A DECISION TREE BASED INTRUSION DETECTION SYSTEM WITH BOOTSTRAP AGGREGATING, DISCRETIZATION, AND FEATURE SELECTION

by

Seray Özdemir

B.S., Telecommunication Engineering, Istanbul Technical University, 2011

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2014

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my advisor, Prof. Emin Anarım, for his great expertise and understanding. He became not only a mentor but also a family member to me with his encouragement, motivation, and patience. Without his supervision and constant support this study would not have been possible.

I must also acknowledge Prof. Fatih Alagöz and Prof. Mehmet Akar for spending their valuable time and sharing their comments.

Many thanks to my dear friends Cemil Eren Kayataş, Yasir Korkusuz, Ramin Fouladi, and Derya Erhan. It was great to work with them.

Special thanks to my parents Serap Özdemir, Nihat Özdemir and my sisters Neşe Özdemir, Hatice Özdemir who stand by me with all their hearts in all circumstances. They always trust me and support my dreams. I am very lucky to have such an excellent family.

Finally, I dedicate this thesis to love of my life Gökhan Çoban. With his presence, sun is brighter, rain is more romantic, weekends are more exciting, sitting at home is more comfortable, shopping is crazier, dancing is funnier, food is more delicious, films are more enjoyable, problems are easier, and future is more promising.

ABSTRACT

A DECISION TREE BASED INTRUSION DETECTION SYSTEM WITH BOOTSTRAP AGGREGATING, DISCRETIZATION, AND FEATURE SELECTION

In this thesis, many machine learning techniques which are used for network intrusion detection are analyzed in detail. An intrusion detection system based on a combination of bootstrap aggregating, discretization, feature selection and classification methods is proposed for achieving a successful denial of service attack detection rate. Detecting denial of service attacks is the main purpose of this study, but detecting other kinds of network attacks and normal network traffic correctly is also our concern. We use various filters on training dataset before we form the model. Firstly, the bootstrap aggregating method is applied for creating different training datasets from the original dataset and combining the results that come from each of them. Secondly, entropy based discretization, equal-width binning, equal-frequency binning, and proportional k-interval discretization methods are used for discretizing the numeric attribute values. Finally, correlation based feature selection, consistency based feature selection, information gain based feature selection, and symmetrical uncertainty based feature selection methods are applied for decreasing the complexity. After the filtering steps, J48 decision tree classifier is used for learning. Then, the model is tested with a distinct dataset. KDD'99 training and testing datasets are used for experiments. Our combined method has increased the success of single classifier for all performance measures. Especially, adding bootstrap aggregating to filtered and attribute selected classifier provided a remarkable improvement.

ÖZET

YERİNE GERİ KOYARAK ÖRNEKLEME, AYRIKLAŞTIRMA VE ÖZNİTELİK SEÇME KULLANAN KARAR AĞACI TEMELLİ SALDIRI TESPİT SİSTEMİ

Bu tezde saldırı tespit sistemlerinde kullanılan birçok makine öğrenmesi tekniği detaylı olarak incelenmiştir. Başarılı bir hizmeti engelleme saldırısı tespit oranına ulaşmak amacıyla, yerine geri koyarak örnekleme, ayrıklaştırma, öznitelik seçme ve sınıflandırma yöntemlerinin birleşimini temel alan bir saldırı tespit sistemi önerilmiştir. Temel amacımız hizmeti engelleme saldırılarını tespit etmek olmakla birlikte farklı çeşitlerdeki ağ saldırılarını ve normal ağ trafiğini doğru olarak tespit etmek de ilgimiz dahilindedir. Modelimizi oluşturmadan önce eğitme veri kümesinin üzerinde çeşitli filtreler kullandık. İlk olarak, orijinal eğitme veri kümesinden farklı eğitme veri kümeleri oluşturmak ve her birinden gelen sonuçları birleştirmek için yerine geri koyarak örnekleme metodu uygulanmıştır. İkinci olarak, sayısal öznitelik değerlerini ayrıklaştırmak için, düzensizlik temelli ayrıklaştırma, eşit genişlikli gruplama, eşit frekanslı gruplama ve orantısal k aralıklı ayrıklaştırma yöntemleri kullanılmıştır. Son olarak, karmaşıklığı azaltmak için, korelasyon temelli öznitelik seçme, tutarlılık temelli öznitelik seçme, bilgi kazancı temelli öznitelik seçme ve simetrik belirsizlik temelli öznitelik seçme yöntemleri kullanılmıştır. Filtrelerden sonra öğrenme için J48 karar ağacı kullanılmıştır. Deneylerde KDD'99 eğitme ve test etme veri kümeleri kullanılmıştır. Birleştirilmiş yöntemimiz her performans ölçüsü için tekil sınıflandırıcının başarısını arttırmıştır. Özellikle, filtrelenmiş ve öznitelikleri seçilmiş sınıflandırıcıya yerine geri koyarak örnekleme yöntemini eklemek dikkate değer bir gelişme sağlamıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A_i	i_{th} attribute
C_i	i_{th} class
N_m	Number of instances that reach node m
N_{mi}	Number of instances that reach node m and belong to class C_i
$\overline{r_{cf}}$	Average feature-class correlation
$\overline{r_{ff}}$	Average feature-feature inter-correlation
w_{m0}	Threshold value that splits the space
β	Stability
δ	Minimum Description Length (MDL) stopping criterion

LIST OF ACRONYMS/ABBREVIATIONS

ARFF	Attribute-Relation File Format
ART	Adaptive Resonance Theory
AUC	Area Under Curve
Bagging	Bootstrap Aggregating
BPN	Back-Propagation Network
CFS	Correlation Based Feature Selection
CONS	Consistency Based Feature Selection
CWSN	Cluster-bases Wireless Sensor Networks
DARPA	Defense Advanced Research Projects Agency
DDOS	Distributed Denial of Service
DOS	Denial of Service
EBD	Entropy Based Discretization
EFD	Equal Frequency Binning Discretization
EMD	Entropy Minimization Discretization
EWD	Equal Width Binning Discretization
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GUI	Graphical User Interface
HNB	Hidden Naive Bayes
HIDS	Host Based Intrusion Detection System
IDS	Intrusion Detection System
IG	Information Gain Based Feature Selection
KDD	Knowledge Discovery and Data Mining
LAN	Local Area Network
LDA	Linear Discriminant Analysis
MDL	Minimum Description Length
NIDS	Network Based Intrusion Detection System

PCA	Principle Components Analysis
PKID	Proportional K-Interval Discretization
R2L	Remote to Local
ROC	Receiver Operating Characteristic
SOM	Self Organizing Map
SU	Symmetrical Uncertainty Based Feature Selection
SVM	Support Vector Machine
TN	True Negative
TPR	True Positive Rate
TP	True Positive
U2R	User to Root
WEKA	Waikato Environment for Knowledge Analysis

1. INTRODUCTION

Exceeding increase of computer and internet usage created a communication revolution that affects all sectors. This revolution also changed people's lifestyles. Internet has become an essential need for human life. With this extreme usage of internet, network security has become a major challenge. Many solutions have been developed for providing a secure communication. However, numerous threats continue to arise in a faster way. This is a kind of cycle, new security problems appear as soon as the current one has been solved.

Recovering the systems after attacks usually costs much more than establishing a security mechanism. There are various mechanisms such as firewalls, software updates, encryption, etc. According to security requirements and policies, one or more mechanisms can be used.

An intrusion detection system (IDS) is a kind of security mechanism that is used for monitoring the systems dynamically and alerting the administrators. Intrusion detection systems give information about the type, location, and source of the intrusion. An IDS can model the normal behaviour and detects the traffic as an attack if it deviates from the model. This kind of approach is called anomaly detection. An IDS can also model the different attacks' behaviour and detects the traffic as an attack if it fits the model. This kind of approach is called misuse detection or signature detection.

Denial of Service (DOS) Attack is one of the most common attack types. DOS attacks pose a threat against the availability of computing systems. Legitimate users and servers communicate continuously and users send requests for establishing and maintaining connections. The attacker sends numerous illegal requests to server maliciously. A server has a capacity for supplying service and if the requests exceed the capacity, it cannot respond the requests and denies them. Legitimate users cannot access to the server either. If this attack is performed by a single attacker, the attack is called DOS, if it is performed by many attackers simultaneously, the attack is called

Distributed Denial of Service (DDOS) Attack.

The motivation for this thesis is providing a secure network environment and preventing malicious attacks. Many methods have been examined for this purpose. In previous works, researchers have used many methods for dealing with network intrusions. Some researchers proposed intrusion detection systems that use single classifier. Some researchers used discretization or feature selection filters for improving the performance. Some researchers used ensemble classifiers by combining the classifiers with bootstrap aggregating, boosting, or stacking methods. In our work, we prefer to use an ensemble classifier with various filters. We use J48 decision tree and make it ensemble by bootstrap aggregating method. We take the advantage of using bootstrap aggregating which is a great method with decision tree based classifiers. We also use discretization and feature selection filters for improving the results. In literature review, we couldn't find any works that combine all of those four methods; bootstrap aggregating, discretization, feature selection, and decision tree. Therefore, we propose an intrusion detection system that combines them and aim to achieve satisfactory results.

The organization of this thesis is as follows. In the next chapter, general information about intrusion detection systems is given by explaining computer security's keystones. Additionally, many related works are analyzed. In the third chapter, fundamentals of the proposed intrusion detection system is explained in detail with mathematical descriptions about the discretization, feature selection, classification and bootstrap aggregating techniques. In the fourth chapter, the proposed intrusion detection system architecture is introduced. Also, the information about the dataset and implementation environment are given. In the following chapter, numerical evaluations are given and the study is compared with other works. The last chapter is the conclusion and it covers recommendations for future works.

2. OVERVIEW OF INTRUSION DETECTION SYSTEMS

There are many threats against the computing systems in local or wide area network environment. Besides, all computing systems have some vulnerabilities. For the purpose of computing systems not to be affected by the threats, it is needed to have some security procedures, mechanisms, and countermeasures.

Intrusion detection systems are one of the countermeasures. The main goal of intrusion detection is providing resistance to malicious attacks by detecting them at the moment of occurring or before.

In this chapter, computer security is described briefly with its basic aspects. Intrusion detection systems are explained according to different types and approaches. Additionally, many articles have been examined during the studies for this thesis. Literature review and related works are located in the last section of this chapter.

2.1. Computer Security

2.1.1. General Description and Aspects of Computer Security

Computer security can be defined as all the technological mechanisms and processes applied for the purpose of protecting computing systems against threats.

Computer security is examined by three basic aspects [1]:

- Security services
- Security mechanisms
- Security attacks

Security services enhance the security of computing systems of an organization. Services can be classified as [13]:

- Confidentiality is secretion of information or resources. It ensures that information is not accessed, observed, or copied by unauthorized people.
- Integrity is reliability of information or resources. It ensures that information is not modified by unauthorized people.
- Availability is ability to access desired information or resources. It ensures that information is not destructed, monopolized, or contaminated.

Security mechanisms are designed to detect, prevent or recover from a security attack. There are many kinds of mechanisms that provide security services. Most of them rely on cryptographic techniques. Encryption, decryption algorithms, digital signature schemes, integrity check or hash functions are the various examples of mechanisms.

Security attacks are any action that threatens the information security. Attacks can basically be classified into four groups:

- Interruption is an attack against the availability of computing system. Whole system or a part of it becomes unable to be used, obtained, or accessed. Damaging the device or communication cable can be an example for this type of attack.
- Interception is an attack against the confidentiality of computing system. Unauthorized parties gain ability for system access. This is a passive attack and the goal is learning or making use of information from the system but not affecting the system resources. Traffic analysis is an example for this type of attack.
- Modification is an attack against the integrity of computing system. Unauthorized parties gain ability for system access and interfere the system in order to damage or alter something. Changing message contents is an example for this type of attack.
- Fabrication is an attack against the authenticity of computing system. Unauthorized parties gain ability for creating fake objects on the computing system. Adding irrelevant records to a database is an example for this type of attack.

Four main groups of attacks that are mentioned above can be seen in Figure 2.1.

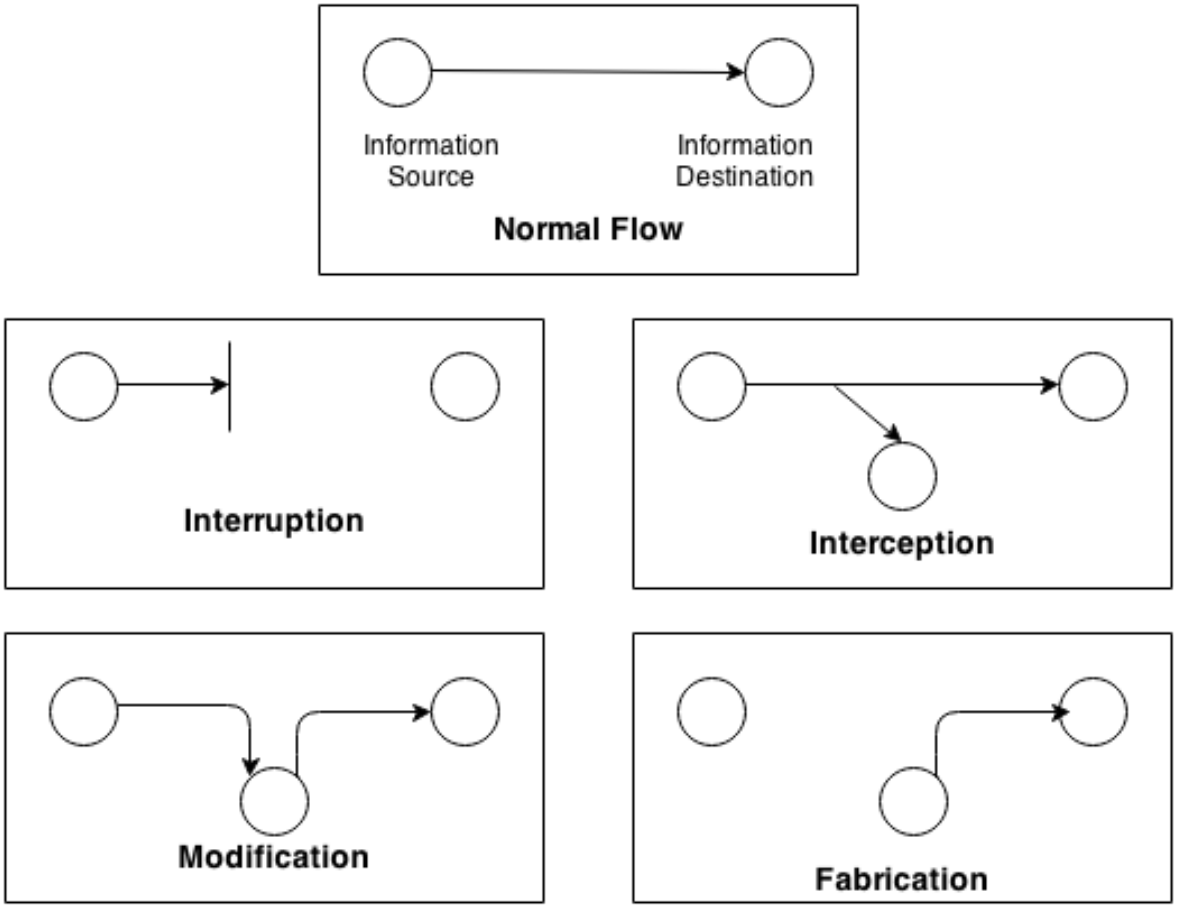

Figure 2.1. Normal flow and four main groups of attacks [1].

