on the TANAGRA website to convert from ARFF files, the format used by WEKA to .xls and CSV. Data splitting in TANAGRA have to be gathered in the same sheet or file since it is not possible to specify two data sources and for this work we added a new column, which enables us to distinguish training set and test set (STATUS) as shown in (appendix E). Direct cost sensitive classifier which is implemented only in TANAGRA and indirect (metacost) in both as far as our knowledge concerns and used by [19] also used for this work.

3.3.3. WEKA Data Mining Tool

The WEKA workbench [44] contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality. A command line interface is also included, for larger scale processing. It was originally designed as a tool for analyzing data from agricultural

domains but is now used in many different application areas, largely for educational purposes and research. The main strengths of WEKA are that it is (a) freely available under the General Public License (GNU), (b) very portable because it is fully implemented in the Java programming language and thus runs on almost any computing platform, (c) contains a comprehensive collection of data preprocessing and modeling techniques, and (d) is easy to use by a novice due to the graphical user interfaces it contains. In this work used for feature selection purpose because as far as our knowledge the metacost sensitive attribute selection approaches are employed in WEKA tool for feature ranking and selection.

In using WEKA for data preparation separate dataset has to be prepared. In this work ARFF file format is used during feature selection and training and testing dataset merged together to build model in TANAGRA. The NSL-KDD dataset is available in text format to change into ARFF format which can be read by the tools, header information is added in training datset separately for feature selection and both training and test dataset combined together. This is accomplished by first converting the file to Comma Separated Value (CSV) format using Excel.

- **Headline**: The relation name is defined as the first line in the ARFF file. The format is: **@ Relation** <name-of-relation> where <relation-name> is a string. The string must be quoted if the name includes spaces.
- **Declaration of attributes**: Attribute declarations take the form of an ordered sequence of @**attribute statements**. Each attribute in the data set has its own attribute statement which uniquely defines the name of that attribute and its data type. The order the attributes are declared indicates the column position in the data section of the file. For example, if an attribute is the third one declared then WEKA expects that all that attributes values will be found in the third comma delimited column. The format for the attribute statement is: **@ attribute** <attribute-name> <data type> <attribute-name>: must start with an alphabetic character. If spaces are to be included in the name then the entire name must be

quoted and data type can be any of the three types currently (version 3.6.4 WEKA) can support (see appendix B: full attribute declaration):

- **Numeric or Real.** Numeric attribute can be real numbers.
- **Integer.** Integer attribute can be integer numbers.
- **Date**. Date attribute is an optional string specifying how date values should be parsed and printed.
- **String.** String attributes allow us to create attributes containing arbitrary textual values.
- **Enumerate.** Enumerate attribute consists of a set of possible values separated by commas (Characters or strings), which can take the attribute. For example, in the dataset we have an attribute that indicates the class that can be expressed: @ attribute class {NORMAL, PROBE, DOS, U2R and R2L}.

- **Data Section**: The data section of the file contains the data declaration line and the actual instance lines. The **@data** declaration is a single line denoting the start of the data segment in the file. Each instance is represented on a single line, with carriage returns denoting the end of the instance (but it can work without in this case). Attribute values for each instance are delimited by commas. They must appear in the order that they were declared in the header section (i.e. the data corresponding to the n^{th} @attribute declaration is always the n^{th} field of the attribute) (see appendix B: sample data declaration).

3.4. Evaluation Metrics

A cost matrix (C) is defined by associating classes as labels for the rows and columns of a square matrix; in the current context for the NSL-KDD dataset, there are five classes,{NORMAL, PROBE, DOS, U2R, R2L},and therefore the matrix has dimensions of 5×5. An entry at row i and column j, C (i, j), represents the non-negative cost of misclassifying a pattern belonging to class i into class j. Cost matrix values employed for the KDD 99 dataset are defined in [16]. These values were also used for evaluating results of the KDD 99 competition. The magnitude of these values was directly

proportional to the impact on the computing platform under attack if a test record was placed in a wrong category. A confusion matrix (CM) is similarly defined in that row and column 5×5 matrix for the KDD 99 dataset. An entry at row i and column j, CM (i, j), represents the number of misclassified patterns, which originally belong to class i yet mistakenly identified as a member of class j.

The form of the cost matrix C will depend on the actual application. In general, it is reasonable to choose the diagonal entries equal to zero, i.e. C (i, j) = 0 for i = j, since correct classification normally incurs no cost as shown in table 3.4. In addition the size of the cost matrix should be the same as that of the confusion matrix.

Class distribution of the proposed work training dataset	Cost matrix					
		normal	probe	DOS	U2R	R2L
67,343(53.5%)	normal	0	1	2	2	2
11,656 (9.3%)	probe	1	0	2	2	2
45,927 (36.5%)	DOS	2	1	0	2	2
52 (0.04%)	U2R	3	2	2	0	2
995 (0.8%)	R2L	4	2	2	2	0

Table 3.4: The 5X5 cost matrix used for the KDD 1999 winner result

3.4.1. Standard Metric to Evaluate Intrusions

To evaluate the approach, the four standard metrics of true positive, true positive, false positive and false negative developed for network intrusions, have been used. Table 3.5 shows these standard metrics.

Confusion metrics (standard metrics)		Predicted Connection label	
		Normal	Intrusions(Attacks)
Actual Connection	Normal	True Negative(TN)	False Alarm(FP)
	Intrusions	False Negative(FN)	Correctly detected Attacks (TP)

Table 3.5: Standard metrics for evaluations of Intrusions (attacks)

The denotations of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are defined as follows:

- True Positive (TP): The number of malicious records that are correctly identified.
- True Negative (TN): The number of legitimate record that correctly classified.
- False Positive (FP): The number of records that are incorrectly identified as attacks however in fact they are legitimate activities.
- False Negative (FN): The number of records that are incorrectly classified as legitimate activities however in fact they are malicious.

3.5. Performance Measure

General performance of intrusion detection systems is compared in terms of numbers of selected features and the classification accuracies of the machine learning algorithms giving the best classification results. As intrusion detection systems require high detection rate, low false alarm rate and lower average misclassification cost, thus we compare overall classification rate (OCA), detection rate (DR), false Positive rate (FPR) , average misclassification cost (AMC), Error rate, and training and testing time and present the comparison results of various attacks .

Error Rate

The error rate, which is only an estimate of the true error rate and is expressed to be a good estimate, if the number of test data is large and representative of the population and is defined as [93]:

$$\text{Error Rate} = \frac{TotalTestsamples - TotalCorrectlyclassifiedsamples}{TotalTestsamples} \times 100\%$$

Accuracy

Overall Classification accuracy (OCA) is the most essential measure of the performance of a classifier. It determines the proportion of correctly classified examples in relation to the total number of examples of the test set i.e. the ratio of true positives and true negatives to the total number of examples. From the confusion matrix, we can say that accuracy is the percentage of correctly classified instances over the total number of

instances in total test dataset, namely the situation TP and TN, thus accuracy can be defined as follows [94]:

$$\text{Accuracy}=\frac{Tota\ln umberofcorrectlyclassifiedsamples}{Tota\ln umberoftestsamples}\times 100\%$$

Or

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}\times 100\%$$

Detection Accuracy

Detection accuracy (rate) refers to the proportion of attack detected among all attack data, namely, the situation of TP, thus detection rate is defined as follows [94]:

$$\text{Detection Accuracy}=\frac{no.ofsamplesclassifiedcorrectly}{no.ofsamplesusedfortesting}\times 100\%$$

Or

$$\text{Detection Accuracy} = \frac{TP}{TP+FN}\times 100\%$$

False Positive Rate

Another name is False Alarm Rate (FAR) measures the number of misclassified positive instances in relative to the total number of misclassified instances. Can be expressed also the proportion that normal data is falsely detected as attack behavior, namely, the situation of FP, thus false alarm rate is defined as follows [94]:

$$\text{False Positive Rate}=\frac{tota\ln o.ofmissclassifiedsamples}{tota\ln o.oftestsamples}\times 100\%$$

Or

$$\text{False alarm rate} = \frac{FP}{FP+TN}\times 100\%$$

Recall and precision are two widely used metrics employed in applications where successful detection of one of the classes is considered more significant than detection of the other classes [95].

Precision

Precision is the number of class members classified correctly over the total number of instances classified as class members. Technically can be expressed the attack has been occurred and the IDS detects correctly [95].

$$\text{Precision} = \frac{TP}{TP + FP} \times 100\%$$

Recall

Also called True Positive Rate (TPR) measures the number of correctly classified examples relative to the total number of positive examples. In other words the number of class members classified correctly over the total number of class members [95].

$$\text{Recall} = \frac{no.ofcorrectlyclassified\,\text{int}\,rusions}{no.of\,\text{int}\,rusions} \times 100\%$$

Or

$$\text{Recall} = \frac{TP}{TP + FN} \times 100\%$$

Average Misclassification Cost

$$\text{AMC} = 1/N \sum_{i=1}^{5} \sum_{j=1}^{5} \text{CM (I, J)*C (I, J)}$$

Where CM corresponds to confusion matrix, C corresponds to the cost matrix, and N represents the number of patterns tested [16].

CHAPTER FOUR

4. EXPERIMENTATIONS AND DISCUSSIONS

This chapter describes experimental study of the algorithm, which is described in the previous chapter, on various feature selection approaches for direct and indirect cost sensitive learning algorithms. In the experiments one dataset, two feature selection algorithms to select and rank feature and two machine learning decision tree algorithms to compare the performance of selected features are used. Data mining tools TANAGRA and WEKA which uses a collection of machine learning tools and data analysis methods from exploratory data analysis, statistical learning, mathematical learning and database learning are employed.

4.1. Experimentation Setup

All experiments are performed in a computer with the configurations Intel(R) Core(TM) 2 CPU 2.00GHz, 2 GB RAM, and the operating system platform is Microsoft Windows 7. TANAGRA version 1.4.37 [51] and WEKA (the latest stable Windows version 3.6.4) [23] are used. For WEKA the default memory value 128m for maxheap option was increased to 1024m because of the memory heap problem for 125,973 records during feature selection. The two tools are collections of machine learning algorithms for data mining tasks that contain facilities for data preprocessing, classification, regression, clustering, association rules, and visualization. This empirical study, however, only deals with a subset of classifier algorithms of TANAGRA for classification and WEKA for feature selection. In addition Microsoft Excel is used to filter records for manual labeling of the five classes from the NSL-KDD dataset. There are four steps in our experiments which are taken step by step as listed below:

a. In the beginning, in order to build the experiment environment arrangement, deciding the data mining software and dataset used are decided.
b. Preprocessing of the dataset.

c. Feature ranking and selection of relevant features and exclude unimportant features. After processing, the number of features of the NSL-KDD dataset is narrowed down from 41 to average 25 and 24 on each cost sensitive and insensitive feature ranking and selection algorithms respectively.
d. Train the cost sensitive CS-CM4 and metacost sensitive C4.5 algorithms to generate models on the training set for each and to build predicated instance to classify test data test for each of the selected attributes on each algorithms.
e. Lastly, the performance comparison between the two chosen Classifiers in relation to feature selection algorithms with other recent work results are discussed and presented.

4.2. Experimentations on feature ranking and selection

Feature selection and ranking [96] are an important issue in intrusion detection. The question which features are truly useful, which are less significant, and which may be useless? From large number of features that can be monitored for intrusion detection purpose is relevant because the elimination of useless features enhances the accuracy of detection while speeding up the computation, thus improving the overall performance of IDS. In cases where there are useless features, by concentrating on the most important ones may well improve the time performance of IDS without affecting the accuracy of detection.

The second tasks according to the above steps in this experimentation are feature ranking and selection. Therefore, the feature subset evaluating methods are more suitable for selecting features for intrusion detection. A major challenge in the IDS feature selection process is to choose appropriate measures that can precisely determine the relevance of features to the intrusion detection task and the relationship between features of a given dataset. For the experiments two filter approaches on 125,973 records of training dataset are used:

1. Ranker search with Information Gain Ratio (IGR) as a measure to select relevant attributes based on information gain.
2. Genetic search with Correlation based Feature Selection (CFS) as subset evaluating mechanism to measure correlation percentage of feature-feature and feature-class.

The experiments described in this thesis compare runs of selected direct and indirect cost sensitive machine learning algorithms with cost sensitive and insensitive feature selection on the datasets described in the previous chapter. The feature selection here is based on supervised learning methods. For comparison purposes, algorithms which are available in WEKA [23] Information Gain Ratio (IGR) which overcomes the problem of favoring attributes with many values over those with few values and show good results in the filter approach as well as its low computational cost [48], and Correlation-based Feature Selection (CFS) with genetic search effective in relations between feature-feature and class-feature and searches feature subsets according to the degree of redundancy among the features [68]. Genetic search is a stochastic general search method, capable of effectively exploring large search spaces, which is usually required in case of attribute selection [84]. Further, unlike many search algorithms, which perform a local, greedy search, genetic search performs a global search [85]. Generally both algorithms are from the family of filter which ranks features independently without involving any learning algorithm. The details of these functions available in WEKA [98] and cited references in the literature review.

4.2.1. Parameter settings for WEKA

With both feature ranking and selection algorithms, there are often a large number of parameters that can be adjusted. However in this work all parameters and their values are applied using the default parameters defined in the WEKA tool except the modification in the setup. This helps to compare and measure the performances of the algorithms without any modification. Table 4.1 shows components of WEKA used for the experiment.

