PROJECT No. ISNE 01/2559

Network Traffic Anomaly Detection

Jirayuth Kaewprakan 560611007
Supat Kongka 560611029

A Project Submitted in Partial Fulfillment of Requirements
for the Degree of Bachelor of Engineering
Department of Computer Engineering
Faculty of Engineering
Chiang Mai University
2016

โครงงานเลขที่ วศ.สค. 01/2559

เรื่อง

การตรวจจับสิ่งผิดปกติในการจราจรทางเครือข่าย

โดย

นาย จิรายุทธ แก้วประการ รหัส 560611007
นาย สุภัทร คงคา รหัส 560611029

โครงงานนี้
เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
ปีการศึกษา 2559

Project Title	: Network Traffic Anomaly Detection
Name(s)	: Jirayuth Kaewprakan 560611007
	: Supat Kongka 560611029
Department	: Computer Engineering
Project Advisor	: Assoc. Prof. Sansanee Auephanwiriyakul, Ph.D.
Degree	: Bachelor of Engineering
Program	: Information Systems and Network Engineering
Academic Year	: 2016

Department of Computer Engineering, Faculty of Engineering, Chiang Mai University has approved this project to be part of the degree of Bachelor of Engineering (Information Systems and Network Engineering)

..... Head of the department

(Assoc. Prof. Sansanee Auephanwiriyakul, Ph.D.)

Project examination committee:

..... Main Advisor/Chair

(Assoc. Prof. Sansanee Auephanwiriyakul, Ph.D.)

..... Committee

(Assoc. Prof. Nipon Theera-Umpon, Ph.D.)

..... Committee

(Dome Potikanond)

หัวข้อโครงงาน	: การตรวจจับสิ่งผิดปกติในการจราจรทางเครือข่าย
โดย	: นาย จิรายุทธ แก้วประการ รหัส 560611007
	: นาย สุภัทร คงคา รหัส 560611029
ภาควิชา	: วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	: รศ.ดร.ศันสนีย์ เอื้อพันธ์วิริยะกุล
ปริญญา	: วิศวกรรมศาสตร์บัณฑิต
สาขา	: วิศวกรรมระบบสารสนเทศและเครือข่าย
ปีการศึกษา	: 2559

บทคัดย่อ

แม้จะมีความก้าวหน้าอย่างรวดเร็วในด้านเทคโนโลยีสารสนเทศ แต่ปัญหาของการป้องกันคอมพิวเตอร์และเครือข่ายการรักษาความปลอดภัยยังคงเป็นความท้าทายที่สำคัญสำหรับนักวิจัยส่วนใหญ่โดยเฉพาะอย่างยิ่งหลังจากที่การขยายตัวของเครือข่ายและวิวัฒนาการของเทคโนโลยีและการเพิ่มจำนวนของผู้ใช้เครือข่ายอินเทอร์เน็ต เครือข่ายจำเป็นต้องมีเครื่องมือบางอย่างสำหรับการป้องกัน เช่น ไฟร์วอลล์ เป้าหมายของโครงการนี้คือการสร้างระบบตรวจจับความผิดปกติ ระบบนี้จะขึ้นอยู่กับลักษณะการทำงานปกติของเครือข่ายในการที่จะสามารถแยกแยะแต่ละพฤติกรรมที่ผิดปกติในการทดลองและประเมินผลของระบบตรวจจับสิ่งผิดปกติ ได้ดำเนินการโดยใช้ชุดข้อมูลจาก KDD Cup 99 และใช้ร่วมกับข้อมูลจาก CTU-Malware-Capture-Botnet-42 เป็นข้อมูลที่ได้จากการดักจับข้อมูลในระบบเครือข่ายซึ่งมีทั้งข้อมลที่ปกติและผิดปกติในห้องทดลองของมหาวิทยาลัยซีทียูในปี 2011 ซึ่งผลการทดลองสามารถแยกแยะข้อมูลที่ปกติ และข้อมูลที่ผิดปกติ โดยใช้ระบบการตัดสินใจแบบต้นไม้และฟัซซีโลจิก

Project Title	: Network Traffic Anomaly Detection
Name(s)	: Jirayuth Kaewprakan 560611007
	: Supat Kongka 560611029
Department	: Computer Engineering
Project Advisor	: Assoc. Prof. Sansanee Auephanwiriyakul, Ph.D.
Degree	: Bachelor of Engineering
Program	: Information Systems and Network Engineering
Academic Year	: 2016