2.1.2. Countermeasures

All computing systems have various vulnerabilities. These vulnerabilities can be related to hardware, software, or data. In the ideal case, all vulnerabilities should be identified and fixed, but this is impossible because of technical and economical reasons. Modifying the systems too frequently for decreasing the vulnerabilities is also impractical. Therefore, it is needed to determine some countermeasures for avoiding risks and compensating vulnerabilities. Countermeasures are listed below [14]:

- Prevention is the main goal of computer security. Absolute prevention is the case

that all attacks fail due to some preventative mechanisms. It is valid only in theory, not in practice. Prevention can be implemented by abolishing vulnerabilities or inhibiting the attack.

- Deterrence is making the attack more difficult for attackers. If there are heavy punishments for the action, attacker does not risk himself because of his probability of getting caught. Generating fear is a method of deterrence. Another way is creating more secure defence mechanisms. If attacking is difficult and time consuming, attacker will give up dealing.
- Deflection is tricking the attacker by showing another target and making him believe that this target is more valuable.
- Detection is auditing and monitoring the computing system against the possibility of attack occurrence. After accepting the possibility of attack occurrence, the tasks of detection mechanisms are identifying that an attack is occurring, or has occurred, and reporting it. Additionally, log-keeping is a task of detection mechanisms. If prevention mechanisms fail, detection mechanisms should take over the mission, preferably while there's still time to prevent damage. For this reason, detection works collectively with prevention.
- Recovery provides computing systems not to be affected from damages of attacks. If all mechanisms fail, recovery mechanisms restore the damaged files or whole system from scratch by the help of backups. In order to prevent the repetition of attack, these mechanisms should identify and fix the vulnerabilities.

2.2. Intrusion Detection Systems

Up to this time, many methods for secure communication have been developed like firewalls or encryption techniques. Despite these, the security problem still continues. Intrusion detection is an addition to the other techniques. Intrusion detection systems are becoming a major component for today's secure networks. A system intrusion is any attempt to attack a system and compromise its security aspects such as integrity, confidentiality, or availability.

Intrusion Detection Systems are implemented to detect an unusual access or at-

tack when it occurs. In brief, IDS provides a resistance to external attacks. IDS can detect different attack types and classify the attacks, whereas the traditional techniques cannot perform this task.

In general IDS has the task of monitoring the traffic, analyzing the packets and deciding whether they are attacks or not.

2.2.1. Types Of Intrusion Detection Systems

There are two types of intrusion detection system according to the computing system they monitor: Host based IDS and Network based IDS. Host based intrusion detection system (HIDS) monitors files, access attempts, or log files of an individual host or network device such as important servers. Network based intrusion detection system (NIDS) is placed on a particular network segment and monitors the traffic flow on that network segment [15]

2.2.1.1. Host Based Intrusion Detection. Host based intrusion detection systems monitor the inbound and outbound packets of a single device and scan the log files periodically. If a suspicious activity is detected, HIDS warns the system administrator.

2.2.1.2. Network Based Intrusion Detection. Network based intrusion detection systems are more popular than host based intrusion detection systems. The information resource is network packets for network based intrusion detection systems. They are placed at a strategic point for monitoring the traffic of all components of network. They monitor the network, analyze the traffic, and report the findings to the administrator.

2.2.2. Intrusion Detection Approaches

Intrusion detection systems can be categorized into two major groups based on their detection approaches. One of them is anomaly detection and the other is misuse detection. Anomaly detection is behavior based, misuse detection is signature based.

2.2.2.1. Anomaly Detection. Anomaly detection creates a model for normal behavior and compares all of the packets with this model. Because of noise and other external effects, some of the normal packets do not fit to the model of normal behavior. As a consequence anomaly detection has a high detection rate, but false alarm rates are also high [16].

2.2.2.2. Misuse Detection. Misuse detection creates models of different types of attack behaviors and compares the packets with these behaviors. False alarm rate is low in misuse detection; it detects attacks with high accuracy. The disadvantage of misuse detection is low detection rates because against unknown attacks which are not modeled before, misuse detection will not have a good performance [16].

2.3. Literature Review and Related Works

Intrusion Detection is a very popular subject among academia. There are many research about intrusion detection systems.

Derpen *et al.*, proposed an intelligent hybrid intrusion detection system using both anomaly and misuse detection. They selected six features manually and used Self Organizing Map (SOM) structure for anomaly detection, J48 Decision Tree for signature detection. They combined the results by a rule based decision support system [15].

Wang *et al.*, proposed an integrated IDS for cluster-based wireless sensor networks(CWSN). It is integrated because three different intrusion detection systems were used for different kinds of node segments. They used rule-based method for anomaly detection, Back-Propagation Network (BPN) for misuse detection, and Adaptive Resonance Theory (ART) as a learning mechanism that analyzes the system in real-time and makes the IDS intelligent [16].

Lee *et al.*, proposed an IDS that uses decision tree. They applied feature selection

for more meaningful results. With binary classification point of view, they divided the data as positive and negative classes. Positive class represents the targeted attack class. Negative class represents any class labels except the targeted attack class. For example, if the target is detecting DOS type of attack, positive class is the instances which belong to DOS attack type, negative class is the instances which belong to R2L, U2R, probe attack types or normal traffic. For each attack type, an ID3 decision tree was built [17].

Bolon-Canedo *et al.*, compared the performances of intrusion detection systems which combine different discretization, feature selection and classification methods. They handled the problem with binary approach. They used different discretization methods such as Entropy Minimization Discretization (EMD) and Proportional k-Interval Discretization (PKID), different filters such as INTERACT and consistency based. They used different classifiers such as C4.5 decision tree and Naive Bayes. Several combinations were tested on KDD'99 dataset to decide which combination gives better result. The results obtained show that both discretizers and filters have a great influence on the performance of the classifiers [18].

Chandolikar and Nandavadekar proposed an IDS that extracts features from training dataset and applies a classification algorithm for learning. They used REP-Tree, RandomTree, J48 tree based classifiers, a rule generator algorithm (OneR), and Bayesian classifier. As a result, they saw that decision tree algorithms showed better performance. Among decision tree algorithms, J48 is the most advanced one and it has the best performance as a classifier in this case [19].

Koshal and Bag proposed a network based IDS by combining two algorithms: C4.5 decision tree and Support Vector Machine (SVM). They aimed a better performance and low false positive rate by combining the advantages of these two algorithms. Correlation based feature selection (CFS) was applied as a filter. Their structure was putting preprocessed training set into two different classifiers; then, taking the predictions that came from decision trees and passing it through the support vector machine block. At the end, they combined the results that came from the single SVM structure

and hybrid Decision Tree plus SVM structure [20].

Most hybrid intrusion detection systems use two different classifiers as anomaly detection module and misuse detection module and combine the results. Kim *et al.*, proposed something different from these approaches. A method that hierarchically integrates a misuse detection module and an anomaly detection module was proposed. The authors of this paper think that normal training data can be grouped into different classes based on the characteristics. They claim that a normal dataset has various types of normal connections and using only anomaly detection module cannot represent the data precisely. Misuse detection module is used for classifying normal data. After classifying the normal data into more specific groups, anomaly detection module was applied for each subset. C4.5 decision tree was used for misuse detection and one-class support vector machine was used for creating multiple anomaly detection modules. In the proposed method, the anomaly detection module can indirectly use the known attack information in order to enhance its ability to build profiles of normal behaviour. The detection rates for unknown and known attacks were better than the traditional intrusion detection systems. Additionally, their method also reduced the time complexity because its anomaly detection module works on smaller data subsets [21].

Thaseen and Kumar combined discretization, feature selection and classification methods on their work. They used CFS and CONS feature selection methods and a supervised discretization technique. They used several decision tree algorithms: ADTree, C4.5, LADTree, NBTree, RandomTree, RandomForest, REPTree and compared their results. They reached better accuracy with CFS+Discretization+RandomTree combination [22].

Koc *et al.*, proposed an intrusion detection system that uses Hidden Naive Bayes (HNB) classifier. They claim HNB results better for high dimensional data with highly correlated features. They handled the problem with multi-class classification approach. They combined two discretization method: Entropy Minimization Discretization (EMD) and PKID with three feature selection methods: CFS, CONS, INTERACT. After preprocessing they used several classifiers for comparing the performances.

They reached the best results with the combination of PKID+INTERACT+HNB [9].

Gyanchandani *et al.*, combined classifiers by several methods. They used bootstrap aggregating (bagging), boosting, and stacking methods. With bagging, they produced classifiers for different training sets. With boosting, they gave weight to classifiers and increased the effect of the more successful models. With stacking, they used different learning algorithms as classifiers and combined the predictions by Meta Learner. Bagging gave the best results in their work [23].

Alhaddad *et al.*, used bagging and boosting in their works. They used J48 and NaiveBayes classifiers for testing bagging and boosting. Their boosting method was AdaBoost.M1. They also used Random Forest, which is also an ensemble technique. Random Forest is a decision tree method, but it does not always choose best attribute during the tree construction. It chooses an attribute randomly among a few of the best options. This creates different trees every time. As a result, they found that decision tree ensembles works better than Naive Bayes ensembles and bagging gives the best predictions. Random forest also is better than boosting [24].

3. FUNDAMENTALS OF THE PROPOSED INTRUSION DETECTION SYSTEM

In the proposed intrusion detection system, many machine learning techniques and algorithms are used. In this chapter, discretization, feature selection, classification, and bootstrap aggregating methods are explained in detail with mathematical descriptions.

Discretization is applied to make continuous numeric attributes more meaningful for calculations by quantizing them. Feature selection is applied to ignore unnecessary or ineffective features for preventing the calculations to be complicated and the results to be inaccurate. For classification, decision tree algorithm is applied because it is a simple and fast algorithm that provides possibility to visualize data in a comprehensible way. Bootstrap aggregating is applied to create new training sets from the original set, build a classifier for each new set and combine the predictions that come from each classifier.

3.1. Discretization

Some of the algorithms that are used for classification can only deal with nominal attributes. Datasets generally contain both nominal and numeric attributes. Numeric attributes must be discretized before being used in these algorithms [3]. Moreover, some of the classifiers can deal with numeric attributes by assuming that numeric attributes have a normal distribution. This assumption prevents classifier to create satisfactory results because in practice this is not a very reasonable assumption. On the other hand, some other classifiers such as decision trees can also handle numeric attributes, but sorting the attributes' values and processing them brings much longer running time to build the model.

There are different ways of discretization. The first one is global method. Global

method is applied before learning procedure. Instances are sorted by attribute's value and values are assigned into ranges at the points that the class value changes. There are also a minimum number of instances that should be in a range. Thus, different classes can be found in a range. If the global method is used, each discretization interval is interpreted as a nominal value. However, numeric attributes are sorted and interpreting them as nominal can cause to important information loss about data. It is better to transform each attribute into a set of binary attributes before learning procedure.

Secondly, there is local method used by decision tree learners. Decision trees can handle numeric attributes on local basis, which means they discretize the attributes at each node of the tree while the tree building process continues. Because decision tree evaluates the attributes at each node, it produces different discretizations of the same attribute at different places.

3.1.1. Discretization Methods

Discretization methods can be categorized according to knowledge of classes. Unsupervised discretization means quantizing each attribute without knowing the class information. Alternatively, supervised discretization takes the class information into account [3].

3.1.1.1. Equal Width Binning (Unsupervised). In this method, discretization is done by dividing the data into equal width intervals [3]. An obvious disadvantage of this method is that instances cannot be separated equivalently. Some intervals can have many instances.

Think of an example dataset with attribute values [0, 4, 12, 16, 16, 18, 24, 26, 28] and bin width is 10. Bins are:

Bin 1: 0,4 [0-10]

Bin2: 12,16,16,18 [10-20]

Bin3: 24,26,28 [20-30]

3.1.1.2. Equal Frequency Binning (Unsupervised). Because of the important disadvantage of equal width binning, another method was proposed. In equal-frequency binning, discretization is done by dividing the data into intervals that have the same number of instances [3]. This is a better method than equal width binning but it has also some disadvantages. For example, if all the instances belong to same class "C_1" in the "n_{th} interval" and all the instances except the first one belong to a different class "C_2" in the "$n+1_{st}$ interval", there is an obvious mistake in putting that instance into the "$n+1_{st}$ interval". So, equal frequency binning can also cause bad boundaries.

Again think of the same example dataset with attribute values [0, 4, 12, 16, 16, 18, 24, 26, 28] and bin density is 3. Bins are:

Bin 1: 0,4,12

Bin2: 16,16,18

Bin3: 24,26,28

3.1.1.3. Proportional K-Interval Discretization (PKID) (Unsupervised). This method is a version of equal frequency binning. Number of intervals are chosen based on the number of instances in the data. If the number of instances is N, there are square root of N intervals ($k = \sqrt{N}$) in this method. PKID method gives good results with Naive Bayes learning scheme [3].

The example dataset in Equal Frequency Binning is also an example for PKID, because instance number is 9 with attribute values [0, 4, 12, 16, 16, 18, 24, 26, 28]. This means there will be ($k = \sqrt{9} = 3$) bins. The bins are same as above:

Bin 1: 0,4,12

Bin2: 16,16,18

Bin3: 24,26,28

3.1.1.4. Entropy Based Discretization (Supervised). This method was proposed by Fayyad and Irani in 1993. This is a hierarchical method that uses entropy for recursively partitioning the continuous attribute values and Minimum Description Length (MDL) principle for stopping the splitting [25]. Each value of an attribute is a possible splitting point in this method.