Tab	Operator	Comment
Preprocess	Filter	Specify the method of selection
Attribute selection	CostSensitiveAttributeEval CostSensitiveSubsetEval	Making the feature selection algorithm cost sensitive
Attribute Evaluator	IGR and CFS	To evaluate the worth of attribute
Search Methods	Ranker and Genetic	To search the space of attribute

Table 4.1: Components of WEKA used for both feature selection approaches

An important step of using a feature selection method is to setup the stopping criterion, which determines when the feature selection algorithm stops and concludes that the subset found at that point is relevant. In the case of IGR, WEKA produced a ranked list of the features without performing the actual feature selection based on their information gain value. Due to this fact, best stopping criteria after an extensive empirical study are determined to select from the ranked lists as used by [68], [19].

4.2.2. Cost insensitive feature ranking

Using the training data set, we rank the features according to the information gain value assigned by the IGR with ranker search and correlation of feature-feature and feature-class assigned by CFS with genetic search strategy. The rank is based on cost insensitive (without cost matrix). After extensive experiment the features are identified based on their information gain value and correlation percentage.

4.2.3. Cost sensitive feature ranking

It is a metacost subset evaluator that makes its base subset evaluator cost sensitive. There is an option that asks file name of a cost matrix to use. If this is not supplied, a cost matrix will be loaded on demand. The name of the on-demand file is the relation name of the training data plus ".cost", and the path to the on-demand file is specified with the option and the directory to search for cost files when loading costs on demand (default current directory) must be supplied. The cost file name is the matrix of the class size like

the confusion matrix. In this case the KDD 99 winner cost matrix shown in table 3.4 is saved in a separate file used [97].

In both cases the metacost sensitive feature selection algorithms in comparison with cost insensitive algorithms that obtains a training set and outputs a subset of relevant features according to their relevance rank based on their information gain value and correlation percentage as shown in (appendix D). The worth of the attribute subset is determined using the full set of training data and also attribute subset is determined by a process of 10fold cross-validation. The fold and Seed fields set were left as default in both cases. The numbers given to the attribute in (appendix D) and table 4.2 are mapped to the name of attributes indicated in (appendix A) and full WEKA experiment result (Appendix B and C).

4.3. Cost Sensitive and insensitive Feature Selection Approaches

After repeated empirical investigation based on both approaches using WEKA each feature receives a rank and correlation percentage representing its expected value for the classification task. After feature ranking we do feature selection, by applying the top ranked features one by one as long as the accuracy of selected model is non-decreasing on features rank in both approaches. We stop when the accuracy drops, as an indication of model overfitting and follow this procedure on selected algorithms by varying the numbers of inputs. The variation focuses on the gain value and correlation percentage which have remarkable value and correlation with repeated experimentation as shown in the summary table (appendix D).

Experiments with the highest ranked features on different gain value and correlation as accurate as using the original set are considered. However, it relies on the user to decide how many features to include from the ranked list in the final subset based on the corves. Due to the lack of an analytical model, only we determine the relative importance of the input variables through empirical methods. Complete analysis would require examination of all possibilities, example taking two variables at a time to analyze their gain value or correlation, then taking three at a time, etc. This, however, is both infeasible (requiring 2^n

experiments). Here more investigation is required to have pattern even though difficult in such domains where there is class imbalance. By convention, features that score high are indicative of a valuable variable and sorted in decreasing order.

Eventually, after obtaining the ranked set of features through the IGR and CFS the optimal subset of features are determined by the CS-CM4 and C4.5 (as metacost). The top ranked features are then used to build models. Features are selected that give better performance than previous models. After iteration of the classifiers the most relevant feature in accordance to the IGR and CFS, are added to the optimal subset of features. List of features with remarkable gain value and correlation percentage are shown in table 4.2 and for full result of WEKA (see Appendix C).

Feature Selection algorithm	Search strategy	Gain value/ Correlation percentage	Features selected for model building	No. of features
Cost insensitive feature selection approach				
IGR	Ranker	>=0.15	26,25,4,12,39,30,38,6,5,29,3,11,22,37,35,8,14,10,34,31,33,27,23	23
		>=0.123	26,25,4,12,39,30,38,6,5,29,3,11,22,37,35,8,14,10,34,31,33,27,23,9,2,41,36,17,32,18	30
CFS	Genetic	>=50%	2,3,4,5,6,12,15,23,26,29,30,35,37,38,39	15
		>=1%	2,3,4,5,6,12,15,23,26,29,30,35,37,38,39,1,8,9,13,14,17,21,22,24,25,31,32,34,36	29
Cost sensitive feature selection approach				
IGR	Ranker	>=0.189	26,24,25,12,39,38,30,5,6,29,11,22,3,37,35,10,14,8,9	19
		>=0.114	26,24,25,12,39,38,30,5,6,29,11,22,3,37,35,10,14,8,9,34,31,33,27,23,2,17,18,41,36,32 ,28,1,13,40	34
CFS	Genetic	>=50%	3,4,5,6,12,17,25,26,29,30,31,32,34,35,37,38,39	17
		>=1%	3,4,5,6,12,17,25,26,29,30,31,32,34,35,37,38,39,2,7,9,11,13,14,15,18,19,21,22,23,24,	31

Table 4.2: Top ranked features using information gain value and CFS correlation percentage

4.4. Training Phase for the Experiments

In the model building the first phase is the training phase. It involves training of CS-CM4 and C4.5 algorithms using the selected features from the training dataset. For the experiment, direct cost sensitive decision tree algorithm CS-CM4 and the standard C4.5 (cost insensitive) version of CS-CM4 by making indirect cost sensitive (as metacost). The metacost handles the costs during the classification phase, without modification of the learning algorithm [15]. We use C4.5 as metacost decision tree learner for empirical comparisons and it is used as the base learner by [15] for the metacost method.

These two variant algorithms represent two different approaches to learning and state-of-the-art algorithms that are often used in data mining applications. To save space confusion matrix for models build on the training data are not presented. For the classifier the full set features as well as the features sets obtained by removing irrelevant features by means of IGR and CFS measures are presented. Subsequently, we used the features inside each selected optimal subset of features to feed on decision trees algorithms.

4.4.1. Parameter settings for TANAGRA

In all model building and testing, there are often a large number of parameters that can be adjusted. However in this work all parameters and their values used default except the cost metrics shown in table 4.4. This helps to compare and measure the performances of the algorithms without any modification and details are shown on table 4.5 and 4.6. Table 4.3 also shows components and operators used during model building and testing.

Tab	Operator	Comment
Instance selection	Discreet sample selection	Specify the training and test set
Feature selection	Define status	Specify the attributes to use
SPV learning	CS-CM4 and C4.5(as meta cost sensitive)	To build models
SPV learning assessment	Test	To test the models on unselected instances

Table 4.3: Components of Tanagra used in the experiment for model building and testing

Misclassification Cost Matrix					
	Observed vs. Predicted				
Cost(i,j)	1	2	3	4	5
1	0	1	2	2	2
2	1	0	2	2	2
3	2	1	0	2	2
4	3	2	2	0	2
5	4	2	2	2	0

Table 4.4: Cost matrix setting for CS-CM4 and metacost sensitive C4.5 decision tree

Cost sensitive C4.5 parameters	
Min size of leaves	5
Lambda for laplacian	3

Table 4.5: Default parameters for cost sensitive C4.5 as direct cost sensitive decision tree

Decision tree (C4.5) parameters	
Min size of leaves	5
Confidence-level for pessimistic	0.25

Table 4.6: Default parameters for C4.5 as metacost decision tree

4.5. Testing Phase for the Experiments

To compare the effectiveness of feature selection, feature sets chosen by each approach are tested with direct cost sensitive decision tree CS-CM4 and indirect (metacost) sensitive decision tree learner C4.5 on testing dataset. The output of each test is presented in a confusion metrics. Comparison results of Detection Accuracy (Rate) (which refers to the proportion that a type of data is correctly classified) of normal and four different attacks (Probe, Dos, U2R, R2L), False Positive Rate (FPR), Average Misclassification Cost (AMC), Overall Classification Accuracy (OCA) and Time Taken to build Model (TTBM) are included at the bottom of each confusion metrics. The metrics are attached at the appendix using all features (Appendix F), cost sensitive approach (appendix G) and cost insensitive approach (Appendix H) accordingly. Tests are done before (41) and after feature selection to validate and compare the approach.

4.6. Result Analysis and Discussions

In the result analysis and discussion section the experimental results are analyzed and discussed. Comparisons of the feature selection results yielded by the two types of approaches are used to determine the best method for intrusion feature selection. The classification results of each classifier are analyzed based on suggested metrics and evaluated to show the performance of the best classifier in relation to each feature selection approach.

4.6.1. Result Analysis using Detection Accuracy

Table 4.7 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 (Probe, Dos, U2R, R2L) based on CS-MC4 and C4.5 classifiers and IGR and CFS feature selection algorithms and all the features in comparison with recent works shown in table 4.10.

Cost insensitive feature selection approach											Cost sensitive feature selection approach									
Algorithms	NORMAL		PROBE		DOS		U2R		R2L		NORMAL		PROBE		DOS		U2R		R2L	
	CS-CM4	C4.5	CS-CM4	C4.5	CS-CM4	C4.5	CS-CM4	C4.5	CS-CM4	C4.5	CS-CM4	C4.5	CS-CM4	C4.5	CS-CM4	C4.5	CS-CM4	C4.5	CS-CM4	C4.5
IGR-1	98.7	99.6	98.8	99.2	99.4	99.7	79.8	88.3	91.6	90.0	98.5	99.5	99.1	99.3	99.6	99.7	88.3	90.4	**94.8**	93.8
IGR-2	98.8	99.5	**99.3**	90.0	99.6	**99.8**	**91.5**	92.6	**93.2**	93.1	99.1	99.5	99.2	**99.4**	99.6	**99.8**	**89.4**	**91.5**	91.4	93.8
CFS-1	98.6	99.5	99.1	99.2	**99.7**	**99.8**	68.8	**91.5**	90.8	91.8	98.3	99.3	98.9	98.7	**99.8**	**99.8**	85.1	**91.5**	91.0	92.8
CFS-2	98.8	99.5	**99.3**	99.2	99.6	**99.8**	89.4	90.4	91.8	**93.6**	98.9	99.5	99.1	99.1	99.6	**99.8**	**89.4**	89.4	91.4	93.6
All	98.7	99.5	**99.3**	**99.4**	99.6	**99.8**	82.9	**91.5**	88.9	94.0	98.7	99.5	**99.3**	**99.4**	99.6	**99.8**	82.9	**91.5**	88.9	**94.0**

Table 4.7: Comparison of detection accuracy of each feature selection approach

KEYS

IGR-1 =23 (cost insensitive) and 19 (cost sensitive) features

IGR-2=30 (cost insensitive) and 34 (cost sensitive) features

All=41 features

CFS-1=15 (cost insensitive) and 17 (cost sensitive) features

CFS-2=29 (cost insensitive) and 31 (cost sensitive) features

4.6.1.1. Detection Accuracy for Cost Sensitive Feature Selection Approach

Tests are done on the models developed for the proposed approaches. Feature selection algorithm IGR and CFS by making cost sensitive on CS-CM4 and C4.5 learning algorithm are used. Features are taken at different information gain ratio value for the IGR (0.189, 0.114) to select (19 and 34 features). In the same way different correlation percentage values are taken for CFS (50%, 1%) to select (17 and 31 features) as shown in table 4.2. In this section, Detection Accuracy (Rate) of CS-CM4 and C4.5 built using cost sensitive feature selection approach on IGR and CFS in relation to each attack types are discussed.

a) For PROBE and DOS attack: Detection accuracy of C4.5 is better than that of CS-CM4 in all cases except when CS-CM4 combined with CFS-17 features which is comparable. For the two attacks compared to using 41 features both IGR-34 and CFS-17 and 31 features perform equal 99.8% for C4.5 classifier; whereas CS-CM4 perform better than using 41 features when combined with CFS-17 features. Here the gain is the reduction in feature space from 41 to 34 for IGR and 41 to 17 for CFS. For PROBE attack this approach achieved better accuracy 99.6 % compared to [19] except C4.5 combined with CFS-17 features achieved detection accuracy 98.7 % with minimal difference. For DOS attack the result is comparable with [19] with minimal value difference 0.1%.

b) For U2R attack: The C4.5 detection result is better than CS-CM4 in all cases except both combined with CFS-31 features perform same detection accuracy of 89.4%. CS-CM4 achieved the same detection accuracy when combined with IGR-34 and CFS-31 features. Compared to using 41 features, 82.9% accuracy for CS-CM4 classifier, best detection accuracy of 89.4% when combined with IGR-34 and CFS-31 features is achieved. Generally in all cases accuracy is better when feature selection applied. Whereas C4.5 perform better using all features except combined with IGR-34 features and CFS-17 features result equal accuracy 91.5 % with feature space reduction 41 to 17 which require less computational cost. For U2R attack this approach achieved better accuracy compared to [19] in all cases.

c) For R2L attack: According to the best detection result, CS-CM4 outperforms its counterpart C4.5. In terms of feature selection IGR perform better accuracy for CS-CM4 classifier when combined with IGR-19 features with accuracy of 94.8%; in the rest case comparable result is achieved. For C4.5 classifier in all feature selection cases achieved comparable result. Detection accuracy is better for CS-CM4 compared to using 41 features. But for C4.5 using all feature shows slight increase of 0.2% and 0.3 % compared to using 19 features for the IGR and 31 features for CFS. The gain here is feature space reduction of 22 and 10 for the two cases respectively.