ABSTRACT

Despite the rapid progress in information technology, the problem of protecting computer and network security remained a major challenge for most researchers, especially after the expansion of networks and evolution of technology and the increasing number of network users and the internet. Tools are needed for network protection, such as firewall. The aim of this project is to build an anomaly detection system. This system depends on the normal behavior of the network, that we will alert abnormal behavior. The experiments and evaluations of the proposed intrusion detection system are performed on the KDD Cup 99 intrusion detection dataset and the CTU-Malware-Capture-Botnet-42 dataset of botnet traffic that was capture in the CTU University, Czech University, in 2011. The result we can identify which one is normal or attack by using decision tree and fuzzy logic.

Acknowledgement

I would like to express my special thanks of gratitude to my advisor, Assoc.Prof. Sansanee Auephanwiriyakul, Ph.D, who gave us guidance and the golden opportunity to do this wonderful project. We are also thankful to Mr. Dome Potikanondand Assoc. Prof. Nipon Theera-Umpon, Ph.D. who provided insight and expertise that greatly assisted the project. And we are also thankful to Dr. Mahamah Sohbakor who guided we to the dataset and helped we to complete this the project.

This project would not have been possible if we did not have a support of many friends. Therefore, we would like to thank all of them.

Lastly we would like to extend our deepest gratitude to our parents who unconditionally supported us so that we can achieve our goal.

This research was partially supported by the Computational Intelligence Laboratory, Electronic Engineering, Chiang Mai University.

Jirayuth Kaewprakan
Supat Kongka
13 May 2017

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Project Rationale

Monitoring the whole network traffic in a network is a challenging issue, there is a difficulty in knowing whether a contact is normal or attack. Accuracy and efficiency are two very important performance measures for the anomaly detection It is necessary to provide valuable information admin who monitors the computer system and the network. In addition, it is important to work in real-time, or as close to real-time as possible. Real-time operation is necessary for the anomaly detection analyst so as to help protect network before they do much harm to the protected systems.

1.2 Objectives

1. To study and development a network anomaly detection system that helps to detect abnormal behavior of network traffic.

1.3 Project Scope

1. Data set from the KDD'99 CUP Dataset. [1]
2. Data set of capture Wireshark from the CTU-13 Dataset.

1.4 Expected outcomes

The system is able to detect abnormalities on the network.

1.5 Technology and Tools

1. Anaconda-Navigator(Python)
2. Library for this support system
    a. Sk-lean
    b. Pandas

1.6 Project Plan

Research Schedule	2560				
	Jan	Feb	Mar	Apr	May
1. Researching about Anomaly Detection.					
2. Analyze data about attack analysis of Anomaly Detection					
3. Study about pattern of attacks types					
4. Writing program for analyze data about attack analysis of Anomaly Detection					
5. System testing and collecting data					
6. Writing report					

1.7 Roles and Responsibilities

Jirayuth Kaewprakan

- Responsible for the classify by fuzzy logic.
- Extract data from Wireshark and test the system from the dataset of Wireshark.
- Split files for doing Cross-validation of the system from the dataset of Wireshark.
- Make a document.

Supat Kongka

- Responsible for the classify by Decision tree
- Split files for doing Cross-validation of dataset of KDD Cup-99 dataset.
- Testing systems and find the result of the decision tree.
- Make a document.

Chapter 2
Background Knowledge and Theory

2.1 Introduction

With the growth of network services and sensitive information on networks, network security is getting more and more importance than ever. Intrusion attacks are serious security risk in a network environment.

2.2 Anomaly Detection

Anomaly detection assumes that intrusions are anomalies that necessarily differ from normal behavior. Basically, anomaly detection establishes a profile for normal operation and marks the activities that deviate significantly from the profile as attacks. The main advantage of anomaly detection is that it can detect unknown attacks.

2.3 Dataset

2.3.1 KDD Cup 1999 dataset

The KDD datasets is employed in international knowledge discovery and data mining tools. KDD Cup 1999 dataset[1] is consist of large number of redundant records. This dataset is given as a input to proposed system to performed training and testing operations. For each connection of TCP/IP, 41 various continuous data type and discrete data types features were extracted. Among this 41 features, 34 features are continuous or numeric and 7 features are symbolic as shown in Table 2.1.