If S is a set of instances, A is a continuous attribute, and T is a splitting point; T which maximizes the information gain is chosen. Information Gain:

$$IG(S,T) = H(S) - H(S,T) \tag{3.1}$$

where H(S,T) is the class information entropy, which is calculated by the formula:

$$H(S,T) = \frac{|S_1|}{S}H(S_1) + \frac{|S_2|}{S}H(S_2) \tag{3.2}$$

where S_1 and S_2 are instances that are separated by splitting point T as $A < T$ and $A \geq T$.

The process recursively continues until MDL stopping criterion given below is satisfied:

$$IG(S,T) < \delta \tag{3.3}$$

where δ is

$$\delta = \frac{log_2(n-1) + log_2(3^k - 2) - [kH(S) - k_1 H(S_1) - k_2 H(S_2))]}{n} \tag{3.4}$$

where k_i is the number of classes represented in the set S_i and n is a number of examples in S.

3.2. Dimensionality Reduction

Complexity of learning algorithms depends on the number of instances and the number of attributes that the training dataset contains. Reducing the complexity of training dataset yields better cost for space and time requirements of algorithms. Also representing the data with fewer dimensions without information loss brings out simple models. Simple models are more robust and may produce better knowledge extraction as well as providing better understanding. Thus, attribute selection should be applied before learning starts [2].

Some learning schemes have their own attribute selection capabilities by their nature. One example of these schemes is the decision tree. Decision trees, calculate the importance of attributes by information gain method and split the data into different regions. Attributes are used as splitting points according to their importance in decreasing order so that irrelevant attributes are not selected. Although this characteristic of decision trees seems enough for feature selection, it does not work as expected for all datasets. For example, in the deeper nodes of a decision tree, few instances can be separated in a region. This makes the learning algorithm get confused and it may select irrelevant features as better choices. These wrong choices affect the classification performance negatively [3].

The situation in decision trees is present for most of the machine learning algorithms which have their own feature selection mechanisms. Thus, performing feature selection before running the learning algorithms seems a good idea. Additionally, run time of the learning algorithms become shorter.

Manual selection approach is the best way of decreasing dimension if we have deep expertise about problem space, instances, and attributes. Otherwise, there exist two main methods for dimensionality reduction: feature selection and feature extraction. Feature selection is selecting a subset of features which have necessary information about data, whereas feature extraction is extracting new and smaller feature set by using all features. Principle Components Analysis (PCA) and Linear Discriminant Analysis (LDA) are the most known and widely used variations of feature extraction [2].

3.2.1. Feature Selection

3.2.1.1. Correlation Based Feature Selection. Correlation Based Feature Selection, also called as CFS, is a simple and fast method that evaluates the attribute subsets instead of evaluating attributes one by one. CFS chooses best subsets of attributes according to two criteria [3]:

- Individual attributes should be highly correlated with the class,
- Attributes of a subset should have a low inter - correlation with each other.

The heuristic given below contains these two criteria:

$$Merit_S = \frac{k\overline{r_{cf}}}{\sqrt{k(k+1)\overline{r_{ff}}}} \tag{3.5}$$

where $Merit_S$ is the heuristic "merit" of a feature subset S containing k features, $\overline{r_{cf}}$ is average feature-class correlation, $\overline{r_{ff}}$ is average feature-feature inter - correlation.

The numerator is the predictive ability of each feature; the denominator is the inter - correlation between features.

3.2.1.2. Consistency Based Feature Selection. Consistency Based Feature Selection, also called CONS, selects a subset of attributes by searching the subset space according to a consistency measure. It evaluates the subsets by looking for the attributes

associated with the same class. The consistency of full attribute set is the best and cannot be improved. The aim is finding the smallest subset with equal consistency to consistency of full attribute set [3].

Consistency measure is:

$$Consistency_S = 1 - \frac{\sum_{i=0}^{J} |D_i| - |M_i|}{N} \tag{3.6}$$

where S is an attribute set, J is the number of distinct combinations of attribute values for S, $|D_i|$ is the number of occurrences of the i_{th} attribute value combination, $|M_i|$ is the cardinality of the majority class for the i_{th} attribute value combination, N is the total number of instances in the dataset.

3.2.1.3. Information Gain Based Feature Selection. Information Gain Based Feature Selection, also called as IG, is a very popular and effective method for high dimensional datasets. It evaluates attributes individually by measuring their information gain with respect to the class [3].

If $C = c_1, ..., c_k$ is the classes of a dataset, the information gain of an attribute A_i

$$IG(A_i) = H(C) - H(C|A_i) \tag{3.7}$$

where $H(C) = -\sum_{i=1}^{k} p(c_i) log_2 p(c_i)$ and $H(C|A_i) = p(A_i) \sum_{i=1}^{k} p(c_i|A_i) log_2 p(c_i|A_i)$.

Information Gain is calculated for all attributes across all classes. Ranker method is used for selecting attribute subset.

3.2.1.4. Symmetrical Uncertainty Based Feature Selection. Symmetrical Uncertainty Based Feature Selection, also called as SU, evaluates attributes individually based on symmetrical uncertainty with respect to the class [3].

If $C = c_1, ..., c_k$ is the classes of a dataset, the symmetrical uncertainty of an attribute A_i

$$SU(C, A_i) = 2\frac{H(C) - H(C|A_i)}{H(C) + H(A_i)} \tag{3.8}$$

Symmetrical Uncertainty is calculated for all attributes across all classes. Ranker method is used for selecting attribute subset.

3.2.2. Ranking and Search Methods

A dataset with d features has 2^d possible subsets of features. In feature selection, the aim is to select a subset whose accuracy is maximum while the number of the features in the subset is minimum. A brute-force search is not achievable due to exponential complexity.

There exist two main approaches for subset search: forward selection and backward selection [3]. In forward selection, initial subset consists of zero feature. At each step, a feature is added to the subset whose addition decreases the error. The search continues until further addition does not decrease the error significantly. In backward selection, initial subset consists of all features. At each step, a feature is removed from the subset whose removal results in best error. The search continues until further removal does not improve the error.

Backward selection is more expensive than forward selection since error is calculated for subsets with more features. On the other hand, backward selection does not separate features which work well together. For example, two attributes which provide good knowledge as a couple can be useless separately and backward selection may remove them in different steps. If there are too many redundant features present in the dataset, forward selection would have better performance.

3.2.2.1. Best First Search Method. Best First Search is an extended version of traditional graph search algorithms whose node selection is based on a heuristic function. The heuristic is the function that is used by attribute subset evaluator. For correlation based feature selection, heuristic is correlation; for consistency based feature selection, heuristic is consistency.

Best First is a greedy search method that performs hill climbing with backtracking [3]. It can apply forward selection, backward selection, or search in both directions. Best First method also remembers all subsets evaluated and sorts them according to the performance measure for efficiency.

3.2.2.2. Ranker Method. Ranker Method is used with feature selection methods that evaluate attributes individually, such as information gain based feature selection method or symmetrical uncertainty based feature selection method. Ranker sorts attributes according to the metric used by attribute selection method [3]. For example, ranker method sorts attributes according to their information gain with respect to the class while being used with information gain based feature selection method. For selecting attributes, a threshold is set or attribute number is specified.

3.3. Classification

3.3.1. Decision Trees

Decision tree is one of the most known nonparametric machine learning methods. It uses divide and conquer approach to split the input space to sub regions. Then it creates a model depending on these regions.

A decision tree is a hierarchical structure. It consists a root node, internal decision nodes, leaf nodes, and branches. Internal decision nodes label the branches according to their test function. Leaf nodes correspond to labeled instances.

Decision tree is an easy method to comprehend. Its representation can be exported to IF-THEN rules that can be understood by anyone. Decision tree algorithms run faster regarding other learning algorithms because their hierarchical structure lets elimination of some decision nodes. Instead of learning error-free model from decision tree, it is important to find the model with the simplest tree so that performance on test data can be improved [2].

3.3.1.1. How Decision Tree Algorithms Work?. Decision tree algorithms map input features to tree nodes. Features are ordered by their importance via information gain method. The most important feature is selected as the root node. After selecting the root node, the algorithm begins to run recursively. For each node, new branches are created for each value of corresponding features. Then training instances are divided into these branches according to their value for that feature. For each branch a new feature is selected as a node, and recursive algorithm continues to run on these nodes. The algorithm runs until all of the leaf nodes have instances with the same label.

Decision tree approach can be explained in a mathematical way. In each node, input dimensions of the instances allocated for that node, are used for creating the tree. If the input dimension x_j has n possible value, this means the node is divided into n branches. If x_j is a numerical comparison test for building the tree;

$$f_m(x) : x_j > w_{m0} \tag{3.9}$$

where w_{m0} is the threshold value. Decision node splits the space into two: $L_m = x|x_j > w_{m0}$ and $R_m = x|x_j < w_{m0}$. This is called binary split. For a successful decision tree, splits should be orthogonal to each other.

This process is repeated recursively by determining another feature until reaching the leaves (instances belong to the same class under a node). Below an example of the process is illustrated in Figure 3.1.

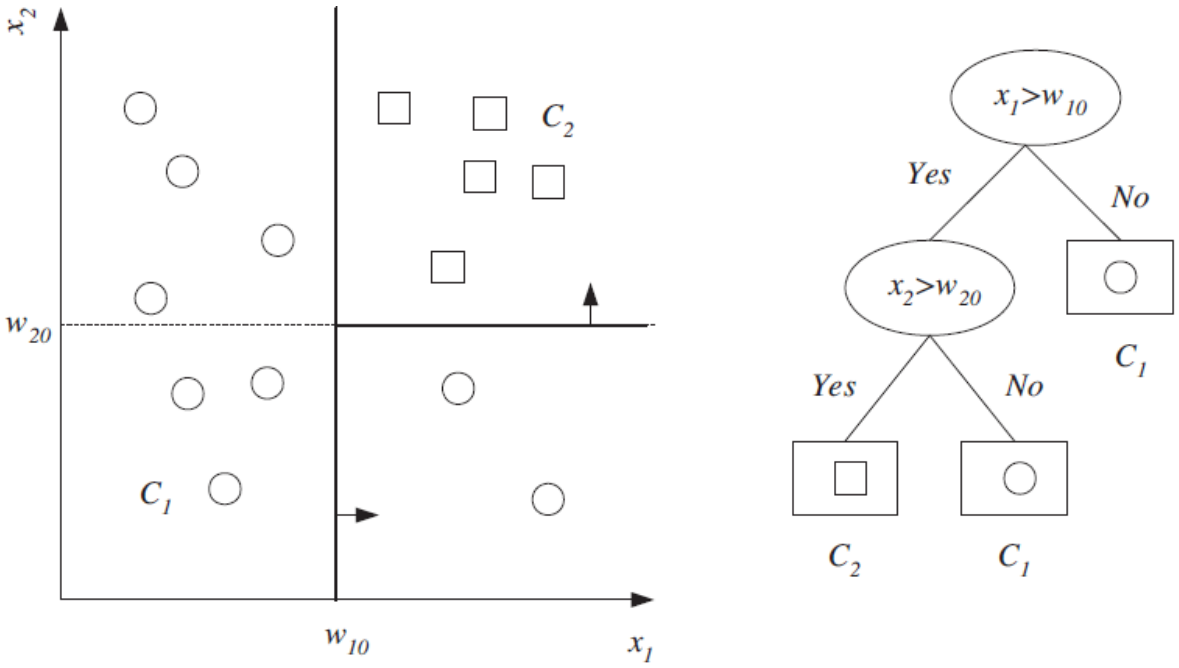

Figure 3.1. Decision tree construction process [2].

For a decision tree, the aim is building the smallest tree; thus, it is very important to select attributes. Attributes that produce the purest daughter nodes are chosen based on an impurity measure.

If "N" is the number of all training instances, "N_m" is the number of instances reach "node m", "N_{m_i}" is the number of instances reach "node m" and belong to class "C_i", given that an instance reaches "node m", the estimate for the probability of class "C_i" is;

$$P(C_i|x,m) = p_{m_i} = N_{m_i}/N_m \tag{3.10}$$

If $p_{m_i} = 0$, this means none of the instances reach node m belongs to class C_i. If $p_{m_i} = 1$, this means all of the instances reach node m belong to class C_i. In both cases, "node m" is pure.

So what is the impurity measure? For a two class problem where $p_1 = p$ and $p_2 = 1 - p$, the impurity measure is a function that satisfies the following properties;

- $f(1/2, 1/2) \geq f(p, 1-p)$ for any p in [0, 1]
- $f(0, 1) = f(1, 0) = 0$
- $f(p, 1-p)$ is increasing in p on [0, 1/2] and decreasing in p on [1/2, 1]

Entropy, Gini index and Misclassification error functions satisfy these properties and can be used as impurity measure [2]. For a two class problem;

$$\text{Gini index:} f(p, 1-p) = 2 * p * (1-p) \tag{3.11}$$

$$\text{Misclassification error:} f(p, 1-p) = 1 - max(p, 1-p) \tag{3.12}$$

$$\text{Entropy:} f(p, 1-p) = -p * \log_2 p - (1-p) * \log_2(1-p) \tag{3.13}$$

Among these, entropy is the most used impurity measure. With information theoretical perspective, it shows the minimum number of bits needed for encoding the class label of an instance. Entropy is "0", if $p_1 = 0$ and $p_2 = 1$ (all instances belong to C_2) or $p_1 = 1$ and $p_2 = 0$ (all instances belong to C_1). That is the best case and there is no need to send any bits in this case. Entropy is "1", if $p_1 = 0.5$ and $p_2 = 0.5$ (instances can belong to C_1 or C_2 with equal probability). That is the worst case and it is needed to send one bit to specify the class label.

Briefly, to construct a decision tree, for each branch the impurity measure is calculated on branch instances and the attribute with maximum information gain (minimum entropy) is selected as split attribute. Thus, in each step, splitting the instances causes maximum possible decrease in impurity.

3.3.1.2. Decision Tree Construction with an Example Dataset. An example dataset is used for explaining tree construction. It can be seen on Table 3.1.