4.6.1.2. Detection Accuracy for Cost insensitive feature Selection Approach

Table 4.7 shows summary of detection accuracy of models built using features selected based on cost insensitive and sensitive approaches. Here in this section the cost insensitive approaches are discussed in terms of attack types. In this case features are selected without cost information gain ratio value (0.15, 0.123) to select (23 and 30 features) and correlation percentage (>=50%, >=1%) to select (15 and 29 features) as presented in table 4.2.

a) For PROBE and DOS attack: Detection accuracy (DR) of C4.5 is better than that of CS-CM4 in all cases except when IGR-23 and CFS-29 features are used. But numbers of features are large compared to the CS-CM4 classifier using 23 and 15 features respectively. This indirectly needs more computational cost. For the two attacks CFS based feature selection achieved better compared to IGR in detection accuracy in all cases. Compared to using 41 features both IGR-30 and CFS-29, achieved equal detection accuracy 93.3 % for CS-CM4. C4.5 using 41 features result is comparable to using IGR-30 and CFS-29 features. For PROBE attack this approach achieved better accuracy 99.3 % compared to [19] except CS-CM4 combined with IGR-23 features achieved equal result 98.8%. On the other hand for DOS attack [19] result performs better with accuracy of 99.9 %.

b) For U2R attack: The C4.5 detection result is better than CS-CM4 in all cases and achieved 92.6% using IGR-30 features. CS-CM4 detection accuracy 91.5% for IGR-30 is

comparable as shown in table 4.7. From the table we can say IGR feature selection outperforms its counterpart CFS in all cases. Compared to using 41 features for the two classifier 82.9% and 91.5% accuracy, IGR-30(91.5%) and CFS-29 (89.4%) perform better for CS-CM4 classifier; whereas C4.5 only when combined with IGR-30 (92.6%). For U2R attack this approach achieved 89.4 % accuracy which is better compared to [19] (72.5 %) in all cases and this is the most dangerous attack and one of the focus of this study.

c) For R2L attack: According to the detection result, CS-CM4 combined with IGR-23(91.6%) and 30(93.2%) features perform better than C4.5. On the other hand C4.5 combined with CFS- 15(91.5%) and 29(93.6%) outperform CS-CM4. In all cases for this type of attack the CFS feature selection with C4.5 decision tree is better. For this attack in all feature selection cases detection accuracy is better for CS-CM4 compared to using 41 features. In comparison with [19] work (91.7%) a detection accuracy of 94.0% is achieved for C4.5 using 41 features.

Generally, in both approaches CFS feature selection approach selects the smallest number of relevant features in comparison with IGR and the full feature sets without degrading the accuracy. In addition except U2R attack, features selected using CFS is found to result better detection accuracy for PROBE, DOS and R2L attacks. Especially in some cases, our new method compresses the full set of features extremely. For example, only 15 features are selected out of 41 features in the case of CFS and 19 for IGR with comparable accuracy to the rest cases. This is a great save in computational time and space while increasing performance without degrading classification accuracy as evident from table 4.7.

4.6.1.3. Detection Accuracy comparison for U2R and R2L attacks

Table 4.8 shows detection accuracy of U2R and R2L attacks for the proposed approach. Because, misclassification costs are often closely related with imbalanced distribution of class values in the data set (rare classes usually being of higher interest); close investigation is important on theses dangerous attacks. In this case U2R and R2L are the rare classes that require cost matrix to identify minority class values which have unfrequent pattern records. The experiment output expected rankings and correlation percentage according to the assigned weights on the training dataset that give more weight to minority classes than the majority classes as shown in table 3.4.

Algorithms	Cost insensitive Approach					Cost sensitive Approach				
	Attack type	U2R		R2L		Attack type	U2R		R2L	
	no. of features	CS-CM4	C4.5	CS-CM4	C4.5	No. of features	CS-CM4	C4.5	CS-CM4	C4.5
IGR	23	78.8	88.3	91.6	90.6	19	88.3	90.4	**94.8**	93.8
IGR	30	**91.5**	**92.6**	**93.2**	93.1	34	**89.4**	**91.5**	91.4	93.8
CFS	15	68.8	91.5	90.8	91.8	17	85.1	**91.5**	91.0	92.8
CFS	29	89.4	90.4	91.8	93.6	31	**89.4**	89.4	91.4	93.6
-	41	82.9	91.5	88.9	**94.0**	41	82.9	**91.5**	88.9	**94.0**

Table 4.8: comparison of the two approaches for two attacks parallel with all features

For the two especial attacks, U2R and R2L, CS-CM4 combined with IGR cost sensitive feature selection approach outperform the cost insensitive approach using 19 and 23 features. Even the numbers of features are reduced from 41 to 19 for cost sensitive approach whereas the insensitive reduced 41 to 23 to achieve detection accuracy of 88.3 % and 78.8 % respectively. Similarly C4.5 has accuracy of 90.4% for cost sensitive and 88.3 % for the insensitive. Compared to all features both approaches for the C4.5 classifier result a better accuracy. But for CS-CM4 classifier only for cost sensitive approach detect better accuracy.

IGR when using 34 for cost sensitive and 30 for insensitive on classifiers, the cost insensitive outperform the sensitive ones. From this we can say the IGR can rank features related to the two attacks at the top. In addition using cost helps to identify the most relevant at the top with minimum number of features without degrading the others performance.

CFS-17 features on cost sensitive approach and 15 features for insensitive ones, CS-CM4 result a detection accuracy of 85.1 % and 68.8 % for U2R attack respectively. The accuracy gap is maximal 6.3 % and the feature space reduction is negligible compared to the first gain value. C4.5 classifier in both cases the accuracy is comparable as shown in table 4.8.

Using 31 features for cost sensitive and 29 for insensitive, CS-CM4 achieved accuracy of 91.4 % and 91.8 % for R2L attack respectively; whereas U2R the same 89.4 % in both approaches. For CFS feature selection better accuracy achieved when correlation of the two attacks are below 50% and greater than 1% as we can see from table 4.3. We should mention here that the highest ratio for the U2R and R2L class has never exceeded 72.5% and 91.5% respectively according to the different works results [16], [99], [100], [7], [19] available in the literature as indicated in table 4.10. But using this approach the two attacks are detected as an attack with a detection accuracy of 89.4 % and 94.8 % for CS-CM4 and 91.5 % and 94.0% for C4.5 classifiers. Generally from empirical results of U2R and R2L classes which have small training data this approach using decision tree by far gives better performance.

4.6.2. Performance Result Analysis using (FPR, OCA, AMC, TTBM)

Table 4.9 summarizes the efficiency of information gain value and correlation percentage for feature selection and the performance comparison of CS-MC4, C4.5 classifier and other recent works shown in table 4.10 based on the number of features, overall classification accuracy, average misclassification cost, false positive rate, training and testing time. The comparison is based on cost sensitive and insensitive feature selection

approach. The logic using these many performance measurement metrics is one of the achievements of the study because using only accuracy or detection rate gives little information about the algorithms. For example, the run time on training which will give us the idea about which algorithm can be implemented in a real-time network intrusion detection system. In addition using only classification accuracy as the only performance measure, is in practice not necessarily optimal, as the feature selectors that are given more execution time may have a higher accuracy, but also an overall higher cost.

Algorithms	After Cost insensitive feature selection approach						After Cost sensitive feature selection approach					
	No. of features	Average misclassification cost	Overall Accuracy (%)	False alarm rate (%)	Training time (ms)	Testing time (ms)	No. of features	Average misclassification cost	Overall Accuracy (%)	False Alarm rate (%)	Training time (ms)	Testing time (ms)
C4.5-IGR	23	0.158	99.32	0.674	7722	63	19	**0.0130**	**99.41**	0.589	6677	63
C4.5-IGR	30	**0.0140**	99.37	0.621	11981	**47**	34	0.0132	**99.41**	**0.585**	11014	63
C4.5 –CFS	15	0.0317	99.27	0.727	4665	63	17	0.0142	99.38	0.616	**4150**	63
C4.5-CFS	29	0.0143	**99.38**	**0.616**	8596	63	31	0.0142	99.38	0.612	8892	**62**
CS-CM4-IGR	23	0.0213	98.84	1.28	6630	63	19	0.0182	99.03	1.15	5491	63
CS-CM4-IGR	30	0.0178	98.95	1.046	9859	63	34	0.0164	99.05	0.93	9890	62
CS-CM4-CFS	15	0.0232	98.92	1.388	**3620**	63	17	0.0221	98.68	1.317	4337	63
CS-CM4-CFS	29	0.0187	98.93	1.064	8596	**47**	31	0.0181	98.96	1.033	10483	**62**
C4.5-All	41	**0.0124**	**99.45**	**0.545**	16146	62	41	**0.0124**	**99.45**	**0.545**	16146	**62**
CS-CM4-All	41	0.0172	98.94	1.05	14867	62	41	0.0172	98.94	1.05	14867	**62**

Table 4.9: Comparison of evaluation metrics used for CS-CM4 and C4.5 Classifiers

Category	Evaluation Metrics	C4.5[77]	KDD winner[16]	LAMSTAR[99]	Multi-classifier[100]	J48[7]	CS-CM4[19]	Best result from Proposed approach	
								C4.5	CS-CM4
Normal	DR	98.1	99.5	99.7	-	-	98.7	99.5	99.1
PROBE	DR	88.5	83.3	98.9	88.7	99.2	98.8	99.4	99.2
DOS	DR	97.1	97.1	99.2	97.3	99.9	99.9	99.8	99.6
U2R	DR	16.2	13.2	30.3	29.8	50.0	72.5	**91.5**	**89.4**
R2L	DR	3.4	8.4	41.2	9.6	92.9	91.7	**94.0**	**93.4**
	OCA	92.23	92.71	-	-	99.7	98.9	**99.45**	**99.05**
	AMC	0.2426	0.2331	0.0130	0.2285	-	0.0199	**0.0124**	**0.0164**
	FPR	-	0.55	5.58	-	-	1.3	**0.545**	**0.930**
	TTBM(ms)	-	-	112000	-	41975	-	16146	9890

Table 4.10: Performance comparison with recently proposed classifiers in the literature

Note that in table 4.10 they are provided for reference instead of direct comparison, as their sampling on training and test data are different from one another except for [19] applies NSL-KDD data set for both training and test that enable us compare results directly. The hyphen marks indicate that results are not reported in the papers.

4.6.2.1. Performance for cost sensitive approach

A comparison of IGR and CFS algorithms shows that the AMC achieved with IGR subsets is better than CFS results (see table 4.9) with the same classification accuracy for C4.5 classifier and it also outperform its counterpart CS-CM4 in all cases. When looking at the size of the features it is also comparable. CFS as shown in table 4.9 achieved the same result using 17 and 31 features except the difference in the training time. The main point here both selection methods identify the smallest number of features with

comparable result. For CS-CM4 classifier IGR-19 and CFS-31 are comparable with feature space reduction of 12 for IGR and 10 for CFS.

Generally it is possible to say for the cost sensitive feature selection approach IGR fits well in all cases. Among the feature selection methods, CFS tends to produce the smallest feature subsets with very competitive performance. The table also shows how often each method performs significantly better than performing with no feature selection. Comparing results with the works of [16], [99], [100], [7], [19]; it can be seen that the proposed approach clearly outperforms all result.

4.6.2.2. Performance for cost insensitive approach

In table 4.9, it can be observed that C4.5 outperforms CS-CM4 in all cases. When feature selection is applied, similarly C4.5 performs better in all metrics using CFS-29 features. In the same way CS-CM4 achieved better result when IGR is applied using 30 features. Compared to all features the cost insensitive feature selection approach for C4.5 classifier performs less in all metrics except a training time reduction 7550 ms. On the other hand CS-CM4 using 30 features is comparable to using 41 features with reduction of 5008 ms training time. In this approach C4.5-CFS combination outperform C4.5-IGR. Similarly CS-CM4-IGR outperforms CS-CM4-CFS. The average misclassification cost, overall classification accuracy and false alarm rate are slightly different from the ones obtained by using all features. The absolute difference between them does not overcome 0.0019%, 0.07% and 0.71% for C4.5 and 0.0006%, 0.01% and 0.004% for CS-CM4 respectively.

Generally possible to say, for cost insensitive feature selection approach CFS fits the metacost C4.5 learner and IGR fits direct cost sensitive learner CS-CM4. Rather than using all features better to use feature selection algorithms with comparable result having a reduction in training time. Even though, the gain in all metrics is not very high compared to all features, the overall gain of the feature selection classification procedure lies in a significantly improved training time and in obtaining the classification results due to reduced number of relevant features. Therefore, based on all these experiments we can say that our new method outperforms [19] methods by removing 7 features and even

getting better performances. Thus it can be used to find optimal subsets of relevant features by means of the IGR measure instead of IG for intrusion detection systems.