Table 2.1 A complete list of features given in KDD cup 99 dataset

Feature index	Feature name	Description	Type
1	duration	length (number of seconds) of the connection	Continuous
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	Symbolic
3	service	network service on the destination, e.g., http, telnet, etc.	Symbolic
4	flag	normal or error status of the connection	Symbolic
5	src_bytes	number of data bytes from source to destination	Continuous

6	dst_bytes	number of data bytes from destination to source	Continuous
7	Land	1 if connection is from/to the same host/port; 0 otherwise	Symbolic
8	wrong_ fragment	number of ``wrong'' fragments	Continuous
9	urgent	number of urgent packets	Continuous
10	hot	number of ``hot'' indicators	Continuous
11	num_failed_ logins	number of failed login attempts	Continuous
12	logged_in	1 if successfully logged in; 0 otherwise	Symbolic
13	num_compro mised	number of ``compromised'' conditions	Continuous
14	root_shell	1 if root shell is obtained; 0 otherwise	Continuous
15	su_attempted	1 if ``su root'' command attempted; 0 otherwise	Continuous
16	num_root	number of ``root'' accesses	Continuous
17	num_file_ creations	number of file creation operations	Continuous
18	num_shells	number of shell prompts	Continuous
19	num_access_ files	number of operations on access control files	Continuous
20	num_outboun d_cmds	number of outbound commands in an ftp session	Continuous
21	is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	Symbolic
22	is_guest_login	1 if the login is a ``guest'' login; 0 otherwise	Symbolic
23	count	number of connections to the same host as the current connection in the past two seconds	Continuous
24	srv_count	number of connections to the same service as the current connection in the past two seconds	Continuous
25	serror_rate	% of connections that have ``SYN'' errors	Continuous
26	srv_serror_rate	% of connections that have ``SYN'' errors	Continuous
27	rerror_rate	% of connections that have ``REJ'' errors	Continuous

28	srv_rerror_rate	% of connections that have ``REJ'' errors	Continuous
29	same_srv_rate	% of connections to the same service	Continuous
30	diff_srv_rate	% of connections to different services	Continuous
31	srv_diff_host_ rate	% of connections to different hosts	Continuous
32	dst_host_ count	count for destination host	Continuous
33	dst_host_srv_ count	srv_count for destination host	Continuous
34	dst_host_ same_srv_rate	same_srv_rate for destination host	Continuous
35	dst_host_diff_ srv_rate	diff_srv_rate for destination host	Continuous
36	dst_host_ same_src_ port_rate	same_src_port_rate for destination host	Continuous
37	dst_host_srv_ diff_host_rate	diff_host_rate for destination host	Continuous
38	dst_host_ serror_rate	serror_rate for destination host	Continuous
39	dst_host_srv_ serror_rate	srv_serror_rate for destination host	Continuous
40	dst_host_ rerror_rate	rerror_rate for destination host	Continuous
41	dst_host_srv_ rerror_rate	srv_serror_rate for destination host	Continuous

2.3.2 CTU-Malware-Capture-42 Dataset

The CTU-13 is a dataset of botnet traffic that was captured in the CTU University, Czech Republic, in 2011. The goal of the dataset was to have a large capture of real botnet traffic mixed with normal traffic and background traffic. The CTU-13 dataset consists in thirteen captures (called scenarios) of different botnet samples.

These propose system using the CTU-Malware-Capture-Botnet-42, The first hour of capture was composed of only Background traffic and latter we run the

malware. The malware was stopped 5 minutes before ending the capture. This dataset corresponds to a network traffic that run for 6.15 hours in a University network.

The relationship between the duration of the scenario, the number of packets, the number of NetFlows and the size of the pcap file is shown in Table 2.2

Table 2.2 A complete list of features given in CTU-Dataset

Feature index	Feature name	Description
1	Src_address	Source address
2	Dest_address	Destination address
3	Packets	number of data packets
4	Bytes	number of data bytes
5	Src_packets	number of data packets from source to destination
6	Src_byte	number of data bytes from source to destination
7	Dest_packet	number of data packets from destination to source
8	Dest_byte	number of data bytes from destination to source
9	Rel_start	time in seconds between the start of the packet
10	Duration	time in seconds between the first and last packet of the conversation
11	Src_bits_per_sec	number of data bytes from source to destination per second
12	Dest_bits_per_sec	number of data bytes from destination to source per second

2.4 Types of attacks

The system uses large number of dataset that contain 24 attack types. This attacks categories into four main types as follows.