Table 3.1. Example dataset for decision tree construction [3].

protocol-type	**service**	**flag**	**logged-in**	**label**
udp	http	SF	no	normal
udp	http	SF	yes	normal
icmp	http	SF	no	attack
tcp	ecr-i	SF	no	attack
tcp	smtp	S0	no	attack
tcp	smtp	S0	yes	normal
icmp	smtp	S0	yes	attack
udp	ecr-i	SF	no	normal
udp	smtp	S0	no	attack
tcp	ecr-i	S0	no	attack
udp	ecr-i	S0	yes	attack
icmp	ecr-i	SF	yes	attack
icmp	http	S0	no	attack
tcp	ecr-i	SF	yes	normal

In this case the attributes are protocol-type, service, flag, and logged-in. The class labels are normal or attack. Before any calculations for tree construction, we can see that there are fourteen instances in example data and they have nine "attack" and five "normal" class labels. Information value of training examples is;

$$Info([9,5]) = -9/14 * \log_2(9/14) - 5/14 * \log_2(5/14) = 0.940 bits$$

Firstly an attribute among four of them should be selected as the root node. The attribute that has the minimum entropy and maximum information gain about data

is the best for the beginning. The alternatives are given in Figure 3.2.

The calculations for the first attribute "protocol-type" are given below. This attribute has three different values. Protocol-type can be udp, icmp and tcp according to the data. These values represent the branches.

For the "udp" branch, there are five instances and these instances have two "attack" and three "normal" class labels.

$$Info([2,3]) = -2/5 * \log_2(2/5) - 3/5 * \log_2(3/5) = 0.971 bits$$

For the "icmp" branch, there are four instances and these instances have four "attack" and zero "normal" class labels.

$$Info([4,0]) = -4/4 * \log_2(4/4) - 0/4 * \log_2(0/4) = 0 bits$$

For the "tcp" branch, there are five instances and these instances have three "attack" and two "normal" class labels.

$$Info([3,2]) = -3/5 * \log_2(3/5) - 2/5 * \log_2(2/5) = 0.971 bits$$

The average information of these branches is;

$$Info([2,3],[4,0],[3,2]) = 5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971 = 0.693 bits$$

At the very beginning, we calculated the information value of training examples, it was 0.940 bits. So, the information gain is;

$$gain(protocol{-}type) = info([9,5]) - info([2,3],[4,0],[3,2]) = 0.940 - 0.693 = 0.247 bits$$

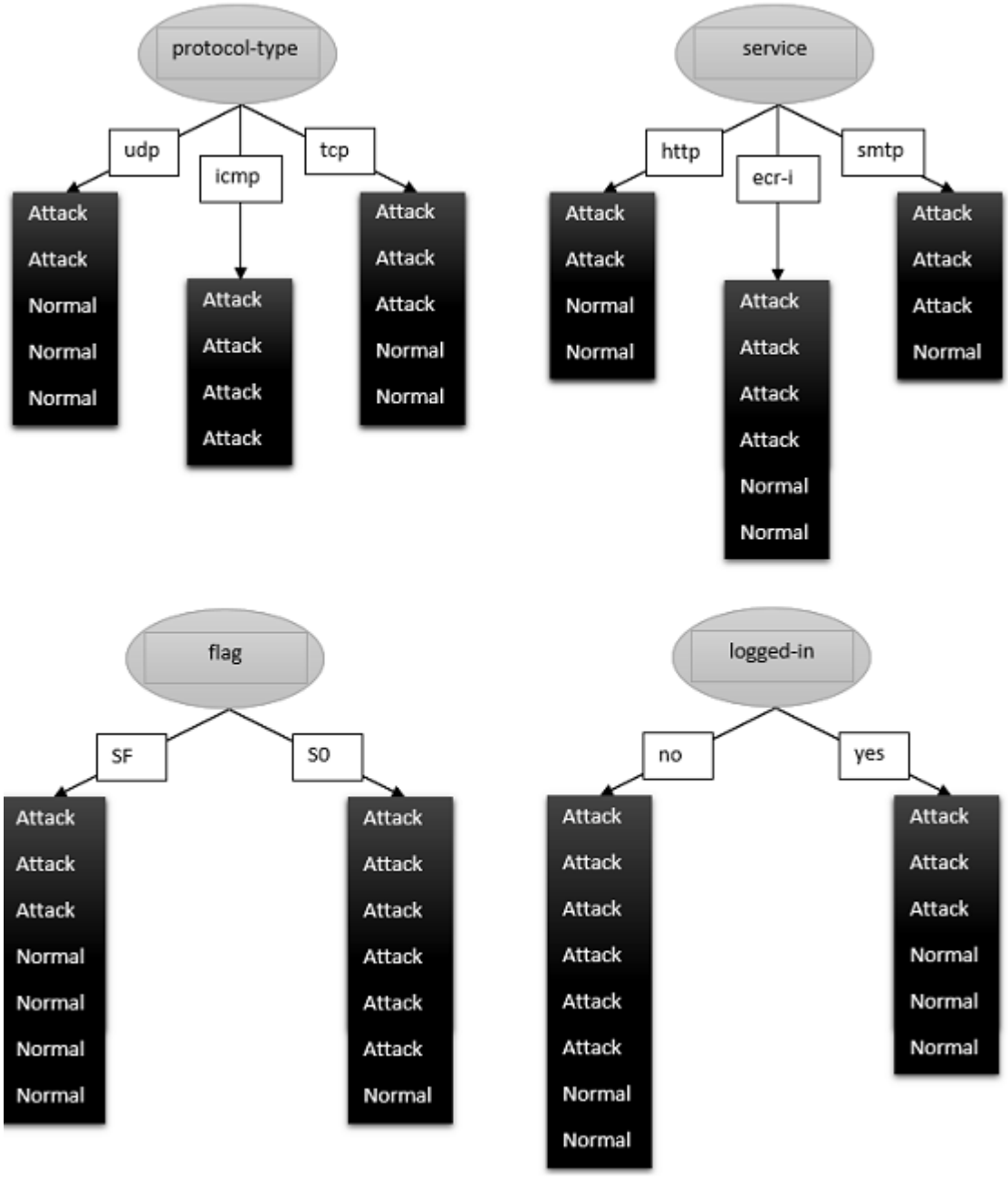

Figure 3.2. Selecting the first attribute [3].

If the same calculations are done for the other attributes (service, flag, logged-in), information gain of all attributes can be found.

gain (protocol-type) = 0.247 bits

gain (service) = 0.029 bits

gain (flag) = 0.152 bits

gain (logged-in) = 0.048 bits

The "protocol-type" attribute should be selected as the root node (for the first splitting) because it has the maximum information gain.

Then the process continues recursively. Now the internal nodes should be selected for all branches. If we start from "udp" branch, we can choose one of three remaining attributes (service, flag, logged-in) as internal node. The alternatives are given in Figure 3.3.

Before any calculations for sub tree construction, we can see that there are five instances in "udp" branch and they have two "attack" and three "normal" class labels. Information value of training examples come to this branch is;

$$Info([2,3]) = -2/5 * \log_2(2/5) - 3/5 * \log_2(3/5) = 0.971 bits$$

The calculations for the first attribute "service" are given below. This attribute has three different values. Service can be http, ecr-i, or smtp according to the data. These values represent the branches.

For the "http" branch, there are two instances and these instances have zero

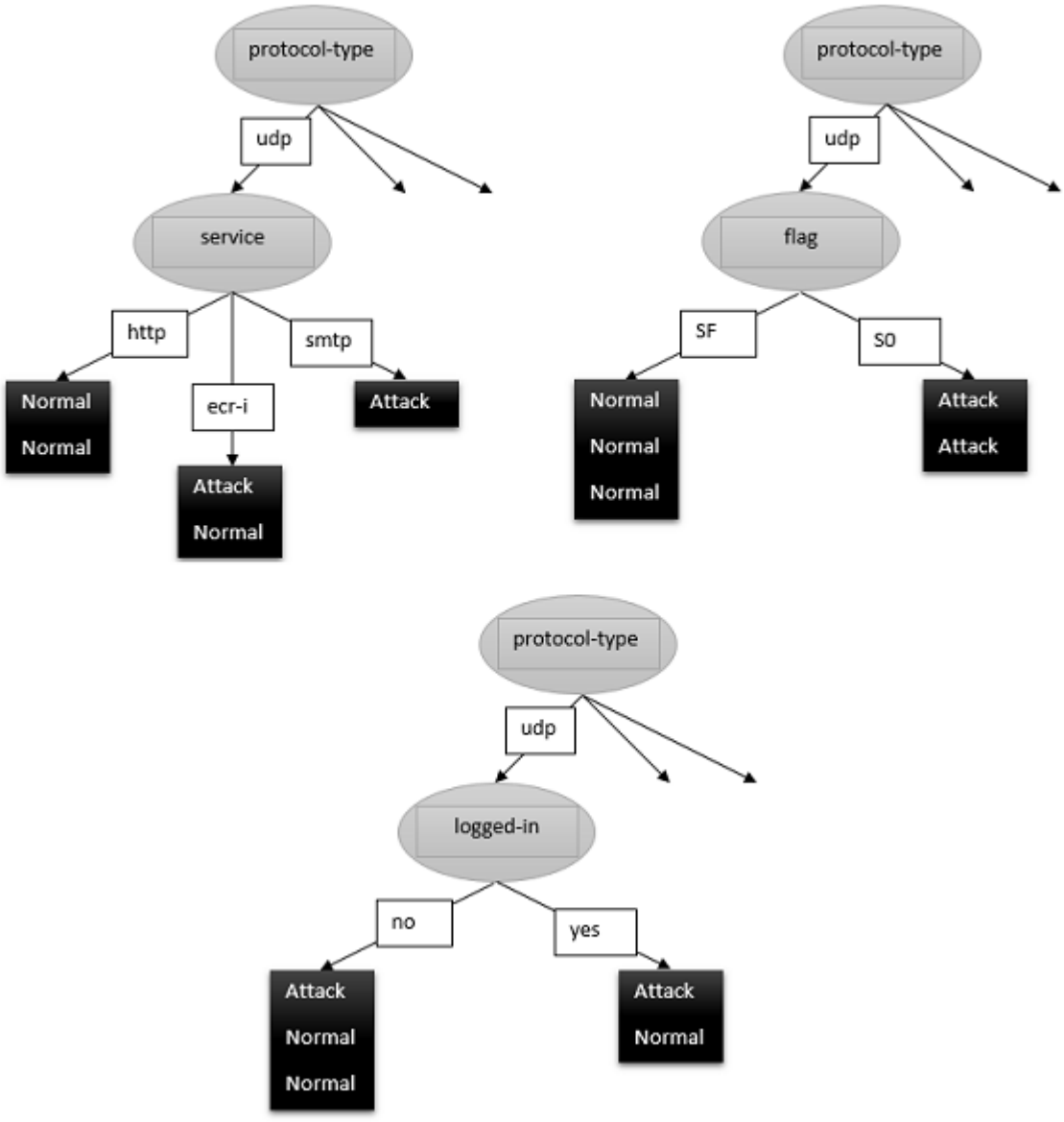

Figure 3.3. Selecting the second attribute [3].

"attack" and two "normal" class labels.

$$Info([0,2]) = -0/2 * \log_2(0/2) - 2/2 * \log_2(2/2) = 0bits$$

For the "ecr-i" branch, there are two instances and these instances have one "attack" and one "normal" class labels.

$$Info([1,1]) = -1/2 * \log_2(1/2) - 1/2 * \log_2(1/2) = 1bits$$

For the "smtp" branch, there is one instance and this instance have one "attack" class label.

$$Info([1,0]) = -1/1 * \log_2(1/1) - 0/1 * \log_2(0/1) = 0bits$$

The average information of these branches is;

$$Info([0,2],[1,1],[1,0]) = 2/5 * 0 + 2/5 * 1 + 1/5 * 0 = 0.4bits$$

At the beginning, we calculated the information value of training examples come to udp branch, it was 0.971 bits. So, the information gain is;

$$gain(service) = info([2,3]) - info([0,2],[1,1],[1,0]) = 0.971 - 0.4 = 0.571bits$$

If the same calculations are done for the other attributes (flag, logged-in), information gain of all attributes can be found.

gain (service) = 0.571 bits

gain (flag) = 0.971 bits

gain (logged-in) = 0.020 bits

The “flag” attribute should be selected as internal node for “udp” branch because it has the maximum information gain.

The process continues with the other branches and nodes until all the instances that come to branches belong to the same class. Decision tree of the data is given below in Figure 3.4.

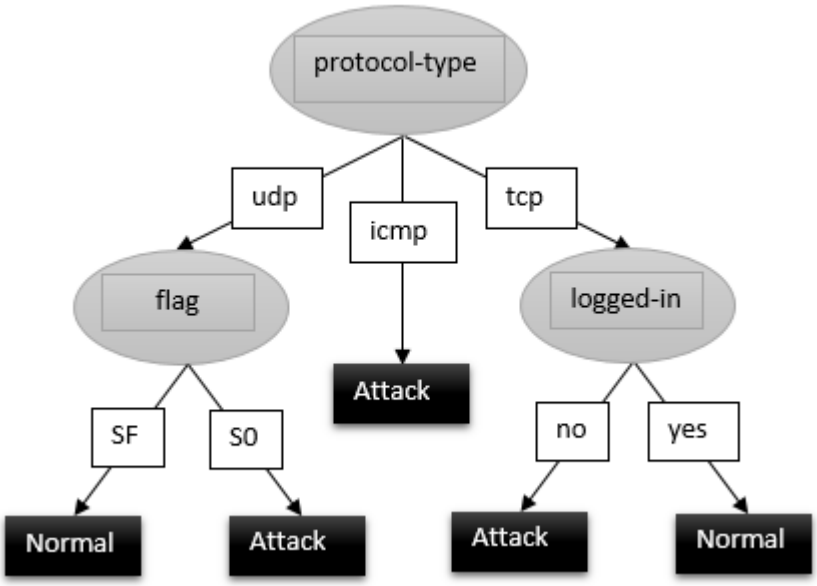

Figure 3.4. Final decision tree for example dataset [3].

3.3.1.3. Pruning. For our example, at the end the purest leaves are achieved. But if there is noise in data, the tree can grow very large until it has the purest leaves. The solution for this is pruning.

Stopping the tree construction early on before it is full is pre-pruning. That seems a good idea because of avoiding all the work of developing sub-trees. A threshold for purity can be determined in advance. Tree construction continues until reaching this threshold. If this threshold is small, the variance is high and tree is large. If this

threshold is large, the variance is small and tree is small. This threshold is determined according to the costs of misclassification, memory and computation.

Another solution for fully expanded and large decision trees is post-pruning and in practice it works better than pre-pruning. The sub-trees that cause over fitting are found and pruned. There are two ways of post-pruning: sub-tree replacement and sub-tree raising. In sub-tree replacement, the idea is replacing some of the sub-trees with single leaves. If a leaf does not perform worse than sub-tree on the pruning set, the sub-tree is pruned and the leaf is kept. This will certainly cause a decrease in accuracy of the training set. But pruning set is different from training set or validation set. In sub-tree raising, internal nodes are replaced with their daughter nodes. The replaced internal node is pruned and daughter node is kept.