4.6.2.3. Performance comparison of cost (in) sensitive approach

For both classifiers in the two approaches IGR achieved better result. From the two approaches in all cases cost sensitive outperform the insensitive using IGR compared to CFS. But CFS when it uses 29 features for cost insensitive and 31 cost sensitive resulted comparable result and even have the same overall classification accuracy. CFS feature selection for CS-CM4 classifier, the cost insensitive approach achieved better OCA using 15 features than 17. Here to conclude its performance is difficult because OCA as the only measure is not appropriate where there is class imbalance to which the minority class contributes very little.

The experiments tested on AMC, OCA, FPR and TTBM among 41 features, C4.5 reduces feature to 19 and achieved comparable result on the proposed approach (cost sensitive). There is also a great difference in the training times that enhances the learning capabilities and reduces the computational intensity. In the same way CS-CM4 reduces features to 34 and performs better than using all features in all cases. Even though, the focus of this study is not about feature reduction rather detecting minority class attacks and increase performance of IDS for those attacks. The features selected by IGR, when applied to the classifiers give higher performance than the features selected by CFS in both approaches, whereas the training time and testing time are significantly reduced for CFS feature selection algorithm.

Generally the proposed approach results are by far better than the results found in [19] and others in both classifiers as shown in table 4.10. There is an increase of 0.0075%, 0.55 % and 75.5% for AMC, OCA and FPR respectively on the proposed approach (Cost sensitive feature selection).

4.7. Result Validation

A straightforward way for result validation is to directly measure the result using prior knowledge about the data if, we know the relevant features beforehand. In real-world applications, however, such prior knowledge usually is not available. Hence, have to rely on some indirect methods by monitoring the change of mining performance with the change of features. For example, if classification accuracy is used as a performance indicator for a mining task, for a selected feature subset, it is simply possible to conduct the "before-and-after" experiment to compare the classification accuracy of the classifier learned on the full set of features and that learned on the selected subset [101],[17].

Analogously, we also used this validation technique before and after feature selection in both approaches. The result of before and after are indicated in the summary table 4.7 and 4.8 and comparison discussions are also included in each cases. Generally, C4.5 has better performance when no feature selection performed compared to CS-CM4. In contrast, CS-CM4 when feature selection performed in all cases. This may result from the way they incorporate the cost in their model building phase.

4.8. Findings and Challenges

This work result both findings and challenges. The findings can be considered as the achievements. The following are the findings that have been found in this research:

First, the best feature selection techniques in selecting informative attributes (features) for minority class members can be achieved as metacost sensitive feature selection. Both IGR and CFS algorithms are found to be the best technique in selecting based detection accuracy, average misclassification cost, overall classification accuracy, false alarm rate and training time as evident from the experiment. The cost given to each class category based on the impact on the misclassification by giving more weight to the minority class which does not have frequent pattern records.

Second, the downside of information gain value in feature selection for the minority class is proven to be true. This is proved by using IGR (modification of IG, not to bias classes

with more values) by applying cost insensitive feature selection approach shown in table 4.10 as a comparison with the work of [19].

Third, CFS which evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them; the experimental results of the algorithm illustrates the proposed approach have improved the CS-CM4 and C4.5 performance with great feature space and training time reduction by gaining more weight using the cost metrics.

Finally, the metacost sensitive classifier C4.5 in both approaches has better performance than the direct cost sensitive CS-CM4 classifier. It is possible to say inducing costs by modifying the algorithm is not as effective as using cost during model building without modifying the learning algorithm. This is because, if the cost of different types of error (false positive and false negative is not the same, it is usually more expensive to misclassify a minority (positive) example into the majority (negative) class, than a majority example into the minority class. This is effectively dealt with indirect cost sensitive learning methods than metacost as stated in [59].

At the same time challenges are encountered during the research process. The first is, deciding the threshold value. The algorithm IGR produces rank of features with their average merit, however, difficult to point out any pattern in their contribution to the classifier to discover unfrequent records. The second, in CFS algorithm no indicator is found to what extent the feature and class are correlated even though possible to estimate the relation between features to feature when there is redundancy among them by having similar value. So difficult to judge when a minority class gets better performance unless gross performance is taken. Lastly, selecting algorithms that best fit for the approach from the many is tiresome. Algorithms fit for one attack may not work for the other that requires testing many algorithms repeatedly. In some cases the selected algorithms perform 99% and other cases 68% that need some combinations of cost sensitive with insensitive to select features and develop models to have an optimal performance in all metrics used in this research.

CHAPTER FIVE

5. CONCLUSIONS AND RECOMMENDATIONS

This final chapter discusses the overall process of the research, the achievements that have been obtained and the conclusions are pointed out. To wrap up the study appropriate suggestions are proposed to future investigate and overcome the problems.

5.1. Conclusions

Data mining to identify intrusive and normal traffic data for intrusion detection has been a fertile field of research. However data mining needs methods to preprocess the data and one of these methods is feature selection which is a growing field of interest about selecting proper features from many features. This is because it is expensive to carry out the entire process and degrades the classification performance of data mining algorithms. Therefore, feature selection approaches reduce the complexity of the overall process by allowing the data mining system to focus on what is really important.

The central claim of this thesis is that cost sensitive feature selection for cost sensitive supervised machine learning decision tree algorithms can be accomplished on the basis of information gain ration and correlation between features and class to detect attacks of both majority and minority class. Metacost sensitive feature selection algorithms are implemented and empirically tested to support this claim. Thus, the performance of the IDS is found to be more efficient and effective that experts get better results quickly in protecting damages due to intrusion attacks.

In this study feature selection and classification are framed jointly as a cost sensitive approach. In most current IDSs, all data features to detect intrusion or misuse patterns consider the importance of each feature equal for both the majority and minority classes. In reality some of the features may be redundant or contribute little to the detection process and even may degrade the performance of the systems. The purpose of this thesis

is to identify important input features in building IDS that is computationally efficient and effective and lower the misclassification cost by considering all misclassification costs are not equal. It is also expected to lower false positive rate and misclassification cost and increase the classification accuracy. In regard to this the main and special emphasis on cost sensitive learning is to detect the dangerous attacks of minority class U2R and R2L.

As discussed in chapter four, classification process are done using two types of classifiers which are CS-CM4 and C4.5 (as metacost) and in both algorithms obtained a better result during feature selection; even though comparable results are achieved using all (41) features for C4.5 decision tree classifier. Thus, the classifications are carried out on top ranked features which are selected by IGR and CFS feature selection algorithms by making them cost sensitive. In this research new techniques are investigated on feature selection for intrusion detection. The feature selection approaches in both algorithms cost sensitive and insensitive are employed and their performance on the benchmark intrusion dataset NSL-KDD is performed and experimental results are analyzed.

Feature selection approaches like cost sensitive ranking and selection using IGR and CFS are applied to the NSL-KDD dataset to identify important features. It is found that IGR and CFS selects a minimum of 19 and 15 features respectively. The identified features are used as input to the two classifiers and the results are compared. The results show that the performance with 19 and 15 features are comparable to the 41 features and even better in performance with reduced training and testing times. Comparing the two algorithms, features identified by IGR gave better detection accuracy, False Positive Rate, AMC, training and testing times than CFS.

Comparing the two classifiers used the C4.5 (metacost) shows better performance for all the classes with comparable training and testing times when 41 features selected by IGR are used as input dataset. The features obtained with IGR shows promising results, although there may be many good approaches for selecting a feature subset, it appears to be the best feature selection method, or a globally optimal feature subset evaluator.

Following this direct cost sensitive CS-CM4 and C4.5 by making indirect (metacost) sensitive classifiers as an intrusion detection models are explored. Performance comparisons are also demonstrated using the different reduced and all features. Features which are identified by the proposed cost sensitive and insensitive approach are experimented.

The experimental results show that the proposed feature selection approaches and models gave better and robust result as it is able to reduce features achieving in a 22% feature space reduction and approximately 94.6% time reduction in training with almost same accuracy achieved in detecting new attacks. The proposed approach especially in detecting dangerous attacks U2R (89.4%, 91.5%) and R2L (93.4%, 94.0%) CS-CM4 and C4.5 achieved robust result respectively. This shows that our proposed algorithms with their approach are comparably reliable in intrusion detection. The CS-CM4 and C4.5 decision tree provided good overall classification accuracy equals to 99.05%, 99.45%; false positive rates 0.930%, 0.545%, and average misclassification cost 0.0164, 0.0124 respectively. The proposed models are better than many other approaches [16], [99], [100], [7], [19] as presented in table 4.10.

In addition it is possible to say no single learning algorithm is superior to all others for all attack types. This leads researchers in machine learning to attempt provide insight into the strengths and limitations of different algorithms so that practitioners can choose which algorithms to apply for the problems encountered.

Finally, the goal of the classification is to distinguish between normal and attack records for IDS. There are various classifiers whether that consider cost of classification or recently developed get attention which considers cost of misclassification that has been studied to classify the intrusions. However, not all the classifiers are evaluated on suitable metric. It is hopeful that this study can give valuable information for the IDS researches using cost sensitive feature evaluation in order to reduce damage caused by intruders by identifying features that contribute much to the classifier.

5.2. Recommendations

There are various techniques of feature selection that can be used to solve the arising problem in features that have large number in order to reduce number of features and improve the overall performance of the classification. By using feature selection, only the appropriate subset of features for intrusion detection can be selected among the many. There are three approaches of feature selection: that include filter, wrapper and embedded methods. Based on our proposed research, filter method is the best and commonly used among other methods. Still there are many interesting problems related to this research open for future investigation and is necessary to overcome the problem arises in the study in order to improve this research. The following are list of some possible directions:

- More investigation on using direct cost sensitive feature ranking and selection algorithms: On this study indirect cost sensitive feature selection is used; better also empirically compare the result with direct cost sensitive feature selection approach that incorporate costs by modifying the algorithm to this work on the same dataset.

- More investigation in non-decision tree classification algorithms: In further work it is possible to check with cost sensitive non-decision tree algorithms listed and described in the literature to measure in terms of all metrics and see the performance gap.

- More investigations on other base feature selection algorithms: In this thesis, two kinds of base feature selection algorithms from the family of filter method, IGR and CFS, are investigated for ranking and selecting features using the proposed cost sensitive approach. The approach is applicable to any base feature evaluation algorithms found in WEKA. There are other feature evaluation methods, such as Wrapper and Embedded all of them available in WEKA and need to be explored. As the approach identified differently with respect to the base evaluator, further

study can investigate how this cost sensitive approach identify the minority class features as well.

- More investigations on other application domains: Several application domains where the unequal misclassification cost problem prevails are listed in the literature. In our experimental study on classifying the intrusion dataset, data taken from the MIT machine learning simulated repository was used. The proposed cost sensitive algorithms can also be applied to other application domains, such as fraud detection and medical diagnosis, to explore their effectiveness in these specific domains. In addition more investigation can be done on cost sensitive feature selection algorithms for real world intrusion detection dataset.

- Finally, more investigation to develop and test adaptive cost sensitive feature selection: In dynamic and complex environment for network intrusion detection, abundance and rare records which creates class imbalance, adaptive cost sensitive feature selection algorithms are required. The algorithms should perform at a lower level of misclassification cost, balanced detection accuracy and that keeps false positives at acceptable level for majority and minority network attack classes.

Misclassification costs and cost of feature selection have potential applications where there is exists class imbalance in a dataset. In real world, unequal cost of misclassification is abundant. Therefore, it will continue to receive more and more attention in both the scientific and the industrial world.

REFERENCES

[1]. Mrutyunjaya P. and Manas R. (2009). A Novel Classification Via Clustering Method for Anomaly Based Network Intrusion Detection System. International Journal of Recent Trends in Engineering, Vol 2, No. 1, PP. 1-6.

[2]. Yongro P. (2005). A Statistical Process Control Approach for Network Intrusion Detection. PhD Dissertation, Georgia Institute of Technology, USA.

[3]. Shipley G. (2001).Intrusion Detection Systems (IDS). In Shelley Johnston Markunday (Ed.), Maximum Security: A Hacker's Guide to Protecting Your Internet Site and Network, Third edition, Sams Publication.

[4]. Alireza O. and Bita S. (2008). Intrusion Detection in Computer Networks based on Machine Learning Algorithms**.** IJCSNS International Journal of Computer Science and Network Security, Vol. 8, No.11, PP. 15-23.

[5]. Meera G. Gandhi and Srivatsa S. (2010). Classification Algorithms in Comparing Classifier Categories to Predict the Accuracy of the Network Intrusion Detection: A Machine Learning Approach. Advances in Computational Sciences and Technology, Vol. 3, PP. 321–334.

[6]. Han W. and Kamber M. (2006). Data Mining: Concepts and Techniques. Second edition, Morgan Kaufmann Publishers, Massachusetts.

[7]. Meera G., Gandhi and Srivatsa S. (2010). Adaptive Machine Learning Algorithm (AMLA) Using J48 Classifier for an NIDS Environment. Advances in Computational Sciences and Technology, Vol. 3, PP. 291–304.

[8]. Saher E. and Shaul M. (2008). Anytime Induction of Low-cost, Low-error Classifiers: A Sampling-based Approach. Journal of Artificial Intelligence Research, Vol. 33, No. 5, PP. 1-33.

[9]. Kehan G., Taghi K. and Jason V. (2010). An Evaluation of Sampling on Filter-Based Feature Selection Methods. Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference, PP. 416-421.