2.4.1 Denial of Service(DoS)

Denial of service is an attack category, which depletes the victim's resources thereby making it unable to handle legitimate requests such as syn flooding.

Relevant features: "source bytes" and "percentage of packets with errors"

2.4.2 Remote to Local(R2L)

Unauthorized access from a remote machine, the attacker intrudes into a remote machine and gains local access of the victim machine such as password guessing.

Relevant features: Network level features – “duration of connection” and “service requested” and - “number of failed login attempts”

2.4.3 User to Root(U2R)

Unauthorized access to local super user (root) privileges is an attack type, by which an attacker uses a normal account to login into a victim system and tries to gain root or administrator privileges by exploiting some vulnerability in the victim such as buffer overflow attacks.

Relevant features: “number of file creations” and “number of shell prompts invoked”

2.4.4 Probe

Surveillance and other probing attack’s objective is to gain information about the remote victim e.g. port scanning.

Relevant features: “duration of connection” and “source bytes”

Table 2.3 Various types of attacks described in four major categories

Attack type	Sub attack types
Denial of Service Attacks	Back, land, neptune, pod, smurf, teardrop
User to Root Attacks	Buffer_overflow, loadmodule, perl, rootkit,
Remote to Local Attacks	Ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
Probes	Satan, ipsweep, nmap, portsweep

In KDD cup 99 Dataset ,all 41 features was clearly explained and distributed in two classes such as normal class and attack class. The list of attack class and normal class done in 10% KDD cup dataset as shown in Table 2.3.

Table 2.4 Attack types and Number of samples in 10% KDD Dataset

Attack type	Sub attack types	Number of samples
Normal	Normal	97277
Denial of Service (DOS)	Smurf	280790
	Neptune	107201
	Back	2203
	Teardrop	979
	Pod	264
	Land	21
User to Root (U2R)	Buffer_overflow	30
	Rootkit	10
	Load module	9
	Perl	3
Remote to Local (R2L)	Warezclient	1020
	Guess_passwd	53
	Warezmaster	20
	Imap	12
	ftp_write	8
	Multihop	7
	Spy	2
	Satan	1589
Probe	Ipsweep	1246
	Portsweep	1040
	Nmap	231

In the capture file of CTU-Malware-Capture-42 that include background, normal traffic in the first hour after that run the malware. The malware was stopped 5 minutes before ending the capture.

Table 2.5 Attack types and Number of samples in 10% KDD Dataset

Attack type	Sub attack types	Number of samples
Normal and background traffic	Normal	898
Attack	Attack	3433

2.5 Decision Tree

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

A decision tree[3] is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. In decision analysis, a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated. A decision tree consists of three types of nodes:

1. Decision nodes – typically represented by squares
2. Chance nodes – typically represented by circles
3. End nodes – typically represented by triangles

2.5.1 Decision tree elements

A decision tree[2] has only burst nodes (splitting paths) but no sink nodes (converging paths). Therefore, used manually, they can grow very big and are then often hard to draw fully by hand. Traditionally, decision trees have been created manually as shown in Figure 2.1

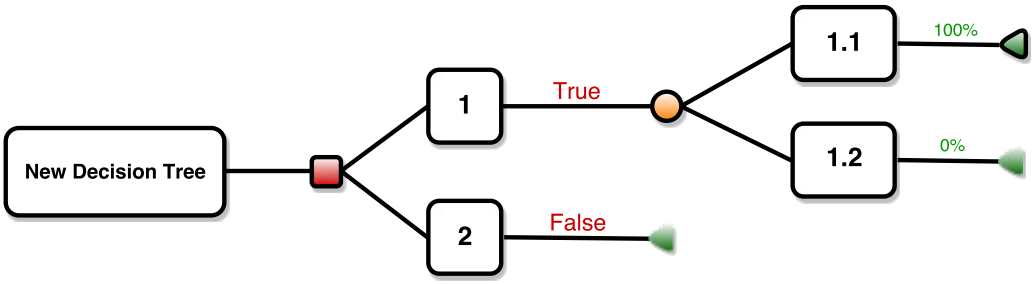


Figure 2.1 Pattern of Decision Tree

2.5.2 Decision rules

The decision tree can be linearized into decision rules,[6] where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if clause. In general, the rules have the form:

if condition1 and condition2 and condition3 then outcome.