Post-pruning is shown in Figure 3.5. Here "node B" is replaced with its daughter "node C". It is needed to reclassify the instances at the nodes marked 4 and 5. This is why at the end "node C" has different daughters from the beginning.

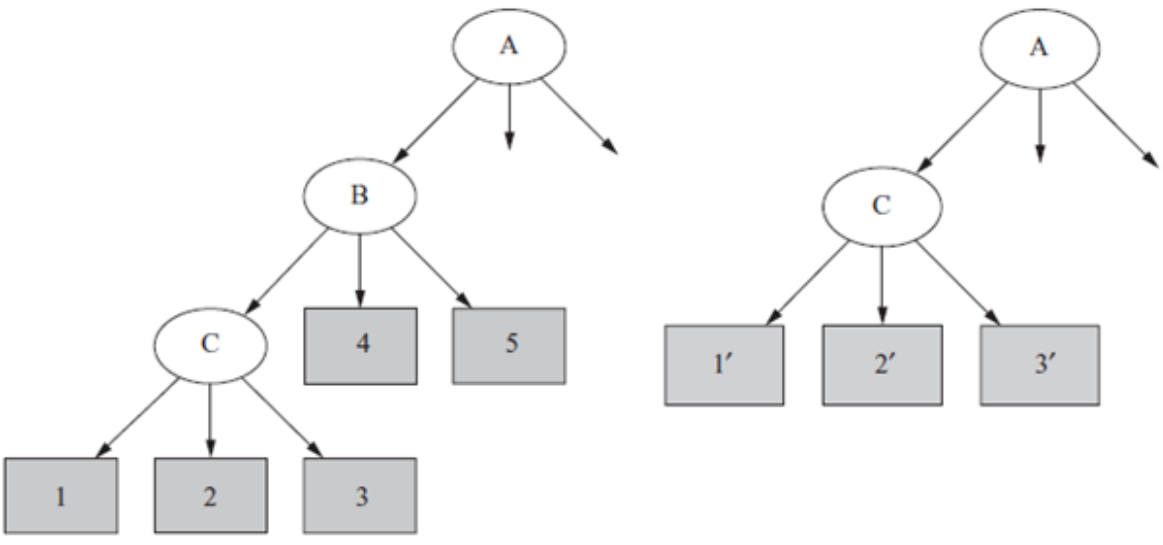

Figure 3.5. Post pruning [3].

3.4. Bootstrap Aggregating

Bootstrap aggregating is a machine learning technique that provides stability and better accuracy to classification algorithms. It is usually used with decision trees, but it is also available to use with different methods. The goal is creating several different decision structures. This is achieved by having several different training sets of the same size. Bootstrap Aggregating is also called as bagging.

Think of a training set with size of n instances. Bagging creates m new training sets with size of n' instances by sampling with replacement. Sampling with replacement means choosing a sample from the set and then putting it back to the set before taking the next sample. So, there is always the possibility of choosing the same samples. This means instances are probably duplicated in new generated training sets. The number of new training sets "m" is determined by us. Also, we choose the size of new sets n' and it can be equal to n.

After producing new training sets, a model is built for each set. In our case, a decision tree is built for each training set. Decision trees predict a label for each test instance. These predictions are combined by voting. For example, if six of the decision trees predict that "instance n" belongs to the "class c" and four of the decision trees predict that "instance n" belongs to the "class b", final decision will be "instance n" belongs to the "class c". A diagram that explains bagging can be seen in Figure 3.6.

This method is very suitable for unstable learning schemes. Unstable schemes are the ones that change significantly with a small change in training set. Decision tree is an example of unstable schemes. For decision trees, which are already unstable, better performance is often achieved by abolishing pruning, which makes them more unstable [3].

If a learning algorithm is very sensitive to small variations in training data, it is said to be an unstable learning algorithm [5]. $X \subset \mathbb{R}$ is an input space and $Y \subset \mathbb{R}$ is an output space. z_i is the input-output pair $z_i = (x_i, y_i)$ $i = 1, ..., m$ and $Z =$

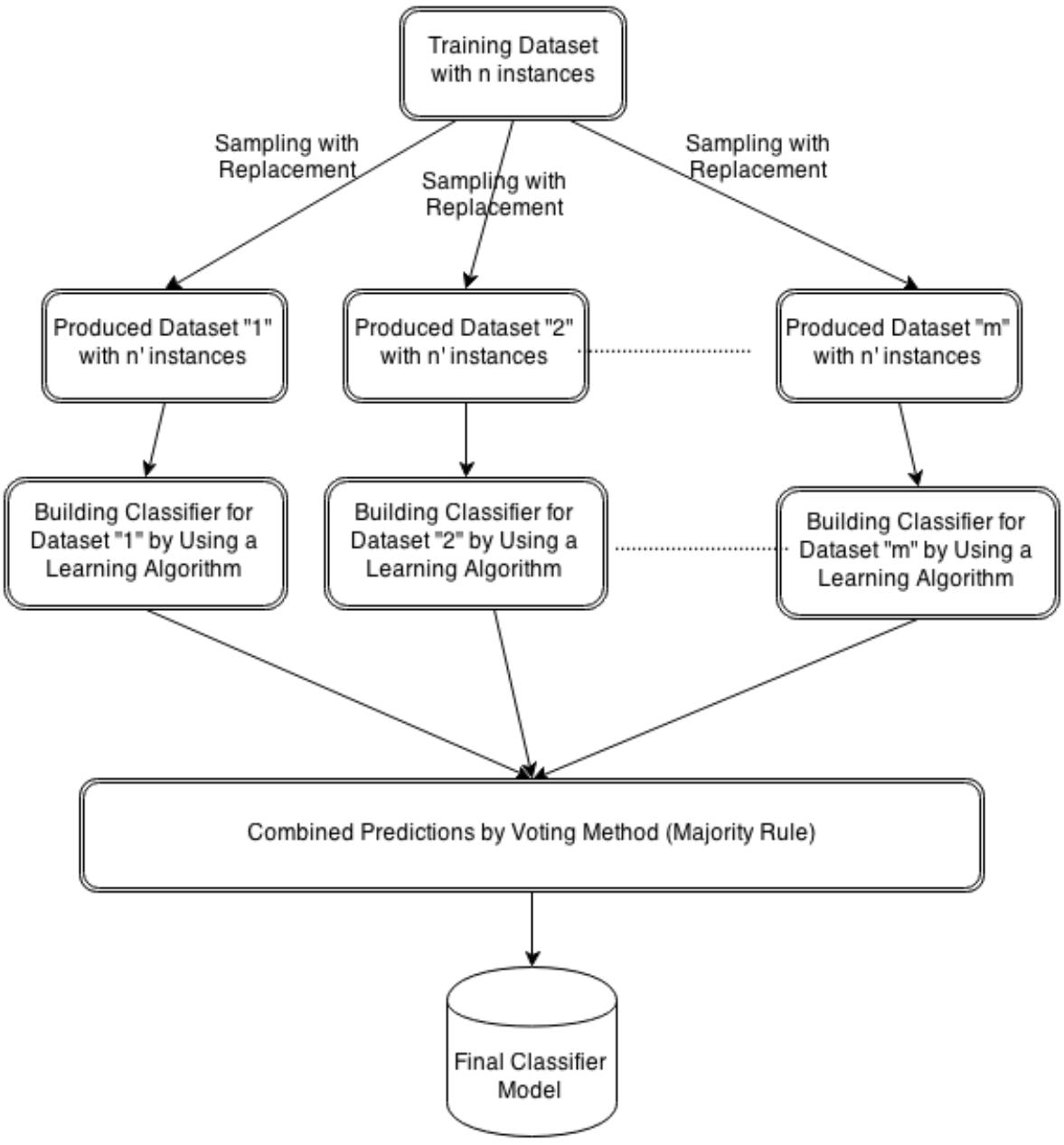

Figure 3.6. Bagging process [4].

$X \times Y$ is independent and identically distributed according to an unknown probability distribution. Then, the training set S is

$$S = \{z_1 = (x_1, y_1), ..., z_m = (x_m, y_m)\}$$

By removing the i_{th} element;

$$S^i = \{z_1, ..., z_{i-1}, z_{i+1}, ..., z_m\}$$

A learning algorithm is a function f which maps a learning set S onto a function f_S from X to Y. $V(y, f(x))$ is the loss function. The choice of this function changes according to different learning algorithms. It is a function that maps events onto a real number which represents the cost related to the event.

A classification algorithm has classification stability β if

$$\forall i \in \{1, ..., m\}, \forall S, \forall (x, y) : |V(f_S(x), y) - V(f_{S^i}(x), y)| \leqslant \beta \tag{3.14}$$

which means the cost of a learning algorithm should not change more than β when the algorithm is trained by a learning set which does not include i_{th} training instance [26].

In figure 3.7, decision tree instability can be seen. On the left, a C4.5 decision tree trained by 106 instances is shown. One more instance is added to the same set. On the right, a C4.5 decision tree trained by 107 instances is shown. Adding a single instance nearly doubles the number of decision nodes.

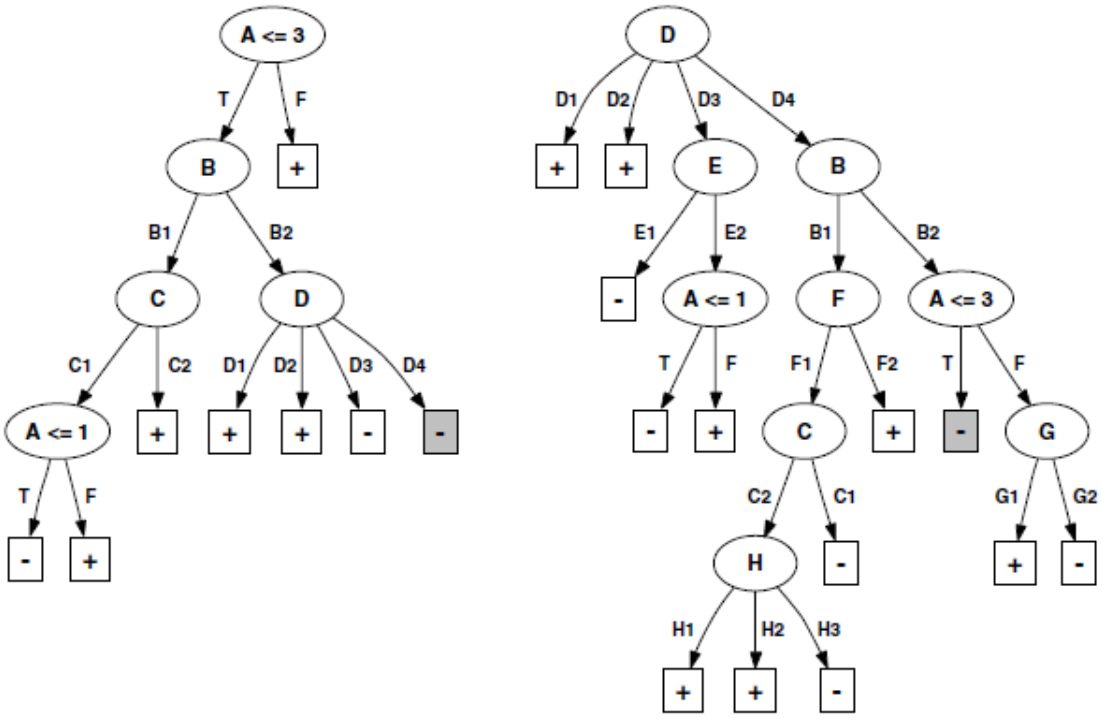

Figure 3.7. Decision tree instability example [5].

4. METHODOLOGY FOR BUILDING AN INTRUSION DETECTION SYSTEM

A decision tree based intrusion detection system is proposed in this thesis. KDD'99 intrusion detection evaluation training and test datasets and WEKA tool that contains many machine learning algorithms are used for implementing the proposed intrusion detection system. In this chapter KDD'99 Datasets and WEKA tool are analyzed. Additionally, detailed structure of proposed IDS is given in the last section of chapter.

4.1. KDD'99 Intrusion Detection Evaluation Dataset

KDD'99 Intrusion Detection Evaluation Dataset is the dataset used for The Third International Knowledge Discovery and Data Mining Tools Competition. The goal of the contest was creating an intrusion detection system that classifies the connections and separates the normal traffic from attack traffic or intrusions.

The KDD'99 data is a version of 1998 DARPA Intrusion Detection Evaluation Program Dataset which was generated by MIT Lincoln Labs. A military network environment simulating U.S. Air Force LAN was set up. TCP dump data from nine weeks of network traffic was recorded. Seven weeks of the data were used as training set and two weeks of the data was used as test set. There are about five million connections in training set and two million connections in test set [6].

4.1.1. KDD'99 Class Labels

Each connection has a label. Label can be either normal or one of the various attack types. There are 22 attack types in training dataset and 17 additional attack types in test dataset. These attacks are basically classified into four groups:

- Denial-of-Service (DOS): Accessing the network without having official permis-

sion or approval for the purpose of interrupting or blocking the normal functioning.

- Remote-to-Local (R2L): Obtaining local user privileges by an unprivileged remote user.
- User-to-Root (U2R): Obtaining superuser or administrator privileges by an unprivileged local user.
- Surveillance or Probing (Probe): Accessing the network without having official permission or approval for the purpose of searching configurations, network topologies or vulnerabilities.

39 specified attack types can be seen on Table 4.1 as divided into four main groups of attacks.

Table 4.1. Training attack types and additional test attack types [6].

Attack-Type	**Training Set**	**Test Set (Additional Attacks)**
DOS	back, land, neptune, pod, smurf,teardrop	apache2, mailbomb, processtable, udpstorm
U2R	bufferoverflow, loadmodule, perl, rootkit	ps, sqlattack, xterm
R2L	ftpwrite, guesspasswd, imap,multihop, phf, spy, warezclient,warezmaster	httptunnel, named, sendmail, snmpgetattack, snmpguess, xlock, xsnoop, worm
Probe	ipsweep, nmap, portsweep, satan	mscan, saint

In this study, 10 percent of labelled KDD'99 training and test dataset is used as it was in The Third International Knowledge Discovery and Data Mining Tools Competition. There are 494021 instances in the training dataset and 311029 instances in the test dataset. The distribution of class labels of these instances can be seen on the Table 4.2 and Table 4.3.

Table 4.2. Training set class distribution [8].