[10]. Kok-Chin K., Choo-Yee T., Somnuk-Phon A. (2009). A Feature Selection Approach for Network Intrusion Detection. International Conference on Information Management and Engineering, IEEE computer Society, K.L., PP.133-137.

[11]. Shichao Z., Li L., Xiaofeng Z. and Chen Z. (2008). A Strategy for Attributes Selection in Cost-Sensitive Decision Trees Induction. IEEE 8th International Conference on Computer and Information Technology Workshops, PP. 1-13.

[12]. Kabiri P. and Ghorbani A. (2005). Research on Intrusion Detection and Response: A Survey. International Journal of Network Security, Vol. 1, No. 2, PP. 84-102.

[13]. Shannon C. (2001).A Mathematical Theory of Communication. SIGMOBILE Computational Communication Review. Vol. 5, No. 1, PP. 3–55.

[14]. Yiming Y. and Jan P. (1997). A Comparative Study on Feature Selection in Text Categorization. In ICML ’97: Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc, PP. 412-420.

[15]. Domingos P. (1999). MetaCost: A general method for Making Classifiers Cost-sensitive. In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, ACM Press, PP. 155-164.

[16]. Charles E. (2001). The foundations of cost-sensitive learning. In Proceedings of the Seventeenth International Joint Conference of Artificial Intelligence. Morgan Kaufmann, Seattle, Washington, PP. 973–978.

[17]. Yael W., Yuval F., Yuval E. and Lior R. (2010). Cost-Sensitive Detection of Malicious Applications in Mobile Devices. Duetsche Telekom Laboratories at Ben-Gurion University, Israel. Available at: http://wnss.sv.cmu.edu/wms/wmstalk01.pdf.[accessed:March 2, 2011].

[18]. Marko R. (2003). Experiments with Cost sensitive Feature Evaluation. In Lavra·c et al. (Eds): Machine Learning, Proceedings of ECML 2003, Springer, Berlin, PP. 325-336.

[19]. Adamu T. (2010). Computer Network Intrusion Detection: Machine Learning Approach. MSC Thesis, AAU, Addis Ababa, Ethiopia.

[20]. Mike W. and Xue-wen C. (2010). Combating the Small Sample Class Imbalance Problem Using Feature Selection. IEEE Transactions on Knowledge and Data Engineering, Vol. 22, No. 10, PP. 1388-1399.

[21]. NSL-KDD Data set for Network-based Intrusion Detection Systems. Available at: http://nsl.cs.unb.ca/NSL-KDD/ [accessed: December 15, 2010].

[22]. Mahbod T., Ebrahim B., Wei L., and Ali A. (2009). A Detailed Analysis of the KDD CUP 99 Data Set. Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications. National Research Council, Canada, PP. 1-6.

[23]. Weka 3.6.4: Data Mining Software in Java. Available at: http://www.cs.waikato.ac.nz/ml/weka/ [accessed: February 01, 2011].

[24]. Lippmann R., Fried D., Graf I., Haines J., Kendall K., McClung D., Weber D., Webster S., Wyschogrod D., Cunningham R. and Zissman M. (2000). Evaluating Intrusion Detection Systems: The 1998 darpa off-line intrusion detection evaluation, Vol. 2, No.3, PP. 1-26.

[25]. Ranum M. (2001). Experiences Benchmarking Intrusion Detection Systems. NFR Security Technical Publication. Available at: http:// www.nfr.com

66.14.166.45/whitepapers/compforensics/ids/Benchmarking%20IDS.pdf[accessed: March 10, 2011].

[26]. Lippmann R., Cunningham K., Fried J., Graf I., Kendall K., Webster S., Zissman M. (1999). Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation, Proceedings of the Second International Workshop on Recent Advances in Intrusion Detection (RAID99), West Lafayette, PP. 1-22.

[27]. Bay S. and Pazzani M. (2000). Characterizing Model Performance in the Feature Space. In ICML 2000 Workshop on What Works Well Where. Proceedings of the 17th International Conference. Morgan Kaufmann, PP. 98-115

[28]. Garfinkel S. and Spafford G. (1996). Practical UNIX and Internet Security. Second Edition. O'Reillly and Association, Inc. Sebastopol, Tokiyo, Japan.

[29]. Russell D. and Gangemi G. (1991). Computer Security Basics. O'Reilly & Associates, Inc., California, USA.

[30]. Anders B. and Nils P. (2004). A Quantitative Evaluation Framework for Component Security in Distributed Information Systems. Institute of Technology Linpionking University.

[31]. Pfleeger C. and Pfleeger S. (2003). Security in computing. Prentice Hall.

[32]. Smaha and Stephen E. (1988). An Intrusion Detection System. In Fourth Aerospace Computer Security Applications Conference, Tractor Applied Science., Texas, PP. 37-44.

[33]. Proctor P. (2001). The practical Intrusion Detection Handbook. Prentice Hall

[34]. Stefan A. (1998). Research in Intrusion-Detection Systems: A Survey. Technical Report 98-17. Department of Computer Engineering, Chalmers University of Technology.

[35]. Stefan A. (2000). Intrusion Detection Systems: A Taxonomy and Survey. Technical Report 99-15, Department of Computer Engineering, Chalmers University of Technology.

[36]. Anita K., Jones and Robert S. (1999). Computer System Intrusion Detection: A Survey. Technical report, Department of Computer Science, University of Virginia. Available at: http://www.cs.virginia.edu/˜jones/ IDS [accessed: January 23, 2011].

[37]. Peyman K. and Ali G. (2005). Research on Intrusion Detection and Response: A Survey. International Journal of Network Security, Vol.1, No.4, PP. 84-102.

[38]. Kapil K., Baikunth N., Kotagiri R. and Ashraf K. (2006). Attacking Confidentiality: An Agent Based Approach. In Proceedings of IEEE International Conference on Intelligence and Security Informatics, Vol. 397, PP. 285-296.

[39]. Cho S. (2002). Incorporating Soft Computing Techniques into A probabilistic Intrusion Detection System. IEEE Transactions on Systems, Man and Sybernetics, Vol. 32, No. 2, PP. 154-160.

[40]. Abraham R. Jain and Thomas J. (2007). D-SCIDS: Distributed soft computing Intrusion Detection System. Journal of Network and Computer Applications, Vol. 30, No. 4, PP. 81-98.

[41]. Howard J. (1997). An Analysis of Security Incidents on the Internet 1989 - 1995, Dissertation. Carnegie Mellon University, Pittsburgh, Pennsylvania.

[42]. Ertoz E. and Lazareviv A. (2004). The MINDS: Minnesota intrusion detection system. Next generation data mining, MIT Press.

[43]. Chang-Tien L., Arnold P. and Prajwal M. (2005). Exploiting Efficient Data Mining Techniques to Enhance Intrusion Detection Systems. Department of Computer Science Virginia Polytechnic Institute and State University. IEEE, PP. 512-517.

[44]. Witten I. and Frank E. (2005). Data Mining: Practical Machine Learning Tools and Techniques. Second editions, Morgan Kaufmann, Massachusetts.

[45]. Jose F. (2009). Data clustering for anomaly detection in Network Intrusion detection. Research Alliance in Math and Science, PP. 1-12.

[46]. Meng J. (2009). The Application on Intrusion Detection Based on K-Means Cluster Algorithm. International Forum on Information Technology and Application, No.15-17, PP. 150-152.

[47]. Hendrickx I. and Vanden B. (2005). Hybrid Algorithms with Instance-based Classification. ILK, Tilburg University, PP. 158-169.

[48]. Quinlan J. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA, Massachusetts.

[49]. Quinlan J. (1986). Induction of Decision Trees. Journal of Machine Learning, Vol. 1, PP. 81-106.

[50]. Breiman L., Freidman R. Olshen and Stone C. (1984). Classification and Regression Trees. New edition , Hapman and Hall/CRC, Wadsworth Belmont.

[51]. http://eric.univ-lyon2.fr/~ricco/tanagra/en/tanagra.html[accessed: January 10, 2011].

[52]. Tsoukalas L. and Uhrig U. (1997). Fuzzy and Neural Approaches in Engineering. John Wiley and Sons, New York, NY.

[53]. Pearl J. (1988). Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, Massachusetts.

[54]. Pfleeger C. and Pfleeger S. (2003). Security in computing. Prentice Hall.

[55]. Hipp U. and Nakhaeizadeh G. (2000). Algorithms for Association Rule Mining: A General Survey and Comparison. Journal machine learning, Vol. 2, No.1, PP. 58-64.

[56].KDDCup1999.Availableat:http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [accessed: January 29, 2011].

[57]. Kayacik H., Zincir-Heywood A. and Heywood M. (2003). On the Capability of an SOM Based Intrusion Detection System. In Proc. Int. Joint Conf. Neural Network Jul, Vol. 3, PP. 1808-1813.

[58]. Turney P. (2000). 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, PP. 15-21.

[59]. Victor S. and Charles L. (2007). Roulette Sampling for Cost-Sensitive Learning. ECML 2007, PP. 724-731.

[60]. Sheng V. and Charles L. (2006). Thresholding for Making Classifiers Cost Sensitive. Aaai Conference on Artificial Intelligence archive, American Association for Artificial Intelligence, Proceedings of the 21st national conference, PP. 476-481.

[61]. Zadrozny B., Langford J. and Abe N. (2003). Cost-sensitive learning by cost proportionate instance weighting. In Proceedings of the 3th International Conference on Data Mining, PP. 1-8.

[62]. Matthew M., Sal S., Kahil J., Christopher P., Erez Z. and Vijay P. (2002). Toward Cost-Sensitive Modeling for Intrusion Detection and Response. Journal of Computer Security, North Carolina State University, Vol. 10, PP. 1-22.

[63]. Hild E., Erdogmus J. (2001). Principe, Blind Source Separation Using Renyi's Mutual Information. IEEE Signal Processing Letters, Vol. 8, PP. 174-176.

[64]. Das S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. In Proceedings of the Eighteenth International Conference on Machine Learning, PP. 74 -81.

[65]. Guyon I. and Elisseeff A. (2003). An Introduction to Variable and Feature Selection. Journal of Machine Learning Research Vol. 3, PP. 1157-1182.

[66]. Liu H. and Yu L. (2005). Toward Integrating Feature Selection Algorithms for Classification and Clustering. IEEE Transactions on Knowledge and Data Engineering, Vol.17, No. 4, PP. 491-502.

[67]. Vidal-Naquet M. and Ullman S. (2003). Object recognition with informative features and linear classification. IEEE Conference on Computer Vision and Pattern Recognition, PP. 112-145.

[68]. Hall M. (1999). Correlation-based Feature Subset Selection for Machine Learning. PhD, Dissertation, Department of Computer Science, University of Waikato, Hamilton, New Zealand.

[69]. Gulshan K., Krishan K. and Monika S. (2010). An Empirical Comparative Analysis of Feature Reduction Methods for Intrusion Detection. International Journal of Information and Telecommunication Technology, Vol.1, PP. 44-51.

[70]. Alexander H. (2004). Feature Selection for Intrusion Detection: An Evolutionary Wrapper Approach. Institute for Computer Architectures, IEEE, PP. 1563-1568.

[71]. Kohavi R. and John G. (1997). Wrappers for Feature Subset Selection. Artificial Intelligence, Vol. 1, PP. 273–324.

[72]. Jain A. and Zongker D. (1997). Feature Selection: Evaluation, Application, and Small Sample Performance. IEEE Trans Pattern Analysis and Machine Intelligence, Vol. 19, PP. 153-158.

[73]. Jain A., Duin R. and Mao J. (2000). Statistical Pattern Recognition: A Review, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.22, PP. 4-37.

[74]. Dash M. and Liu H. (2003). Consistency-based Search in Feature Selection. Artificial Intelligence Vol. 151, PP. 155-176.

[75]. Kira K. and Rendell L. (1992). A Practical Approach to Feature Selection. Proceedings of the 9th International Conference on Machine Learning, Morgan Kaufmann, Los Altos, CA, PP. 249-256.

[76]. Kononenko I. (1994). Estimating attributes: Analysis and extension of RELIEF. Proceedings of European Conference on Machine Learning, Catania, Italy, PP. 171–182.

[77]. Bell D. and Wang H. (2000). A formalism for Relevance and its Application in Feature Subset Selection. Machine Learning, Vol. 41, PP. 175-195.

[78]. Sheinvald D. and Niblack W. (1990). A Modeling Approach to Feature Selection. Proceedings of Tenth International Conference on Pattern Recognition, Atlantic City, NJ, Vol. 1, PP. 535-539.

[79]. Modrzejewski M. (1993). Feature Selection Using Rough Sets Theory. Proceedings of the European Conference on Machine Learning, Vienna, Austria. PP. 213-226.

[80]. Almuallim H. and Dietterich T. (1994). Learning Boolean Concepts in the Presence of Many Irrelevant Features. Artificial Intelligence, Vol. 2, PP. 279-305.

[81]. Blum A. and Langley P. (1997). Selection of Relevant Features and Examples in Machine Learning. Artificial Intelligence, Vol. 97, No. 1-2, PP. 245-271.

[82]. Miller A. (1990). Subset Selection in Regression. Chapman and Hall, New York.

[83]. Pearl J. (1984). Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, USA.