2.5.3 Algorithm

The core algorithm for building decision trees called ID3 which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and Information Gain to construct a decision tree.

a. Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

Entropy $H(S)$ is a measure of the amount of uncertainty in the (data) set S (i.e. entropy characterizes the (data) set S) according to Eq. (2.1)

$$\mathrm{H(S)} = \sum_{\mathrm{x \in X}} \mathrm{p(x) log_2 (\frac{1}{p(x)})} \qquad (2.1)$$

When

S - The current (data) set for which entropy is being calculated (changes every iteration of the ID3 algorithm)

X - Set of classes in S

$p(x)$ - The proportion of the number of elements in class x to the number of elements in set S

When $H(S) = 0$, the set S is perfectly classified (i.e. all elements in S are of the same class).

b. Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Information gain IG(A) is the measure of the difference in entropy from before to after the set S is split on an attribute A. In other words, how much uncertainty in S was reduced after splitting set S on attribute A according to Eq. (2.2)

$$\mathrm{IG(A, S)} = \mathrm{H(S)} - \sum_{\mathrm{t \in T}} \mathrm{p(t) H(t)} \qquad (2.2)$$

When

$\mathrm{H(S)}$ - Entropy of set S

T - The subsets created from splitting set S by attribute A

$\mathrm{p(t)}$ - The proportion of the number of elements in t to the number of elements in set S

$\mathrm{H(t)}$ - Entropy of set T

2.6 Fuzzy Logic

Human behavior can explain the things around us with mathematical model both linear and non-linear. Set theory is one example of a mathematical model, which is to say that anything has certain properties that can be assigned to the same terms as a member of this group, which group of objects is called fuzzy set theory[4]. The theory of fuzzy set can help explain the behavior of the system in daily life more clearly than the mathematical description. The crisp set theory is not causing the main arguments of the theory of fuzzy set. The member that is characterized by a specific subset of them, even if only partially joined by members of all the features offered hefty levels as defined as shown in Figure 2.2, which is different from the theory of the Crisp set, which is the same theory that specifies clearly what was considered a member of the set is not only as shown in Figure 2.3

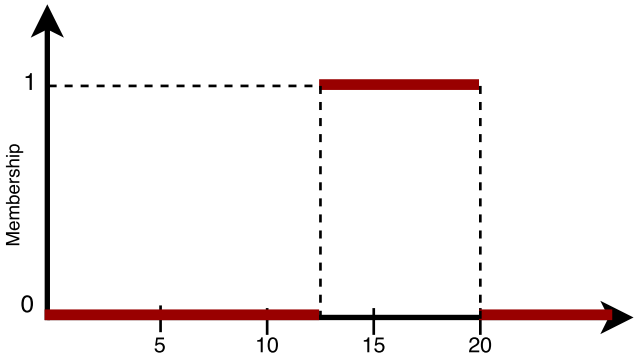


Figure 2.2 The function of a crisp set.

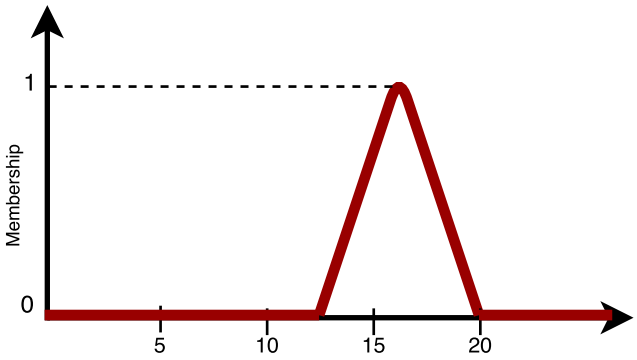


Figure 2.3 The function of a fuzzy set.