Class Label	**Number of Instance**	**Percentage**
DOS	391458	79.24
U2R	52	0.01
R2L	1126	0.23
Probe	4107	0.83
Normal	97278	19.69
Total	494021	100

Table 4.3. Testing set class distribution [8].

Class Label	**Number of Instance**	**Percentage**
DOS	229853	73.90
U2R	228	0.07
R2L	16189	5.20
Probe	4166	1.34
Normal	60593	19.48
Total	311029	100

4.1.2. KDD'99 Features

There are 41 features of each connection in KDD'99 dataset. These features can be grouped into four main categories [27]:

- Basic Features: The features that exist in packet headers.
- Content Features: Some kinds of attacks do not generate sequential patterns because they involve only one connection and they are hidden in the payload of the packet. Content features are the features that are based on domain knowledge and used for investigating suspicious behavior in the payload.
- Time-based Traffic Features: There are two types of time-based traffic features. These features inspect only the connections in the past two seconds. "Same host" features analyzes the connections that have the same destination host. "Same service" features analyzes the connections that have the same service.
- Host-based Traffic Features: Some attacks occur in large time intervals. Thus, it is needed to examine the traffic in a window of a certain number of connections instead of a time window. Host-based features are constructed using a window of 100 connections to the same host.

A list of features that are divided into four main groups can be examined on the Table 4.4 and Table 4.5.

4.2. Waikato Environment for Knowledge Analysis

Waikato Environment for Knowledge Analysis, or WEKA for short, is a collection of machine learning algorithms for data mining tasks [3]. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. The algorithms can either be applied directly to a dataset or called from Java code.

WEKA is an outstanding source that fulfills almost all needs and expectations about data mining. It not only includes implementations of straightforward techniques,

Table 4.4. List of features [6].

Feature Group	**Feature Name**	**Type**
Basic Features	duration	continuous
	protocol-type	discrete
	service	discrete
	src-bytes	continuous
	dst-bytes	continuous
	flag	discrete
	land	discrete
	wrong-fragment	continuous
	urgent	continuous
Content Features	hot	continuous
	num-failed-logins	continuous
	logged-in	discrete
	num-compromised	continuous
	root-shell	discrete
	su-attempted	discrete
	num-root	continuous
	num-file-creations	continuous
	num-shells	continuous
	num-access-files	continuous
	num-outbound-cmds	continuous
	is-hot-login	discrete
	is-guest-login	discrete

Table 4.5. List of features cont. [6].

Feature Group	**Feature Name**	**Type**
Time-Based Traffic Features	count	continuous
	serror-rate	continuous
	rerror-rate	continuous
	same-srv-rate	continuous
	diff-srv-rate	continuous
	srv-count	continuous
	srv-serror-rate	continuous
	srv-rerror-rate	continuous
	srv-diff-host-rate	continuous
Host-Based Traffic Features	dst-host-count	continuous
	dst-host-srv-count	continuous
	dst-host-same-srv-rate	continuous
	dst-host-diff-srv-rate	continuous
	dst-host-same-src-port-rate	continuous
	dst-host-srv-diff-host-rate	continuous
	dst-host-serror-rate	continuous
	dst-host-srv-serror-rate	continuous
	dst-host-rerror-rate	continuous
	dst-host-srv-rerror-rate	continuous

but also offers implementations of advanced learning schemes. It is clear and easy to understand and it is likewise useful for research or practical data mining.

4.2.1. Attribute-Relation File Format

WEKA uses Attribute-Relation File Format (ARFF) for reading datasets. For demonstrating and explaining ARFF file format, a very shortened version of KDD'99 file is given as an example in Figure 4.1.

```
@relation train

@attribute duration numeric
@attribute protocol_type {tcp,udp,icmp}
@attribute service {...,domain_u,...,http,...,telnet,...}
@attribute flag {OTH,REJ,RSTO,RSTOS0,RSTR,S0,S1,S2,S3,SF,SH}
@attribute src_bytes numeric
@attribute dst_bytes numeric
...
...
...
@attribute dst_host_rerror_rate numeric
@attribute dst_host_srv_rerror_rate numeric
@attribute label {dos,u2r,r2l,probe,normal}

@data
0,tcp,http,SF,181,5450,.....,0.00,0.00,normal
0,tcp,http,SF,239,486,.....,0.00,0.00,normal
184,tcp,telnet,SF,1511,2957,.....,0.00,0.00,u2r
305,tcp,telnet,SF,1735,2766,.....,0.00,0.00,u2r
23,tcp,telnet,SF,104,276,.....,0.00,0.00,r2l
0,tcp,telnet,RSTO,125,179,.....,0.50,0.50,r2l
0,tcp,http,SF,54540,8314,.....,0.00,0.00,dos
0,tcp,http,SF,54540,8314,.....,0.00,0.00,dos
0,udp,domain_u,SF,1,0,.....,0.00,0.00,probe
3,tcp,telnet,SF,54,114,.....,0.01,0.00,probe
```

Figure 4.1. KDD'99 in Attribute-Relation File Format [6].

ARFF files have two different parts. These are header information part and data information part [3].

- Header information part: In header information part, there are relation declara-

tion and attribute declarations.

(i) The relation name is the first line of the ARFF file. The format is **@relation "relation-name"** where "relation-name" is a string.

(ii) Relation declaration is followed by the attribute declarations. Each feature is represented by its own @attribute statement. The format is **@attribute "attribute-name" "datatype"** where "attribute-name" must start with an alphabetic character and "datatype" can be any of numeric, "nominal-specification", string, or "date".

Numeric attributes can be real or integer numbers.

Nominal values are defined by providing a "nominal-specification" listing the possible values: {"nominal-name1", "nominal-name2", "nominal-name3", ...}.

String attributes allow us to create attributes containing arbitrary textual values.

Date attributes allow us to create attributes containing dates with specific date formats.

- Data information part: In data information part, there are data declaration and instance lines.

(i) The data declaration is a single line indicates the start of the data information part. The format is @data.

(ii) The instance lines represent instances and their attribute values. Attribute values are in the same order with the header information part and separated with commas.

4.2.2. WEKA Explorer

WEKA has a graphical interface called Explorer. It provides help to use all algorithms easily. From explorer, it is very simple to reach data pre-processing, classification, regression, clustering, association rules, and visualization tools. It has menus stated clearly and in detail, leaving no room for confusion. Every tool has illustrative and brief explanations. Besides, sensible default values for all tools are selected to

simplify users' tasks [3].

WEKA also has two other interfaces: Experimenter and KnowledgeFlow. There is no need to mention these two interfaces because in this work, only Explorer interface has been used.

In the Figure 4.2 WEKA Gui Chooser can be observed. Furthermore, WEKA Explorer with KDD'99 test dataset can be seen in the Figure 4.3.

Figure 4.2. WEKA Gui Chooser [7].

4.3. The Proposed Intrusion Detection System Architecture

In our work, we combined bootstrap aggregating, discretization, feature selection, and classification methods. We assembled many different machine learning algorithms and proposed a decision tree based intrusion detection system.

The method has two phases: training phase and testing phase.

- Training phase consists of two parts: Preprocessing and learning.
  (i) In preprocessing part, we apply bootstrap aggregating, discretization, and feature selection sequentially.

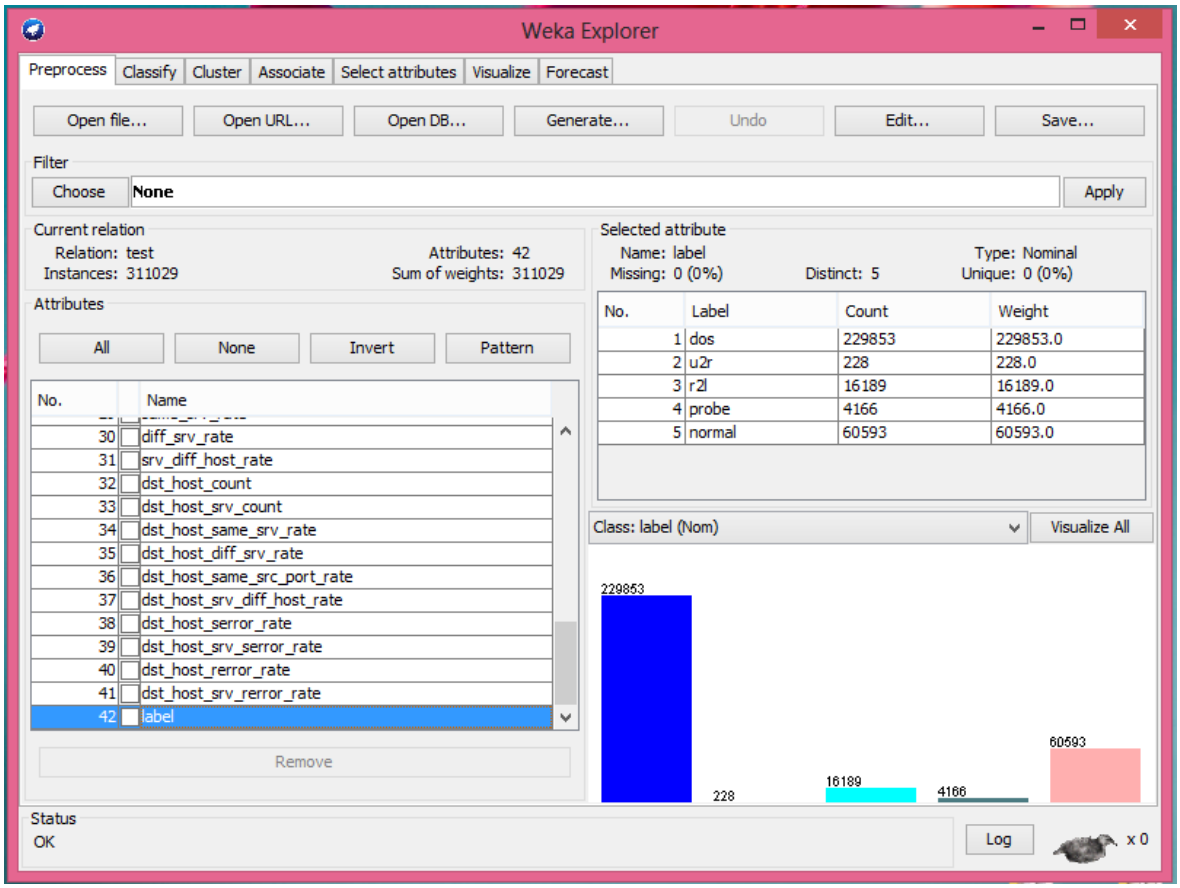

Figure 4.3. WEKA Explorer with KDD'99 test dataset [7].

We take the KDD'99 training set and put it into bagging block. Bagging block creates 10 different training sets sampled with replacement from the original training set. The size of new training sets is specified by us according to the experiments. All of them are equal sized and size can be any percentage of the training set. If the size is too big, there are too many duplicated instances in new training datasets. Usually, 10%-20% of the original dataset gives better results.

After bagging, 10 new training sets enter the discretization block. We use four different discretization methods. Equal Width Binning (EWD), Equal Frequency Binning (EFD), Propotional K Interval Discretization (PKID), and Entropy Based Discretization (EBD). In this block, numeric attributes are discretized for better results.

Subsequently, filtered and discretized training sets enter the feature selection blocks. We use four different feature selection methods. Correlation Based Feature Selection (CFS), Consistency Based Feature Selection (CONS), Information-Gain Based Feature Selection (IG), and Symmetrical Uncertainty Based Feature Selection (SU). Best First Search is used with Correlation Based Feature Selection Evaluator and Consistency Based Feature Selection Evaluator. Ranker Method is used with Information Gain Based Feature Selection Evaluator and Symmetrical Uncertainty Based Feature Selection Evaluator. Each method chooses different attribute subsets.

(ii) In learning part, we use J48 decision tree algorithm for classification. For each preprocessed training set, a decision tree is built by using selected attributes. The details of building a decision tree were explained in the theoretical knowledge chapter. Experiments are performed both pruned and unpruned trees. Unpruned trees usually give better results because bagging performs better on unstable algorithms. The predictions that are achieved by each decision tree are evaluated by voting method. The final decision tree is built. The output of learning part is our "model".

- In the testing phase, we use KDD'99 test dataset. Firstly, test dataset is preprocessed. It is discretized by using the intervals that are generated in the training

phase. Then, without performing feature selection algorithms for test dataset again, the features selected in training phase are used. After that, preprocessed test dataset enters the model block. Each instance is evaluated according to attribute values. Decision tree is tracked from root node to leaves and predictions are done for each instance.

In Figure 4.4, proposed method can be analyzed.

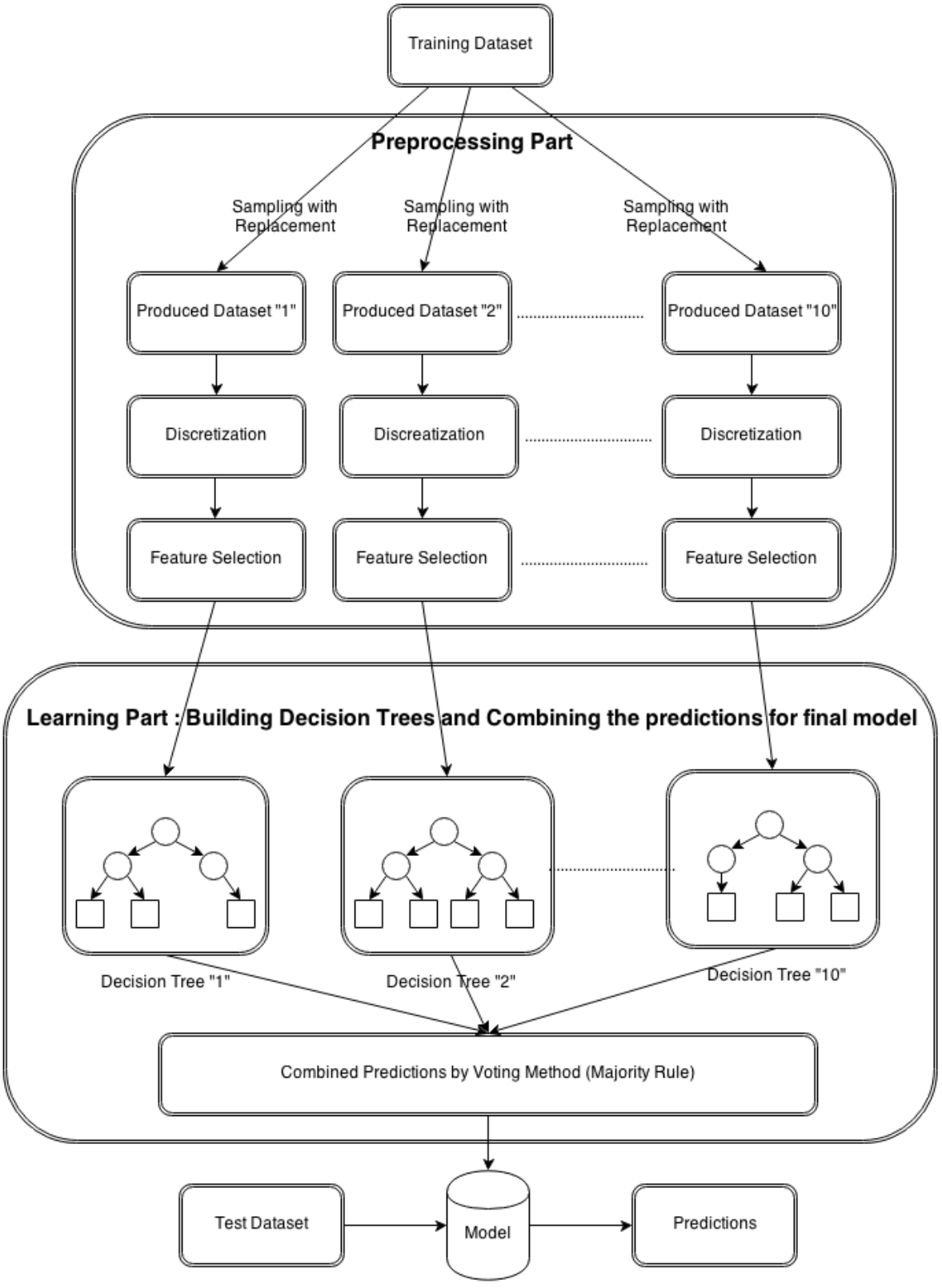

Figure 4.4. Proposed method.