[84]. Nigel W., Sebastian Z. and Grenville A.(2006). Evaluating Machine Learning Algorithms for Automated Network Application Identification. Centre for Advanced Internet Architectures (CAIA).Technical Report Swinburne University ofTechnologyMelbourne.Avaliableat:citeseer.ist.psu.edu/viewdoc/[accessed: march 15, 2011].

[85]. Yang J. and Honavar V. (1998). Feature Subset Selection Using a Genetic Algorithm. IEEE Intelligent Systems (Special Issue on Feature Transformation and Subset Selection), Vol. 13, PP. 44-49.

[86]. Sikonja M. and Kononenko I. (2003). Theoretical and Empirical Analysis of Relief and ReliefF. Machine Learning journal, PP. 1-52.

[87]. Song A., Smola A. Gretton K. and Bedo J. (2007). Supervised Feature Selection Via Dependence Estimation. International Conference on Machine Learning, proceedings of the 24th international conference, New York, NY, USA, PP. 55-110.

[88]. Hall M. (2000). Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, PP. 359-366.

[89]. Gamberger D. and Lavrac N. (1997). Conditions for Occam's razor Applicability and Noise Elimination. Proceedings of the Ninth European Conference on Machine Learning. Berlin, PP. 108-123.

[90]. Fayyad U., Piatetsky-Shapiro G., and Smyth P. (1996). The KDD process for Extracting Useful Knowledge from Volumes of Data. Communications of the ACM, Vol. 39, PP. 27-34.

[91]. Marko R. (2003). Experiments with Cost Sensitive Feature Evaluation. In Lavra·c et al. (Eds): Machine Learning, Proceedings of ECML 2003, Springer, Berlin, PP. 325-336.

[92]. Amita S. and Aruna T. (2010). Network Traffic Classification Using Semi-Supervised Approach. Proceedings of the 2010 Second International Conference

on Machine Learning and Computing, IEEE Computer Society Washington, DC, USA ,PP. 345-349.

[93]. Mrutyunjaya P. (2009). Evaluating Machine Learning Algorithms for Detecting Network Intrusions. International Journal of Recent Trends in Engineering, Vol. 1, No. 1. PP. 472-477.

[94]. Farhan A., Zulkhairi M., Dahalin and Shaidah J. (2010). Distributed and Cooperative Hierarchical Intrusion Detection on MANETs. International Journal of Computer Applications, Vol. 12, No.5, PP. 33-40.

[95]. Ferri C., Flach P. and Henrandez-Orallo J. (2002). Learning Decision Trees Using Area under the ROC Curve. Proceedings of the 19th International Conference on Machine Learning, Morgan Kaufmann, PP. 139-146.

[96]. Sung A. (1998). Ranking Importance of Input Parameters of Neural Networks. Journal of Expert Systems with Applications, Vol. 15, PP. 405-41.

[97]. Cost Sensitive Attribute Evaluation and Selection. Available at: http://sourceforge.jp/projects/sfnet_weka/ [accessed: March 10, 2011].

[98]. Chi-Ho T., Sam K. and HanliWang (2007). Genetic-fuzzy Rule Mining Approach and Evaluation of Feature Selection Techniques for Anomaly Intrusion Detection. The Journal of The pattern Recognition Society, PP. 2374-2390.

[99]. Venkatachalam V. and Selvan S. (2007). Performance Comparison of Intrusion Detection System Classifiers Using Various Feature Reduction Techniques. International Journal of Simulation, Vol. 9, No 1, PP. 30-38.

[100]. Maheshkumar S., Gursel S. (2003). Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection context. In Proceedings of International Conference on Machine Learning, Models, Technologies and Applications, Las Vegas, Nevadat, USA, CSREA Press, PP. 209–215.

[101]. Liu H. and Motoda H. (1998). Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers, Boston.

Appendix A

The three categories of intrusion features

Basic features of individual TCP connection		Description	Type
Label	**Network Data Features**		
1	Duration	length (no. of seconds) of the connection	continuous
2	protocol_type	type of the protocol	continuous
3	Service	network service on the destination	discrete
4	flag	status flag of the connection	discrete
5	src_bytes	no. of data bytes from source to destination	continuous
6	dst_bytes	no. of data bytes from destination to source	continuous
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong fragment	no. of wrong fragments	continuous
9	Urgent	no. of urgent packets	continuous
Content features within a connection suggested by domain knowledge			
10	Hot	no. of "hot" indicators	continuous
11	num_failed_logins	logins no. of failed logins	continuous
12	logged_in	1 if successfully logged in; 0 otherwise	discrete
13	num_compromised	no. of "compromised" conditions	continuous
14	root_shell	1 if root shell is obtained; 0 otherwise	discrete
15	su_attempted	1 if "su root" command attempted; 0 otherwise	discrete
16	num_root	no. of "root" accesses	continuous
17	num_file_creations	no. of file creation operations	continuous
18	num_shells	no. of shell prompts	continuous
19	num_access_files	no. of operations on access control files	continuous
20	num_outbounds_cmds	no. of outbound commands in an ftp session	continuous
21	is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	discrete
22	is_guest_login	1 if the login is a "guest" login; 0 otherwise	discrete
Traffic features computed using a two-second			
23	Count	no. of connections to the same host as the current connection in the past two seconds	continuous
24	sev_count	no. of connections to the same service as the current connection in the past two seconds	continuous
25	serror_rate	% of connections that have "SYN" errors	continuous
26	sev_serror_rate	% of connections that have "SYN" errors	continuous
27	rerror_rate	% of connections that have "REJ" errors	continuous
28	srv_rerror_rate	% of connections that have "REJ" errors	continuous
29	same_srv_rate	% of connections to the same service	continuous
30	diff_srv_rate	% of connections to different services	continuous
31	srv_diff_host_rate	% of connections to different hosts	continuous
32	Dst_host_count	count of connections having the same destination host	continuous
33	Dst_host_srv_count	count of connections having the same destination host and using the same service	continuous
34	Dst_host_same_srv_rate	% of connections having the same destination host and using the same service	continuous
35	Dst_host_diff_srv_rate	% of different services on the current host	continuous
36	Dst_host_same_src_port_rate	rate % of connections to the current host having the same src port	continuous
37	Dst_host_srv_diff_host_rate	% of connections to the same service coming from different hosts	continuous
38	Dst_host_server_rate	% of connections to the current host that have an S0 error	continuous
39	Dst_host_srv_serror_rate	% of connections to the current host and specified service that have an S0 error	continuous
40	Dst_host_rerror_rate	% of connections to the current host that have an RST	continuous
41	Dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST error	continuous

Appendix B

Header Attribute and data declaration

Full attribute declaration

@relation 'NSL-KDD'

@attribute 'status' {'train', 'test'}

@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', 'PROBE' 'DOS', 'U2R', 'R2L'}

Sample data declaration

@data

train,0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,150,25,0.17,0.03,0.17,0,0,0,0.05,0,NORMAL

train,0,udp,other,SF,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,1,0,0,0,0,0.08,0.15,0,255,1,0,0.6,0.88,0,0,0,0,0,NORMAL

test,0,tcp,http,SF,235,342,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,18,18,0,0,0,0,1,0,0,18,255,1,0,0.06,0.02,0,0,0,0,NORMAL

test,0,udp,domain_u,SF,44,44,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,229,229,0,0,0,0,1,0,0,255,254,1,0.01,0,0,0,0,0,0,NORMAL

Appendix C

Feature evaluation and ranking

Cost sensitive IGR-RankerSearch Result

=== Run information ===

Evaluator: weka.attributeSelection.CostSensitiveAttributeEval -cost-matrix "[0.0 1.0 2.0 2.0 2.0; 1.0 0.0 2.0 2.0 2.0; 2.0 1.0 0.0 2.0 2.0; 3.0 2.0 2.0 0.0 2.0; 4.0 2.0 2.0 2.0 0.0]" -S 1 -W weka.attributeSelection.GainRatioAttributeEval --

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

Relation: NSL-KDDtrain

Instances: 125973

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
0.51 +- 0.005	1 +- 0	26 srv_serror_rate
0.462 +- 0.001	2.4 +- 0.49	4 flag
0.462 +- 0.002	2.7 +- 0.64	25 serror_rate
0.457 +- 0.002	3.9 +- 0.3	12 logged_in
0.442 +- 0.002	5 +- 0	39 dst_host_srv_serror_rate
0.373 +- 0.002	6.5 +- 0.5	38 dst_host_serror_rate
0.362 +- 0.017	6.5 +- 0.5	30 diff_srv_rate
0.288 +- 0.003	8.5 +- 0.5	5 src_bytes
0.281 +- 0.013	9.2 +- 0.98	6 dst_bytes
0.27 +- 0.014	10.1 +- 0.83	29 same_srv_rate
0.257 +- 0.017	11.1 +- 1.3	11 num_failed_logins
0.242 +- 0.006	11.9 +- 0.7	22 is_guest_login
0.224 +- 0.001	13.1 +- 0.54	3 service
0.199 +- 0.003	14.8 +- 0.87	37 dst_host_srv_diff_host_rate
0.197 +- 0.001	15.5 +- 0.81	35 dst_host_diff_srv_rate
0.191 +- 0.003	16.8 +- 0.87	10 hot
0.188 +- 0.018	17.1 +- 1.64	14 root_shell

0.178 +- 0.001 18 +- 0.89 8 wrong_fragment

0.189 +- 0.065 19.5 +- 7.09 9 urgent

0.164 +- 0.004 19.9 +- 0.94 34 dst_host_same_srv_rate

0.163 +- 0.006 20.1 +- 1.14 31 srv_diff_host_rate

0.155 +- 0.004 22 +- 0.89 33 dst_host_srv_count

0.152 +- 0.001 22.5 +- 1.02 27 rerror_rate

0.149 +- 0.003 23.9 +- 0.7 23 count

0.14 +- 0.002 25.7 +- 0.78 2 protocol_type

0.136 +- 0.009 26.7 +- 2.1 17 num_file_creations

0.133 +- 0.022 27.3 +- 5.06 18 num_shells

0.131 +- 0.001 27.5 +- 1.2 41 dst_host_srv_rerror_rate

0.128 +- 0.002 28.6 +- 1.36 36 dst_host_same_src_port_rate

0.126 +- 0.004 29.3 +- 1.68 32 dst_host_count

0.122 +- 0.001 30.8 +- 1.08 28 srv_rerror_rate

0.121 +- 0.003 31.2 +- 1.08 1 duration

0.115 +- 0.005 32.9 +- 1.14 13 num_compromised

0.114 +- 0.001 33.2 +- 0.75 40 dst_host_rerror_rate

0.1 +- 0.004 34.9 +- 0.54 16 num_root

0.091 +- 0.001 35.9 +- 0.3 19 num_access_files

0.073 +- 0.002 37 +- 0 15 su_attempted

0.062 +- 0.001 38 +- 0 24 srv_count

0.03 +- 0.024 39.4 +- 0.49 21 is_host_login

0.035 +- 0.01 39.6 +- 0.49 7 land

0 +- 0 41 +- 0 20 num_outbound_cmds

Cost insensitive IGR-RankerSearch Result

=== Run information ===

Evaluator: weka.attributeSelection.GainRatioAttributeEval

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

Relation: NSL-KDDtrain

Instances: 125973

Attributes: 42

List of attributes here are omitted to save space (As listed in appendix A)

class

Evaluation mode: 10-fold cross-validation

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

average merit average rank attribute

0.508 +- 0.003 1 +- 0 26 srv_serror_rate

0.464 +- 0.003 2.2 +- 0.4 25 serror_rate

0.462 +- 0.001 2.8 +- 0.4 4 flag

0.456 +- 0 4 +- 0 12 logged_in

0.442 +- 0.001 5 +- 0 39 dst_host_srv_serror_rate

0.377 +- 0.001 6.1 +- 0.3 30 diff_srv_rate

0.374 +- 0.001 6.9 +- 0.3 38 dst_host_serror_rate

0.292 +- 0.015 8.5 +- 0.67 6 dst_bytes

0.292 +- 0 8.6 +- 0.49 5 src_bytes

0.276 +- 0.01 9.9 +- 0.3 29 same_srv_rate

0.221 +- 0 11.4 +- 0.49 3 service

0.22 +- 0.007 11.8 +- 0.75 11 num_failed_logins

0.212 +- 0.002 12.8 +- 0.4 22 is_guest_login

0.2 +- 0.001 14 +- 0 37 dst_host_srv_diff_host_rate

0.197 +- 0.001 15 +- 0 35 dst_host_diff_srv_rate

0.177 +- 0 16.5 +- 0.5 8 wrong_fragment

0.177 +- 0.006 16.6 +- 0.66 14 root_shell

0.171 +- 0.001 17.9 +- 0.3 10 hot

0.165 +- 0.002 19.5 +- 0.5 34 dst_host_same_srv_rate

0.165 +- 0.002 19.5 +- 0.5 31 srv_diff_host_rate

0.156 +- 0.003 21.6 +- 0.8 33 dst_host_srv_count

0.153 +- 0.001 22.4 +- 0.66 27 rerror_rate

0.15 +- 0.002 23.4 +- 0.49 23 count

0.145 +- 0.016 24.2 +- 3.6 9 urgent

0.139 +- 0 24.7 +- 0.46 2 protocol_type

0.131 +- 0 26 +- 0.77 41 dst_host_srv_rerror_rate

0.129 +- 0.002 27.5 +- 0.92 36 dst_host_same_src_port_rate

0.128 +- 0.004 27.9 +- 1.58 17 num_file_creations

0.129 +- 0.001 28.4 +- 1.36 32 dst_host_count

0.123 +- 0.006 30.1 +- 1.81 18 num_shells

0.123 +- 0.001 30.2 +- 0.87 28 srv_rerror_rate

0.118 +- 0.002 32 +- 1 1 duration

0.113 +- 0.001 33.3 +- 0.64 13 num_compromised

0.114 +- 0.002 33.3 +- 0.78 40 dst_host_rerror_rate

0.099 +- 0.001 35 +- 0 16 num_root

0.09 +- 0 36 +- 0 19 num_access_files

0.073 +- 0 37 +- 0 15 su_attempted

0.06 +- 0.001 38 +- 0 24 srv_count

0.045 +- 0.015 39.1 +- 0.3 21 is_host_login

0.033 +- 0.007 39.9 +- 0.3 7 land

0 +- 0 41 +- 0 20 num_outbound_cmds

Cost sensitive CFS-GeneticSearch Result

=== Run information ===

Evaluator: weka.attributeSelection.CostSensitiveSubsetEval -cost-matrix "[0.0 1.0 2.0 2.0 2.0; 1.0 0.0 2.0 2.0 2.0; 2.0 1.0 0.0 2.0 2.0; 3.0 2.0 2.0 0.0 2.0; 4.0 2.0 2.0 2.0 0.0]" -S 1 -W weka.attributeSelection.CfsSubsetEval --