2.6.1 Fuzzy rules

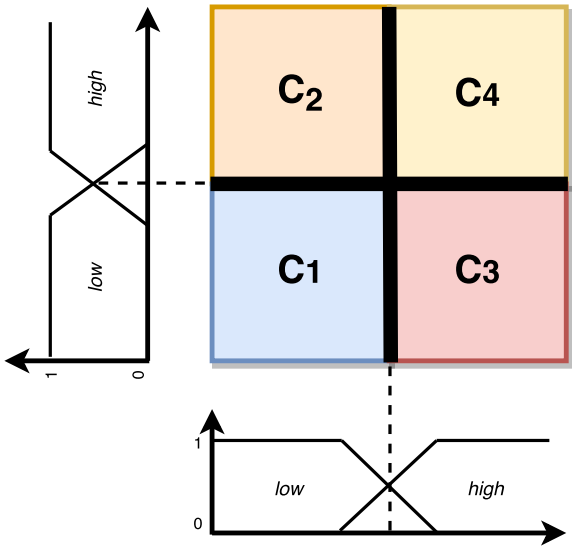


Figure 2.4 Pattern space for grouping by fuzzy rules.

From fig 2.4 rules can be written as follows in a sentence.

Rule 1: If x1 and x2 is **low** then set (x1, x2) as C1

Rule 2: If x1 is **low** and x2 is high then set (x1, x2) as C2

Rule 3: If x1 is **high** and x2 is low then set (x1, x2) as C3

Rule 4: If x1 and x2 is **high** then set (x1, x2) as C4

When x1is a linguistic variable in the first dimension, x2is a linguistic variable in the second dimension, low and high are the linguistic terms, set(x1, x2) is a sequence of objects to group and C1, C2, C3 and C4 is a group 1, 2, 3 and 4.[7]

2.6.2 Fuzzy Logic System

Infrastructure, processing fuzzy which it consists of four major parts as show in a fig 2.5

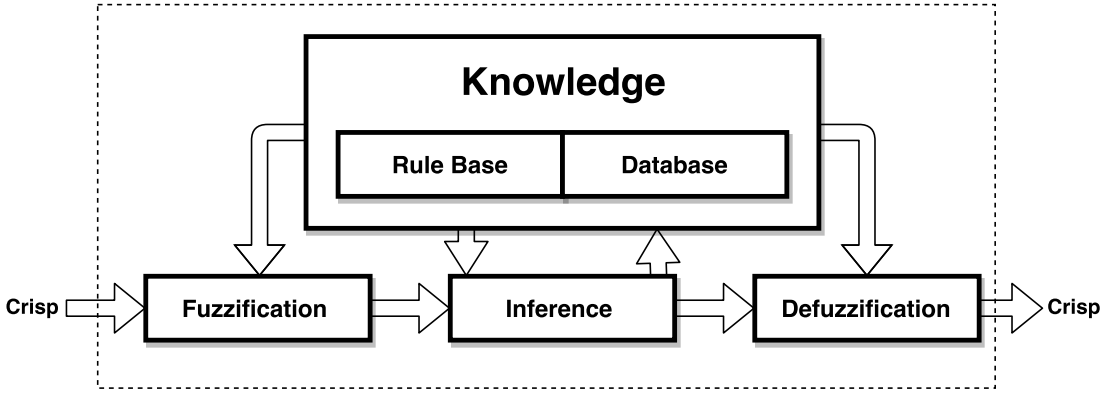


Figure 2.5 Fuzzy Logic System.

1. **Fuzzification** is the part of converting an input common to input fuzzy
2. **Knowledgebase** is the collection of data in the controller, which includes second parts: Rule base and Database.
3. **Inference Engine** is the duty to examine the facts and rules used to justify interpreted like a mechanism for controlling the use of knowledge to solve problems. Including determining how to interpret the answers.
4. **Defuzzification** is transforms data in the pattern of fuzzy to the summary or the control system.

2.6.3 Linguistic Hedges

Linguistic Hedges is unary operation of close range [0,1] i.e. For fuzzy predicate A universal set on X and the modified h, which is used instead of Linguistic Hedges H which fuzzy predicate was modified HA has a member function for $x \in X$ [6] according to Eq.(2.3)

$$\mathrm{HA(x)} = \mathrm{h(A(x))} \qquad (2.3)$$

2.6.4 Central of Gravity

The Center of Gravity method (COG) is the most popular defuzzification technique and is widely utilized in actual applications. This method is similar to the formula for calculating the center of gravity in physics. The weighted average of the membership function or the center of the gravity of the area bounded by the membership function curve is computed to be the most crisp value of the fuzzy quantity. The COG output according to Eq.(2.4)

$$\mathrm{COG} = \frac{\sum_{i=1}^{N} a_i w_i}{\sum_{i=1}^{N} a_i} \qquad (2.4)$$

When

COG - Central of Gravity

N - The value of position at first to position i.

a_i - The fuzzy value of output at position i.

w_i - The area under the curve of fuzzy set at position i.