5. NUMERICAL EVALUATIONS

Experimental results of various method combinations are evaluated according to the different performance metrics. In this chapter, performance metrics are explained. After that, results are given in illustrative charts. These results are compared with other works that are mentioned in Chapter 2.

5.1. Performance Metrics

Performance metrics measure the ability of the classifier for predicting the classes correctly. There are different performance metrics such as accuracy, precision, recall, f-measure, and area under receiver operating characteristic (AUC). Also cost is a performance metric that is used in KDD'99 competition as rating criteria.

For understanding performance metrics, structure of confusion matrix should be figured out first. Table 5.1 shows the confusion matrix of two class classifier.

Table 5.1. Confusion matrix of two class classifier.

Confusion Matrix		**Predicted Class**	
		Positive	Negative
Actual Class	Positive	TP-True Positive	FN-False Negative
	Negative	FP-False Positive	TN-True Negative

Accuracy is the most common performance metric but, for imbalanced data, accuracy can cause wrong impressions. If the instances belong to the dominant class are predicted correctly, accuracy will be high. But if the instances belong to the minor class are predicted wrongly, the classifier will not be successful in reality. Therefore, other metrics should be used for imbalanced datasets. Accuracy, recall, precision, and f-measure can be calculated from the components of confusion matrix with the

equations given below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.4}$$

AUC is the area under ROC curve and calculated by integral. Receiver Operating Characteristic, also called as ROC, is a graphical plot that provides information about classifier performance. The x axis of ROC curve is the false positive rate (FPR) and the y axis is the true positive rate (TPR).

$$TPR = \frac{TP}{TP + FN} \tag{5.5}$$

$$FPR = \frac{FP}{FP + TN} \tag{5.6}$$

Cost, the grading criteria of KDD'99 contest, is calculated by the formula given below:

$$Cost = \frac{[ConfusionMatrix] * [CostMatrix]^T}{TotalNumberofInstances} \tag{5.7}$$

Default cost matrix of a two class classifier is shown on Table 5.2. Logically, predicting the class correctly has zero cost. So, true positive and true negative components of matrix are equal to 0. Wrong predictions should have a cost. In default matrix, false

positive and false negative components have equal costs and are set to 1. Cost matrix can be changed according to the problem.

Table 5.2. Default cost matrix of a two class classifier.

Cost Matrix		**Predicted Class**	
		Positive	Negative
Actual Class	Positive	0	1
	Negative	1	0

KDD'99 is a multi-class dataset. It has 5 class labels. The cost matrix which was used in competition is given below on Table 5.3. We also used this matrix in our cost calculations.

Table 5.3. KDD'99 contest cost matrix [8].

Cost Matrix		**Predicted Class**				
		dos	*u2r*	*r2l*	*probe*	*normal*
Actual Class	*dos*	0	2	2	1	2
	u2r	2	0	2	2	3
	r2l	2	2	0	2	4
	probe	2	2	2	0	1
	normal	2	2	2	1	0

5.2. Experimental Results

Experiments were performed according to the method that was explained in detail in the proposed intrusion detection system architecture section.

Firstly, we tested discretization, feature selection, and classification combination without applying bagging. The results are evaluated according to the different performance measures that were given in the previous section. We performed sixteen different

tests because we have four discretization methods, four feature selection methods, and one classification method. We preprocessed the training set by combining one of four discretization methods (EBD, EFD, EWD, and PKID) with one of four feature selection methods (CFS, CONS, IG, and SU). After preprocessing, we classified the training instances by using J48 decision tree algorithm and built the decision tree model according to the training dataset. Then, we tested our decision tree model by using a labelled test dataset. Therefore, we were able to compare our predictions with real class labels.

The results can be seen on the Table 5.4. According to these results, the best performance is achieved by Proportional K-Interval Discretization method and Information Gain based feature selection method combination for accuracy, precision, recall, f-measure, and cost performance metrics. For Area Under ROC Curve performance metric, Equal Frequency Binning Discretization method and Consistency Based feature selection method combination gives the best result. For DOS detection rate performance metric, Equal Frequency Binning Discretization method and Information Gain based feature selection method combination, and Equal Frequency Binning Discretization and Symmetrical Uncertainty Based Feature selection method combination give the same and best results.

Correlation based feature selection method works better with Entropy Based Discretization method. Consistency based feature selection method works better with Equal Frequency Binning Discretization and Equal Width Binning Discretization methods. Information Gain based feature selection method works better with Proportional K-Interval Discretization method. Symmetrical Uncertainty based feature selection works better with Equal Frequency Binning Discretization.

Secondly, we added bagging process to our structure. We produced ten new training sets from the original set by sampling with replacement. We preprocessed ten new training sets by combining one of four discretization methods (EBD, EFD, EWD, and PKID) with one of four feature selection methods (CFS, CONS, IG, and SU). After preprocessing, we classified each of ten training sets by using J48 decision

Table 5.4. Performance of decision tree based classifier with discretization and feature selection methods in the absence of bagging.

Discretization Method	**Feature Selection Method**	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**AUC**	**DOS Detection Rate**	**Cost**
EBD	**CFS**	0.9188	0.892	0.919	0.897	0.961	0.972	0.2498
EBD	**CONS**	0.9179	0.884	0.918	0.897	0.962	0.968	0.2484
EBD	**IG**	0.9237	0.936	0.924	0.904	0.968	0.974	0.2348
EBD	**SU**	0.9240	0.936	0.924	0.904	0.969	0.974	0.2340
EFD	**CFS**	0.9174	0.928	0.917	0.899	0.962	0.968	0.2480
EFD	**CONS**	0.9221	0.917	0.922	0.902	0.972	0.972	0.2542
EFD	**IG**	0.9260	0.937	0.926	0.910	0.960	0.975	0.2263
EFD	**SU**	0.9260	0.937	0.926	0.910	0.960	0.975	0.2264
EWD	**CFS**	0.9185	0.935	0.918	0.903	0.937	0.966	0.2536
EWD	**CONS**	0.9222	0.94	0.922	0.906	0.941	0.969	0.2485
EWD	**IG**	0.9238	0.937	0.924	0.908	0.942	0.971	0.2408
EWD	**SU**	0.9238	0.937	0.924	0.908	0.942	0.971	0.2408
PKID	**CFS**	0.9041	0.907	0.904	0.882	0.943	0.970	0.2775
PKID	**CONS**	0.9199	0.923	0.92	0.902	0.959	0.969	0.2394
PKID	**IG**	0.9291	0.944	0.929	0.918	0.962	0.971	0.2229
PKID	**SU**	0.9220	0.935	0.922	0.904	0.960	0.973	0.2369

tree algorithm and built ten decision tree models according to each training sets. We built our model according to the predictions that came from each decision tree created for each new training set. We combined the predictions by voting method. Than, we tested our model sixteen times for all combinations of discretization, feature selection, and pruned J48 decision tree method by using a labelled test dataset. Therefore, we were able to compare our predictions with real class labels.

The results can be seen on the Table 5.5. According to these results, the best performance is achieved by Proportional K-Interval Discretization method and Information Gain based feature selection method combination for accuracy, precision, recall, f-measure, dos detection rate, and cost performance metrics. For Area Under ROC Curve performance metric, Entropy Based Discretization method and Consistency Based feature selection method combination gives the best result.

Finally, we switched off the pruning property of J48 decision tree for making the learning scheme more unstable. Bagging performs better with unstable learning schemes. Then, we tested our model sixteen times more for all combinations of discretization, feature selection, and unpruned J48 decision tree method.

The results can be seen on the Table 5.6. According to the results, the best performance is achieved by Proportional K-Interval Discretization method and Information Gain based feature selection method combination for accuracy, precision, recall, f-measure, and cost performance metrics. For Area Under ROC Curve performance metric, Entropy Based Discretization method and Consistency Based feature selection method combination, Entropy Based Discretization method and Information Gain based feature selection method combination, and Proportional K-Interval Discretization and Symmetrical Uncertainty based feature selection method combination give the same and best result. For DOS detection rate performance metric, Proportional K-Interval Discretization and Symmetrical Uncertainty Based Feature selection method combination gives the best results.

With bagging, both pruned and unpruned decision trees have the results ex-

Table 5.5. Performance of pruned decision tree based classifier with discretization methods, feature selection methods, and bagging.

Discretization Method	**Feature Selection Method**	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**AUC**	**DOS Detection Rate**	**Cost**
EBD	**CFS**	0.9193	0.906	0.919	0.897	0.977	0.972	0.2491
EBD	**CONS**	0.9260	0.943	0.926	0.91	0.983	0.972	0.2428
EBD	**IG**	0.9244	0.937	0.924	0.904	0.972	0.974	0.2341
EBD	**SU**	0.9246	0.938	0.925	0.905	0.974	0.974	0.2334
EFD	**CFS**	0.9177	0.935	0.918	0.899	0.950	0.968	0.2480
EFD	**CONS**	0.9239	0.936	0.924	0.906	0.967	0.973	0.2383
EFD	**IG**	0.9264	0.938	0.926	0.910	0.959	0.974	0.2246
EFD	**SU**	0.9263	0.938	0.926	0.910	0.958	0.974	0.2250
EWD	**CFS**	0.9185	0.935	0.918	0.903	0.940	0.966	0.2535
EWD	**CONS**	0.9217	0.94	0.922	0.905	0.959	0.968	0.2492
EWD	**IG**	0.9238	0.937	0.924	0.908	0.943	0.971	0.2408
EWD	**SU**	0.9238	0.937	0.924	0.908	0.943	0.971	0.2408
PKID	**CFS**	0.9060	0.919	0.906	0.886	0.951	0.970	0.2702
PKID	**CONS**	0.9236	0.936	0.924	0.907	0.962	0.972	0.2288
PKID	**IG**	0.9318	0.946	0.932	0.920	0.964	0.975	0.2199
PKID	**SU**	0.9246	0.936	0.925	0.908	0.960	0.974	0.2277

plained below. Correlation based feature selection method works better with Entropy Based Discretization method. Consistency based feature selection method works better with Entropy Based Discretization method. Information Gain based feature selection method works better with Proportional K-Interval Discretization method. Symmetrical Uncertainty based feature selection works better with Equal Frequency Binning Discretization.

Table 5.6. Performance of unpruned decision tree based classifier with discretization methods, feature selection methods, and bagging.

Discretization Method	**Feature Selection Method**	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**AUC**	**DOS Detection Rate**	**Cost**
EBD	**CFS**	0.9188	0.922	0.919	0.897	0.962	0.972	0.2467
EBD	**CONS**	0.9263	0.943	0.926	0.91	0.964	0.973	0.2417
EBD	**IG**	0.9267	0.923	0.927	0.910	0.964	0.975	0.2421
EBD	**SU**	0.9264	0.939	0.926	0.909	0.963	0.975	0.2426
EFD	**CFS**	0.9176	0.934	0.918	0.899	0.950	0.968	0.2487
EFD	**CONS**	0.9226	0.931	0.923	0.903	0.961	0.973	0.2424
EFD	**IG**	0.9266	0.938	0.927	0.911	0.963	0.974	0.2245
EFD	**SU**	0.9265	0.938	0.927	0.910	0.963	0.974	0.2251
EWD	**CFS**	0.9183	0.935	0.918	0.903	0.942	0.966	0.2537
EWD	**CONS**	0.9227	0.94	0.923	0.908	0.946	0.968	0.2443
EWD	**IG**	0.9237	0.937	0.924	0.908	0.941	0.970	0.2415
EWD	**SU**	0.9237	0.937	0.924	0.908	0.941	0.970	0.2415
PKID	**CFS**	0.9073	0.919	0.907	0.888	0.953	0.972	0.2677
PKID	**CONS**	0.9235	0.936	0.924	0.906	0.961	0.973	0.2318
PKID	**IG**	0.9328	0.946	0.933	0.921	0.964	0.975	0.2149
PKID	**SU**	0.9263	0.937	0.926	0.910	0.960	0.976	0.2249

Best results changes according to the performance measure. On the Table 5.7, most successful method combination for each performance measure and their values can be observed.

Table 5.7. The method combinations that achieved best results according to different performance measures.

	Accuracy	**Precision**	**Recall**	**F-Measure**	**AUC**	**DOS Detection Rate**	**Cost**
Value	0.9328	0.946	0.933	0.921	0.983	0.976	0.2149
Method	Bagging PKID IG Unpruned_J48	Bagging PKID IG Unpruned_J48	Bagging PKID IG Unpruned_J48	Bagging PKID IG Unpruned_J48	Bagging EBD CONS Pruned_J48	Bagging PKID SU Unpruned_J48	Bagging PKID IG Unpruned_J48

In general Information Gain based feature selection and Symmetrical Uncertainty based feature selection methods give close and better results among all feature selection methods. Proportional K-Interval Discretization method gives better results among all discretization methods.