Search: weka.attributeSelection.GeneticSearch -Z 20 -G 20 -C 0.6 -M 0.033 -R 20 -S 1

Relation: NSL-KDDtrain

Instances: 125973

Attributes: 42

List of attributes here are omitted to save space (As listed in appendix A)

Class

Evaluation mode: 10-fold cross-validation

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

number of folds (%) attribute

0(0 %) 1 duration

2(20 %) 2 protocol_type

8(80 %) 3 service

10(100 %) 4 flag

10(100 %) 5 src_bytes

10(100 %) 6 dst_bytes

1(10 %) 7 land

0(0 %) 8 wrong_fragment

3(30 %) 9 urgent

0(0 %) 10 hot

2(20 %) 11 num_failed_logins

10(100 %) 12 logged_in

4(40 %) 13 num_compromised

1(10 %) 14 root_shell

2(20 %) 15 su_attempted

0(0 %) 16 num_root

5(50 %) 17 num_file_creations

2(20 %) 18 num_shells

1(10 %) 19 num_access_files

0(0 %) 20 num_outbound_cmds

2(20 %) 21 is_host_login

3(30 %) 22 is_guest_login

4(40 %) 23 count

2(20 %) 24 srv_count

9(90 %) 25 serror_rate

9(90 %) 26 srv_serror_rate

0(0 %) 27 rerror_rate

0(0 %) 28 srv_rerror_rate

10(100 %) 29 same_srv_rate

10(100 %) 30 diff_srv_rate

7(70 %) 31 srv_diff_host_rate

7(70 %) 32 dst_host_count

0(0 %) 33 dst_host_srv_count

6(60 %) 34 dst_host_same_srv_rate

5(50 %) 35 dst_host_diff_srv_rate

3(30 %) 36 dst_host_same_src_port_rate

9(90 %) 37 dst_host_srv_diff_host_rate

8(80 %) 38 dst_host_serror_rate

10(100 %) 39 dst_host_srv_serror_rate

0(0 %) 40 dst_host_rerror_rate

0(0 %) 41 dst_host_srv_rerror_rate

Cost insensitive CFS-GeneticSearch Result

=== Run information ===

Evaluator: weka.attributeSelection.CfsSubsetEval

Search: weka.attributeSelection.GeneticSearch -Z 20 -G 20 -C 0.6 -M 0.033 -R 20 -S 1

Relation: NSL-KDDtrain

Instances: 125973

Attributes: 42

List of attributes here are omitted to save space (As listed in appendix A)

Class

Evaluation mode: 10-fold cross-validation

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

Number of folds (%) attributes

2(20 %) 1 duration

6(60 %) 2 protocol_type

6(60 %) 3 service

10(100 %) 4 flag

10(100 %) 5 src_bytes

10(100 %) 6 dst_bytes

0(0 %) 7 land

2(20 %) 8 wrong_fragment

1(10 %) 9 urgent

0(0 %) 10 hot

0(0 %) 11 num_failed_logins

10(100 %) 12 logged_in

2(20 %) 13 num_compromised

3(30 %) 14 root_shell

5(50 %) 15 su_attempted

0(0 %) 16 num_root

1(10 %) 17 num_file_creations

0(0 %) 18 num_shells

0(0 %) 19 num_access_files

0(0 %) 20 num_outbound_cmds

2(20 %) 21 is_host_login

1(10 %) 22 is_guest_login

8(80 %) 23 count

1(10 %) 24 srv_count

3(30 %) 25 serror_rate

10(100 %) 26 srv_serror_rate

0(0 %) 27 rerror_rate

0(0 %) 28 srv_rerror_rate

7(70 %) 29 same_srv_rate

10(100 %) 30 diff_srv_rate

1(10 %) 31 srv_diff_host_rate

3(30 %) 32 dst_host_count

0(0 %) 33 dst_host_srv_count

2(20 %) 34 dst_host_same_srv_rate

9(90 %) 35 dst_host_diff_srv_rate

2(20 %) 36 dst_host_same_src_port_rate

10(100 %) 37 dst_host_srv_diff_host_rate

10(100 %) 38 dst_host_serror_rate

10(100 %) 39 dst_host_srv_serror_rate

0(0 %) 40 dst_host_rerror_rate

0(0 %) 41 dst_host_srv_rerror_rate

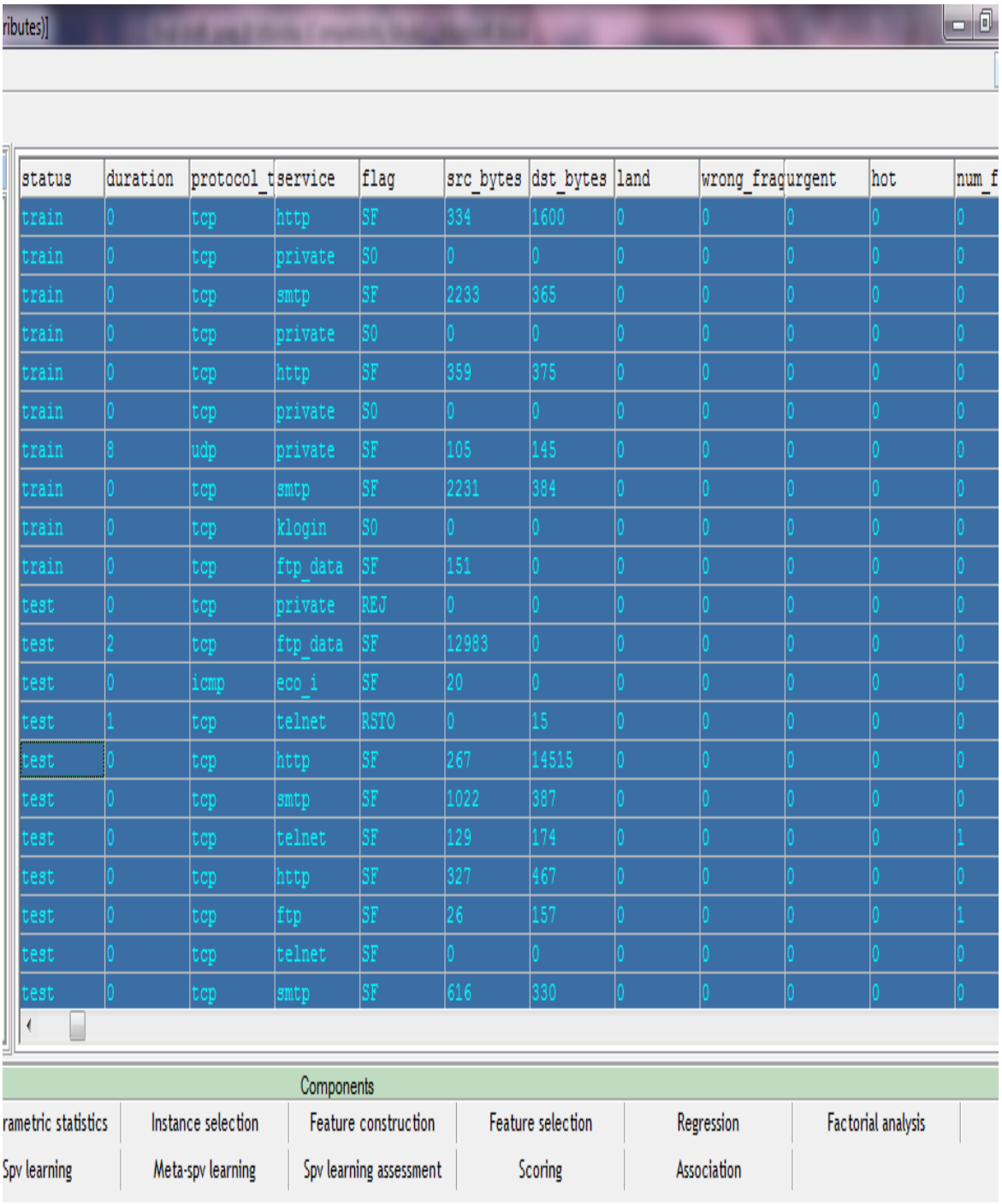
Appendix D

Cost (in) sensitive feature ranking summary from appendix C

S. No	Information Gain Ratio(IGR)				Correlation-based Feature Selection(CFS)		
	Cost insensitive approach		Cost sensitive approach		Cost insensitive approach		Cost sensitive approach
	Gain value	Attributes	Gain value	Attributes	Attributes	Correlation %	Correlation %
1	0.508	26	0.51	26	1	2(20 %)	0(0 %)
2	0.464	25	0.462	4	2	6(60 %)	2(20 %)
3	0.462	4	0.462	25	3	6(60 %)	8(80 %)
4	0.456	12	0.457	12	4	10(100 %)	10(100 %)
5	0.442	39	0.442	39	5	10(100 %)	10(100 %)
6	0.377	30	0.373	38	6	10(100 %)	10(100 %)
7	0.374	38	0.362	30	7	0(0 %)	1(10 %)
8	0.292	6	0.288	5	8	2(20 %)	0(0 %)
9	0.292	5	0.281	6	9	1(10 %)	3(30 %)
10	0.276	29	0.27	29	10	0(0 %)	0(0 %)
11	0.221	3	0.257	11	11	0(0 %)	2(20 %)
12	0.22	11	0.242	22	12	10(100 %)	10(100 %)
13	0.212	22	0.224	3	13	2(20 %)	4(40 %)
14	0.2	37	0.199	37	14	3(30 %)	1(10 %)
15	0.197	35	0.197	35	15	5(50 %)	2(20 %)
16	0.177	8	0.191	10	16	0(0 %)	0(0 %)
17	0.177	14	0.188	14	17	1(10 %)	5(50 %)
18	0.171	10	0.178	8	18	0(0 %)	2(20 %)
19	0.165	34	0.189	9	19	0(0 %)	1(10 %)
20	0.165	31	0.164	34	20	0(0 %)	0(0 %)
21	0.156	33	0.163	31	21	2(20 %)	2(20 %)
22	0.153	27	0.155	33	22	1(10 %)	3(30 %)
23	0.15	23	0.152	27	23	8(80 %)	4(40 %)
24	0.145	9	0.149	23	24	1(10 %)	2(20 %)
25	0.139	2	0.14	2	25	3(30 %)	9(90 %)
26	0.131	41	0.136	17	26	10(100 %)	9(90 %)
27	0.129	36	0.133	18	27	0(0 %)	0(0 %)
28	0.128	17	0.131	41	28	0(0 %)	0(0 %)
29	0.129	32	0.128	36	29	7(70 %)	10(100 %)
30	0.123	18	0.126	32	30	10(100 %)	10(100 %)
31	0.123	28	0.122	28	31	1(10 %)	7(70 %)
32	0.118	1	0.121	1	32	3(30 %)	7(70 %)
33	0.113	13	0.115	13	33	0(0 %)	0(0 %)
34	0.114	40	0.114	40	34	2(20 %)	6(60 %)
35	0.099	16	0.1	16	35	9(90 %)	5(50 %)
36	0.09	19	0.091	19	36	2(20 %)	3(30 %)
37	0.073	15	0.073	15	37	10(100 %)	9(90 %)
38	0.06	24	0.062	24	38	10(100 %)	8(80 %)
39	0.045	21	0.03	21	39	10(100 %)	10(100 %)
40	0.033	7	0.035	7	40	0(0 %)	0(0 %)
41	0	20	0	20	41	0(0 %)	0(0 %)