Chapter 3
Project Structure and Methodology

This chapter discusses the principles of the classification and analysis of data on the network that the behavior is normal or abnormal. By comparison with the existing signature and the results were analyzed and verified to be attack or not

3.1 Decision tree structure

Decision support system provides decision which an algorithm learned from the training data.

- If training data is accurate then learning is more accurate

- Entropy can also be used for finding out missing values

Example: we can learn values from remaining tuples and predict value for unknown tuple.

Data used for training is the vital data and hence its cleaning should be done properly. More accurate the data more accurate the training and more efficient testing can be obtained.

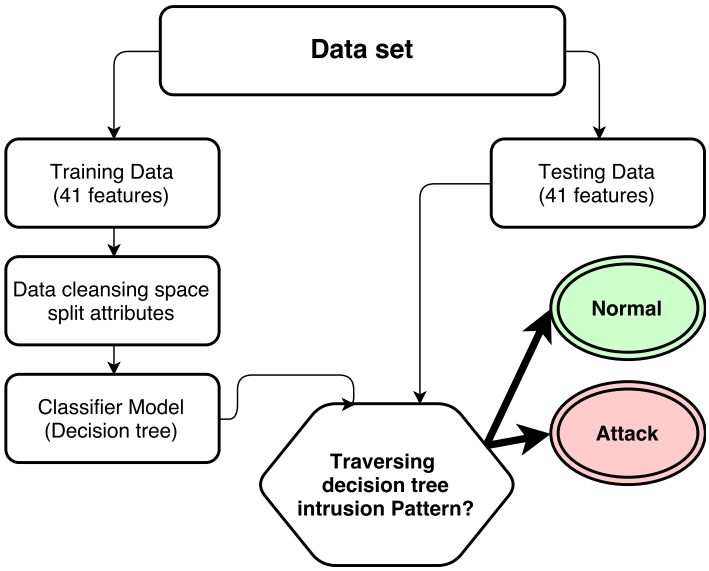


Figure 3.1 Overview of Decision tree system

3.2 Fuzzy logic structure

The fuzzy based network intrusion detection system have demonstrate their ability to identify various kind of anomaly in different applications domains. In general, network intrusion detection relies on expert knowledge of security, in particular their relationship with the computer system. The Network intrusion detection system are mainly used KDD Cup datasets for experimentation. This dataset are divided into training and testing datasets, which are used for detection of intrusion. The different steps involved in the network intrusion detection system are as follows[5]:

1. Classification of training datasets
2. Selection of suitable attributes
3. Generation of fuzzy rule
4. Fuzzy Decision model

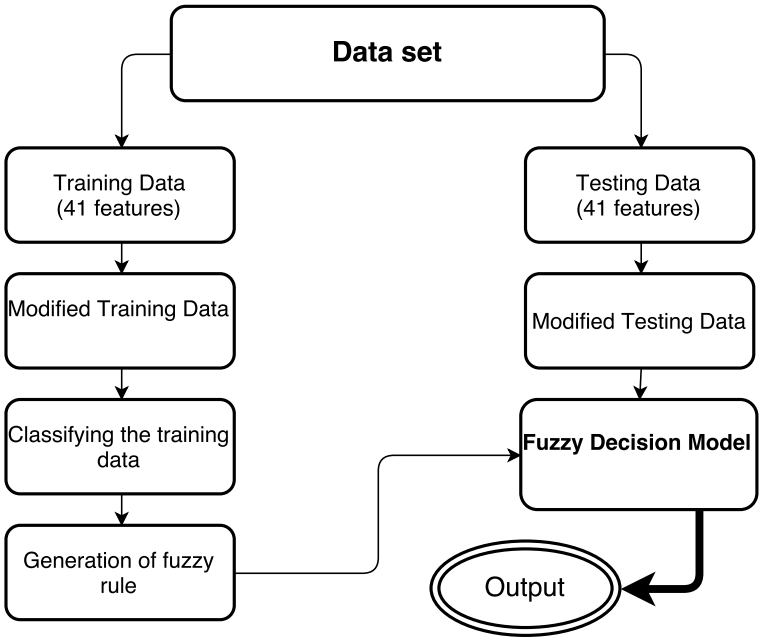


Figure 3.2 Overview of fuzzy logic system

3.2.1 Classification of training datasets

The first component of the proposed system is classifying the input data into multiple classes by taking in mind the different attacks involved in the intrusion detection dataset. The dataset we will be take for analyzing the intrusion detection behavior using the proposed system is KDD-Cup 1999 data. Based on the analysis, this data contains four types of attacks and normal behavior data with 41 attributes that have both continuous and symbolic attributes.