Applying bagging performed better in all sixteen combinations of methods. Bagging increased the performances of classifiers. It affected most the performance of Entropy Based Discretization and Consistency based feature selection method combination. For pruning, the condition is more complicated. Some classifiers work better with pruned J48 decision tree, some classifiers work better with unpruned one. On the Table 5.8, it can be seen which method performs better with which type of decision tree.

5.3. Comparison with Other Works

We compared our results with different works to interpret in a better way. We did not put all of our results for comparison because we concerned that the tables would be too long and obscure.

First of all, we compared methods based on accuracy and cost. It can be seen on Table 5.9. For accuracy performance metric, the method proposed by Koc *et al.* has the best result. Their method is a combination of proportional k-interval discretization method, interact feature selection method, and hidden naive bayes classifier. That

Table 5.8. The type of decision tree implementation that performs better for each filter combination.

Method/Type of Tree	**Pruned Tree**	**Unpruned Tree**
Bagging_EBD_CFS	✓	
Bagging_EBD_CONS		✓
Bagging_EBD_IG		✓
Bagging_EBD_SU		✓
Bagging_EFD_CFS	✓	
Bagging_EFD_CONS	✓	
Bagging_EFD_IG		✓
Bagging_EFD_SU		✓
Bagging_EWD_CFS	✓	
Bagging_EWD_CONS		✓
Bagging_EWD_IG	✓	
Bagging_EWD_SU	✓	
Bagging_PKID_CFS		✓
Bagging_PKID_CONS	✓	
Bagging_PKID_IG		✓
Bagging_PKID_SU		✓

combined method has 0.9372 accuracy [9]. After that, the method that gives the best accuracy result among all of our methods takes the second place. It is a combination of proportional k-interval discretization, information gain based feature selection, unpruned j48 decision tree, and bootstrap aggregating. Our method has 0.9328 accuracy. We have better results than KDD'99 contest's winner. KDD'99 winner has 0.9271 accuracy [8]. For accuracy performance metric, our method works better than all other methods that are proposed in various studies except Koc's method.

For cost performance metric, our proposed method has the best result among all other methods that are proposed in various studies. We have 0.2149 cost with our combined method. For accuracy, the method of Koc *et al.* was better than ours, but for cost metric, we have a better result because our method has less cost than their cost of 0.2224 [9]. We also have better results than KDD'99 constest's winner. KDD winner has the cost of 0.2331 [8].

Secondly, we compared methods based on DOS detection rate because our main concern is detecting DOS attacks. The comparison can be seen on Table 5.10. For DOS detection rate performance metric, the method of Koc *et al.* has the best results with 0.9960 detection rate [9]. The combination of proportional k-interval discretization, symmetrical uncertainty based feature selection, unpruned j48 decision tree, and bootstrap aggregating gives the best result among all of our combined methods. It has 0.9760 DOS detection rate. Our method works better than KDD'99 contest's winner and all other methods that are proposed in various studies except the method of Koc *et al.*.

Table 5.9. Performance comparison with various works based on accuracy and cost [8–11].

Performance of different methods	**Test Accuracies**	**Cost**
HNB_PKI_INT (Koç & Mazzuchi & Sarkani,2012)	0.9372	0.2224
Unpruned_J48_PKID_IG_Bagging (My result)	0.9328	0.2149
Pruned_J48_PKID_IG_Bagging (My result)	0.9318	0.2199
J48_PKID_IG (My result)	0.9291	0.2229
KDD'99 Winner	0.9271	0.2331
AODE_EMD_INT (Koç & Mazzuchi & Sarkani,2012)	0.9269	0.2336
Unpruned_J48_EBD_IG_Bagging (My Result)	0.9267	0.2421
Unpruned_J48_EFD_IG_Bagging (My result)	0.9266	0.2245
Unpruned_J48_EFD_SU_Bagging (My result)	0.9265	0.2251
Pruned_J48_EFD_IG_Bagging (My Result)	0.9264	0.2246
Unpruned_J48_EBD_SU_Bagging (My Result)	0.9264	0.2426
Unpruned_J48_PKID_SU_Bagging (My Result)	0.9263	0.2249
Pruned_J48_EFD_SU_Bagging (My result)	0.9263	0.2250
Unpruned_J48_EBD_CONS_Bagging (My Result)	0.9263	0.2417
NB_EMD_INT (Koç & Mazzuchi & Sarkani,2012)	0.9262	0.2254
TAN_PKI_CONS (Koç & Mazzuchi & Sarkani,2012)	0.9259	0.2393
AODE_PKI_CONS (Koç & Mazzuchi & Sarkani,2012)	0.9259	0.2429
HNB_EMD_CONS (Koç & Mazzuchi & Sarkani,2012)	0.9254	0.2393
JRip (Nguyen & Choi, 2008)	0.9230	NA
NB_PKI_INT (Koç & Mazzuchi & Sarkani,2012)	0.9230	0.2443
WAODE_EMD_CONS (Koç & Mazzuchi & Sarkani,2012)	0.9230	0.2454
NBTree (Nguyen & Choi, 2008)	0.9228	NA
WAODE_PKI_INT (Koç & Mazzuchi & Sarkani,2012)	0.9227	0.2477
LBk (Nguyen & Choi, 2008)	0.9222	NA
SVM (Ambwani 2003)	0.9218	NA
J48 (Nguyen & Choi, 2008)	0.9206	NA

Table 5.10. Performance comparison with various works based on DOS detection rate [8–10, 12].

Performance of different methods for DOS detection	**Detection Rate**
HNB_PKI_INT (Koç & Mazzuchi & Sarkani)	0.9960
Unpruned_J48_PKID_SU_Bagging (My Result)	0.9760
TAN_PKI_CONS (Koç & Mazzuchi & Sarkani)	0.9757
DTNB_PKI_CONS (Koç & Mazzuchi & Sarkani)	0.9751
AODE_PKI_CONS (Koç & Mazzuchi & Sarkani)	0.9750
Unpruned_J48_PKID_IG_Bagging (My result)	0.9750
Unpruned_J48_EBD_IG_Bagging (My result)	0.9750
Unpruned_J48_EBD_SU_Bagging (My result)	0.9750
Pruned_J48_PKID_IG_Bagging (My result)	0.9750
J48_EFD_IG (My result)	0.9750
J48_EFD_SU (My result)	0.9750
JRip_DecisionTable_OneR (Nguyen & Choi)	0.9740
KDD'99 Winner	0.9712
BinaryClassifier_IGRatio_Multiboosting (Dartigue & Jang & Zeng)	0.9700
J48_BayesNet_OneR (Nguyen & Choi)	0.9680
BinaryClassifier_IGRatio (Dartigue & Jang & Zeng)	0.9620

6. CONCLUSION AND FUTURE WORK

We proposed a decision tree based intrusion detection system which uses discretization, feature selection, and bootstrap aggregating methods for performance improvement.

KDD'99 dataset is used because it is very common in literature among computer security research. It provided a possibility to compare our work with different studies.

WEKA is used for implementation of machine learning algorithms. It is easy to understand and use. It contains many different techniques.

The main contribution of this thesis is combining various discretization and feature selection methods and additionally using bootstrap aggregating with them. There are four discretization and four feature selection methods that are used in experiments. EBD, EFD, EWD, and PKID are the discretization methods. CFS, CONS, IG, and SU are the feature selection methods. We used both pruned and unpruned J48 decision tree classifier. Experiments were performed with and without bootstrap aggregating.

Effect of pruning to the results changes according to the method combination. On the other hand, applying bootstrap aggregating improves the performance in all cases significantly.

We achieved our best result with the combination of PKID, IG, unpruned J48, and bootstrap aggregating. Using only PKID, IG and pruned J48, the results achieved are 92.91% accuracy and 0.2229 cost. Adding bootstrap aggregating to this structure, accuracy becomes 93.1768% and cost becomes 0.2199. If we make the tree unpruned for making the structure more compatible with bootstrap aggregating, accuracy becomes 93.2807% and cost becomes 0.2149.

The method combination that bootstrap aggregating has affected most is EBD, CONS, and J48. The accuracy was 91.7873% and cost was 0.2484. After bagging and unpruning the results become 92.6341% accuracy and 0.2417 cost.

The method combination that achieved best DOS detection rate is PKID, SU, UnprunedJ48, and Bagging with a detection rate of 97.6%.

We achieved better results than KDD'99 winner for accuracy, cost and DOS detection rate performance metrics.

For future work;

- These methods can be performed on different training and testing datasets to verify the efficiency.
- Different classifier algorithms can be used with discretization and attribute selection filters while applying bootstrap aggregating simultaneously.
- Using bootstrap aggregating with hybrid intrusion detection systems can be tried.
- Bootstrap aggregating and random forests can be combined.

REFERENCES

1. Stallings, W., *Cryptography and Network Security: Principles and Practice*, Prentice Hall, New Jersey, 2nd edition, 1999.

2. Alpaydın, E., *Introduction to Machine Learning*, The MIT Press, Massachusetts, 2nd edition, 2010.

3. Witten, I. H., E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers, Massachusetts, 3rd edition, 2011.

4. Breiman, L., "Bagging Predictors", *Machine Learning*, Vol. 24, pp. 123–140, 1996.

5. Dwyer, K. and R. Holte, "Decision Tree Instability and Active Learning", *18th European Conference on Machine Learning (ECML), Warsaw, Poland, September 17-21, 2007*, Springer-Verlag Berlin Heidelberg, Germany, 2007.

6. Hettich, S. and S. D. Bay, "KDD Cup 1999 Data", 1999, `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, August 2014.

7. Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update", *SIGKDD Explorations*, Vol. 11, No. 1, pp. 10–18, 2009.

8. Elkan, C., "Results of the KDD'99 Classifier Learning Contest", *SIGKDD Explorations*, Vol. 1, No. 2, pp. 63–64, 2000.

9. Koc, L., T. A. Mazzuchi, and S. Sarkani, "A Network Intrusion Detection System Based on a Hidden Naïve Bayes Multiclass Classifier", *Expert Systems with Applications*, Vol. 39, pp. 13492–13500, 2012.

10. Nguyen, H. A. and D. Choi, "Application of Data Mining to Network Intrusion Detection: Classifier Selection Model", *11th Asia-Pacific Network Operations and Management Symposium, Beijing, China, October 22-24, 2008*, Springer-Verlag, Germany, 2008.

11. Ambwani, T., "Multi Class Support Vector Machine Implementation to Intrusion Detection", *International Joint Conference on Neural Networks, Portland, Oregon, July 20-24, 2003*, IEEE, New York, 2003.

12. Dartigue, C., H. I. Jang, and W. Zeng, "A New Data-Mining Based Approach for Network Intrusion Detection", *Seventh Annual Communications Networks and Services Research Conference, New Brunswick, Canada, May 11-13 ,2009*, IEEE Computer Society, California, 2009.

13. Bishop, M., *Introduction to Computer Security*, Addison-Wesley Professional, USA, 2004.

14. Pfleeger, C. P. and S. L. Pfleeger, *Security in Computing*, Prentice Hall, Massachusetts, 4th edition, 2006.

15. Depren, O., M. Topallar, E. Anarim, and M. K. Ciliz, "An Intelligent Intrusion Detection System (IDS) for Anomaly and Misuse Detection in Computer Networks", *Expert Systems with Applications*, Vol. 29, pp. 713–722, 2005.

16. Wang, S.-S., K.-Q. Yan, S.-C. Wang, and C.-W. Liu, "An Integrated Intrusion Detection System for Cluster-based Wireless Sensor Networks", *Expert Systems with Applications*, Vol. 38, pp. 15234–15243, 2011.

17. Lee, J.-H., J.-H. Lee, S.-G. Sohn, J.-H. Ryu, and T.-M. Chung, "Effective Value of Decision Tree with KDD 99 Intrusion Detection Datasets for Intrusion Detection System", *10th International Conference on Advanced Communications Technology, Gangwon-Do, Korea, February 17-20, 2008*, Global IT Research Institute, Korea, 2008.

18. Bolón-Canedo, V., N. Sánchez-Maroño, and A. Alonso-Betanzos, "Feature Selection and Classification in Multiple Class Datasets: An Application to KDD Cup 99 Dataset", *Expert Systems with Applications*, Vol. 38, pp. 5947–5957, 2011.

19. Chandolikar, N. S. and V. D. Nandavadekar, "Efficient Algorithm for Intrusion Attack Classification by Analyzing KDD Cup 99", *Ninth International Conference on Wireless and Optical Communications Networks (WOCN), Indore, India, September 20-22, 2012*, IEEE, New York, 2012.

20. Koshal, J. and M. Bag, "Cascading of C4.5 Decision Tree and Support Vector Machine for Rule Based Intrusion Detection System", *International Journal of Computer Network and Information Security(IJCNIS)*, Vol. 4, No. 8, pp. 8–20, 2012.

21. Kim, G., S. Lee, and S. Kim, "A Novel Hybrid Intrusion Detection Method Integrating Anomaly Detection with Misuse Detection", *Expert Systems with Applications*, Vol. 41, pp. 1690–1700, 2014.

22. Thaseen, S. and C. A. Kumar, "An Analysis of Supervised Tree Based Classifiers for Intrusion Detection System", *International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), Tamilnadu, India, February 21-22, 2013*, IEEE Press, New York, 2013.

23. Gyanchandani, M., R. N. Yadav, and J. L. Rana, "Intrusion Detection using C4.5: Performance Enhancement by Classifier Combination", *The Association of Computer Electronics and Electrical Engineers(ACEEE) International Journal of Signal and Image Processing*, Vol. 1, No. 3, pp. 46–49, 2010.

24. Alhaddad, M. J., A. Ahmed, S. M. Halawani, A. H. Altalhi, and A. S. Mashat, "A Study of the Modified KDD 99 Dataset by Using Classifier Ensembles", *International Organization of Scientific Research Journal of Engineering*, Vol. 2, No. 5, pp. 1961–965, 2012.

25. Fayyad, U. M. and K. B. Irani, "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning.", *13th International Joint Conference on Artificial Intelligence, Chambery, France, August 28 - September 3 ,1993*, Morgan Kaufmann, San Francisco, USA, 1993.

26. Bousquet, O. and A. Elisseef, "Stability and Generalization", *Journal of Machine Learning Research*, Vol. 2, pp. 499–526, 2002.

27. Lahre, K., T. D. Diwan, S. K. Kashyap, and P. Agrawal, "Analyse Different Approaches for IDS Using KDD 99 Data Set", *International Journal on Recent and Innovation Trends in Computing and Communication*, Vol. 1, No. 8, pp. 645–651, 2013.