Appendix E

Screen shot of data preprocessing merged training and testing dataset



ributes)]

status	duration	protocol_t	service	flag	src_bytes	dst_bytes	land	wrong_frag	urgent	hot	num_f
train	0	tcp	http	SF	334	1600	0	0	0	0	0
train	0	tcp	private	S0	0	0	0	0	0	0	0
train	0	tcp	smtp	SF	2233	365	0	0	0	0	0
train	0	tcp	private	S0	0	0	0	0	0	0	0
train	0	tcp	http	SF	359	375	0	0	0	0	0
train	0	tcp	private	S0	0	0	0	0	0	0	0
train	8	udp	private	SF	105	145	0	0	0	0	0
train	0	tcp	smtp	SF	2231	384	0	0	0	0	0
train	0	tcp	klogin	S0	0	0	0	0	0	0	0
train	0	tcp	ftp_data	SF	151	0	0	0	0	0	0
test	0	tcp	private	REJ	0	0	0	0	0	0	0
test	2	tcp	ftp_data	SF	12983	0	0	0	0	0	0
test	0	icmp	eco_i	SF	20	0	0	0	0	0	0
test	1	tcp	telnet	RSTO	0	15	0	0	0	0	0
test	0	tcp	http	SF	267	14515	0	0	0	0	0
test	0	tcp	smtp	SF	1022	387	0	0	0	0	0
test	0	tcp	telnet	SF	129	174	0	0	0	0	1
test	0	tcp	http	SF	327	467	0	0	0	0	0
test	0	tcp	ftp	SF	26	157	0	0	0	0	1
test	0	tcp	telnet	SF	0	0	0	0	0	0	0
test	0	tcp	smtp	SF	616	330	0	0	0	0	0

Components

rametric statistics | Instance selection | Feature construction | Feature selection | Regression | Factorial analysis

Spv learning | Meta-spv learning | Spv learning assessment | Scoring | Association

Appendix F

Confusion and computed metrics for all (41) features

Error rate			0.0105						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9873	0.0035	NORMAL	11498	95	13	4	36	11646
PROBE	0.9930	0.0599	PROBE	10	2136	5	0	0	2151
DOS	0.9963	0.0022	DOS	5	25	8125	0	0	8155
U2R	0.8293	0.0562	U2R	4	4	0	84	2	94
R2L	0.8817	0.0757	R2L	21	12	0	1	464	498
			Sum	11538	2272	8143	89	502	22544
Computed metrics			**FPR=1.05 , AMC= 0.0172, OCA= 98.94 , TTBM=14867**						

Confusion Matrix for CS-CM4 (41features)

Error rate			0.0055						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9953	0.0046	NORMAL	11591	14	8	4	29	11646
PROBE	0.994	0.0088	PROBE	8	2138	4	1	0	2151
DOS	0.9979	0.0015	DOS	12	5	8138	0	0	8155
U2R	0.9149	0.0753	U2R	5	0	0	86	3	94
R2L	0.9398	0.064	R2L	28	0	0	2	468	498
			Sum	11644	2157	8150	93	500	22544
Computed metrics			**FPR= 0.545, AMC= 0.0124, OCA=99.45,TTBM=16146**						

Confusion Matrix for MetaCost C4.5 (41features)

Appendix G

Confusion and computed metrics for Cost sensitive feature selection approach

Error rate			0.0129						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9866	0.0065	NORMAL	11490	111	11	14	20	11646
PROBE	0.9912	0.0708	PROBE	22	2125	1	3	0	2151
DOS	0.9956	0.0018	DOS	7	40	8108	0	0	8155
U2R	0.8825	0.1935	U2R	11	8	0	75	0	94
R2L	0.9437	0.042	R2L	35	3	3	1	456	498
			Sum	11565	2287	8123	93	476	22544
Computed metrics			**FPR= 1.15 , AMC= 0.0182 , OCA=99.03, TTBM=5491**						

Confusion Matrix for CS-CM4-IGR cost sensitive (19 features)

Error rate			0.0094						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9905	0.0044	NORMAL	11535	81	11	5	14	11646
PROBE	0.9921	0.0537	PROBE	12	2134	4	0	1	2151
DOS	0.9962	0.0018	DOS	5	26	8124	0	0	8155
U2R	0.8936	0.0769	U2R	3	4	0	84	3	94
R2L	0.9337	0.0381	R2L	31	10	0	2	455	498
			Sum	11586	2255	8139	91	473	22544
Computed metrics			**FPR= 0.93, AMC= 0.0164, OCA= 99.05, TTBM=9890**						

Confusion Matrix for CS-CM4-IGR cost sensitive (34 features)

Error rate			0.0067						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9958	0.0069	NORMAL	11597	20	11	8	10	11646
PROBE	0.9925	0.0143	PROBE	14	2134	3	0	0	2151
DOS	0.9969	0.002	DOS	14	11	8130	0	0	8155
U2R	0.8998	0.117	U2R	8	0	0	83	3	94
R2L	0.9379	0.0282	R2L	45	0	2	3	448	498
			Sum	11678	2165	8146	94	461	22544
Computed metrics			**FPR= 0.589, AMC= 0.0130, OCA= 99.41, TTBM=6677**						

Confusion Matrix for MetaCost C4.5-IGR cost sensitive (19 features)

Error rate			0.0059						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9946	0.0046	NORMAL	11583	15	9	4	35	11646
PROBE	0.994	0.0093	PROBE	8	2138	3	1	1	2151
DOS	0.9979	0.0015	DOS	12	5	8138	0	0	8155
U2R	0.9149	0.0753	U2R	5	0	0	86	3	94
R2L	0.9378	0.0771	R2L	29	0	0	2	467	498
			Sum	11637	2158	8150	93	506	22544
Computed metrics			**FPR= 0.585, AMC= 0.0132, OCA= 99.41, TTBM=11014**						

Confusion Matrix for MetaCost C4.5-IGR cost sensitive (34 features)

Error rate			0.0073						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9933	0.0055	NORMAL	11568	20	11	5	42	11646
PROBE	0.9874	0.0135	PROBE	17	2124	5	2	3	2151
DOS	0.9982	0.002	DOS	7	8	8140	0	0	8155
U2R	0.9149	0.0851	U2R	5	1	0	86	2	94
R2L	0.9277	0.0923	R2L	35	0	0	1	462	498
			Sum	11632	2153	8156	94	509	22544
Computed metrics			**FPR= 0.616, AMC= 0.0142, OCA= 99.38, TTBM=4150**						

Confusion Matrix for MetaCost C4.5-CFS cost sensitive (17 features)

Error rate			0.0061						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9951	0.0052	NORMAL	11589	8	8	2	39	11646
PROBE	0.9912	0.0079	PROBE	13	2132	5	1	0	2151
DOS	0.9975	0.0016	DOS	11	9	8135	0	0	8155
U2R	0.8936	0.0562	U2R	6	0	0	84	4	94
R2L	0.9357	0.0845	R2L	30	0	0	2	466	498
			Sum	11649	2149	8148	89	509	22544
Computed metrics			**FPR= 0.612, AMC= 0.0142, OCA=99.38, TTBM=8892**						

Confusion Matrix for MetaCost C4.5-CFS cost sensitive (31 features)

Error rate			0.0132						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9831	0.0052	NORMAL	11449	128	12	11	46	11646
PROBE	0.9888	0.07	PROBE	19	2127	1	2	2	2151
DOS	0.9979	0.0016	DOS	4	13	8138	0	0	8155
U2R	0.8511	0.1489	U2R	9	3	0	80	2	94
R2L	0.9096	0.0994	R2L	28	16	0	1	453	498
			Sum	11509	2287	8151	94	503	22544
Computed metrics			**FPR= 1.317, AMC= 0.0221, OCA= 98.68, TTBM=4337**						

Confusion Matrix for CS-CM4-CFS cost sensitive (17 features)

Error rate			0.0103						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9887	0.0052	NORMAL	11514	91	11	6	24	11646
PROBE	0.9912	0.052	PROBE	10	2132	7	1	1	2151
DOS	0.9964	0.0025	DOS	9	20	8126	0	0	8155
U2R	0.8936	0.0968	U2R	5	3	0	84	2	94
R2L	0.9137	0.056	R2L	36	3	2	2	455	498
			Sum	11574	2249	8146	93	482	22544
Computed metrics			**FPR=1.033, AMC= 0.0181, OCA= 98.96, TTBM=10483**						

Confusion Matrix for CS-CM4-CFS cost sensitive (31 features)

Appendix H

Confusion and computed metrics for Cost insensitive feature selection approach

Error rate			0.0115						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9852	0.0039	NORMAL	11474	111	8	5	48	11646
PROBE	0.9875	0.0649	PROBE	15	2132	2	1	1	2151
DOS	0.9936	0.0012	DOS	5	27	8123	0	0	8155
U2R	0.7978	0.0879	U2R	5	6	0	83	0	94
R2L	0.9157	0.0940	R2L	20	4	0	2	472	498
			Sum	11519	2280	8133	91	521	22544
Computed metrics	**FPR= 1.28, AMC=0.0213, OCA= 98.84, TTBM=6630**								

Confusion Matrix for CS-CM4-IGR cost insensitive (23 features)

Error rate			0.0105						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9876	0.0034	NORMAL	11502	91	9	4	40	11646
PROBE	0.9930	0.0582	PROBE	6	2136	7	2	0	2151
DOS	0.9957	0.0020	DOS	4	31	8120	0	0	8155
U2R	0.9149	0.0851	U2R	4	3	0	86	1	94
R2L	0.9317	0.0812	R2L	25	7	0	2	464	498
			Sum	11541	2268	8136	94	505	22544
Computed metrics	**FPR= 1.046, AMC= 0.0178, OCA= 98.95, TTBM=9859**								

Confusion Matrix for CS-CM4-IGR cost insensitive (30 features)

Error rate			0.0139						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9843	0.0062	NORMAL	11463	134	11	17	21	11646
PROBE	0.9921	0.0814	PROBE	15	2134	1	0	1	2151
DOS	0.9946	0.0015	DOS	7	37	8111	0	0	8155
U2R	0.7553	0.1932	U2R	13	8	0	71	2	94
R2L	0.9076	0.0504	R2L	36	10	0	0	452	498
			Sum	11534	2323	8123	88	476	22544
Computed metrics			**FPR= 1.388, AMC= 0.0232, OCA= 98.92, TTBM=3620**						

Confusion Matrix for CS-CM4-CFS cost insensitive (15 features)

Error rate			0.0106						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.988	0.0048	NORMAL	11506	109	10	6	15	11646
PROBE	0.993	0.0627	PROBE	7	2136	6	1	1	2151
DOS	0.9958	0.002	DOS	14	20	8121	0	0	8155
U2R	0.8936	0.0769	U2R	5	3	0	84	2	94
R2L	0.9177	0.0379	R2L	30	11	0	0	457	498
			Sum	11562	2279	8137	91	475	22544
Computed metrics			**FPR=1.064, AMC= 0.0187, OCA= 98.93, TTBM=8596**						

Confusion Matrix for CS-CM4-CFS cost insensitive (29 features)

Error rate			0.0059						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9961	0.0052	NORMAL	11589	15	6	6	30	11646
PROBE	0.9924	0.0107	PROBE	12	2136	2	0	1	2151
DOS	0.9974	0.001	DOS	13	8	8134	0	0	8155
U2R	0.8827	0.086	U2R	6	0	0	85	3	94
R2L	0.8997	0.0679	R2L	29	0	0	2	467	498
			Sum	11649	2159	8142	93	501	22544
Computed metrics			**FPR= 0.674, AMC=0.0158, OCA= 99.32, TTBM=7722**						

Confusion Matrix for MetaCost C4.5-IGR cost insensitive (23 features)

Error rate			0.0062						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9948	0.0051	NORMAL	11585	13	9	4	35	11646
PROBE	0.9902	0.0088	PROBE	12	2130	8	1	0	2151
DOS	0.9979	0.0021	DOS	11	6	8138	0	0	8155
U2R	0.9255	0.0745	U2R	4	0	0	87	3	94
R2L	0.9317	0.0757	R2L	32	0	0	2	464	498
			Sum	11644	2149	8155	94	502	22544
Computed metrics	**FPR= 0.621, AMC= 0.0140, OCA= 99.37, TTBM=11981**								

Confusion Matrix for MetaCost C4.5-IGR cost insensitive (30 features)

Error rate			0.0062						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9954	0.0056	NORMAL	11592	14	9	7	24	11646
PROBE	0.9921	0.0097	PROBE	8	2134	4	4	1	2151
DOS	0.9977	0.0017	DOS	13	6	8136	0	0	8155
U2R	0.9149	0.1224	U2R	4	1	1	86	2	94
R2L	0.9177	0.0558	R2L	40	0	0	1	457	498
			Sum	11657	2155	8150	98	484	22544
Computed metrics	**FPR= 0.727, AMC= 0.0317, OCA= 99.27, TTBM=4665**								

Confusion Matrix for MetaCost C4.5-CFS cost insensitive (15 features)

Error rate			0.0062						
Values prediction			Confusion matrix						
Value	Recall	1-Precision		NORMAL	PROBE	DOS	U2R	R2L	Sum
NORMAL	0.9946	0.0053	NORMAL	11583	9	12	9	33	11646
PROBE	0.9921	0.007	PROBE	11	2134	4	1	1	2151
DOS	0.9978	0.002	DOS	12	6	8137	0	0	8155
U2R	0.9043	0.1053	U2R	7	0	0	85	2	94
R2L	0.9357	0.0717	R2L	32	0	0	0	466	498
			Sum	11645	2149	8153	95	502	22544
Computed metrics	**FPR= 0.616, AMC= 0.0143, OCA= 99.38, TTBM=8596**								

Confusion Matrix for MetaCost C4.5-CFS cost insensitive (29 features)