3.2.2 Selection of suitable attributes

In these, step most suitable attributes are taken for identifying the weather the data is normal or attack. The input data contain 34 attributes for KDD Cup99 dataset and 3 attributes for capture Wireshark dataset from these all attributes are not efficient or effective to detect intrusion. Hence, we need to select only those attributes that are suitable for detecting intrusion. Various method are used for selecting suitable attributes like deviation method.

3.2.3 Generation of fuzzy rule

Fuzzy system is used for obtaining fuzzy rules. If-then rules is successfully used in generating fuzzy rule in many application domains. These if-then rule is gained from knowledge expert. Each rule format as follows:

If <condition> then <action>

Example:

-If (dst_host_srv_count['low'] | dst_host_srv_count['high'] & dst_bytes['high'], types['Normal'])

-If(count['low'] & dst_host_same_srv_rate['low'] & dst_host_diff_srv_rate['low'] & same_srv_rate['low'] & diff_srv_rate['low'] , types['Attack'])

-If(count['meduim'] & same_srv_rate['high'] & diff_srv_rate['low'] & dst_host_same_srv_rate['low'] & dst_host_diff_srv_rate['low'], types['Attack'])

-If(count['high'] & same_srv_rate['high'] & diff_srv_rate['low'] & dst_host_same_srv_rate['meduim'] & dst_host_diff_srv_rate['low'], types['Attack'])

-If(dst_host_count['vvhigh'] , types['Attack'])

All actions usually decides from condition through evaluating present network connection. The action field submits on which condition what action to be taken.

3.2.4 Fuzzy Decision model

Fuzzy decision models contain 34 input and 1 output. The input given to the fuzzy decision models are releated to the 34 attributes where as it produces only one output is releated to the attack data and normal data. The fuzzy decision model is based on fuzzy logic. The fuzzy decision model as shown in Figure 3.3

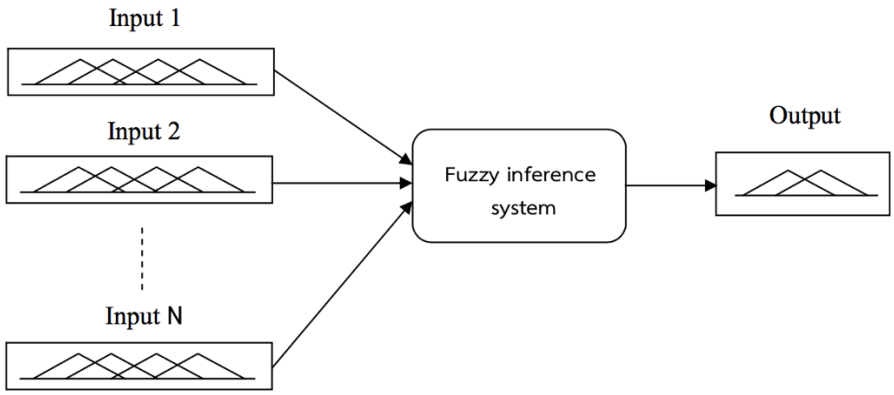


Figure 3.3 Fuzzy Decision model

Chapter 4
Experimentation and Results

This chapter describes the experimental results and performance evaluation of the proposed system. The proposed system is implemented in Python languages run on open source Anaconda. For experimental evaluation, we have taken KDD cup 99 dataset, which is mostly used for evaluating the performance of the intrusion detection system. For evaluating the performance, it is very difficult to execute the proposed system on the KDD cup 99 dataset since it is a large scale. Here, we have used subset of 10% of KDD Cup 99 dataset for training and testing. The number of records taken for testing and training phase is given in according to Table 4.1

Table 4.1 Training and testing dataset of KDD Cup-99 dataset

Type	Training Dataset	Testing Dataset
Normal	97278	80386
Denial of Service (DOS)	391458	230643
Probe	4107	3291
User to Root (U2R)	52	38
Remote to Local (R2L)	1126	1034

Table 4.2 Training and testing dataset of capture Wireshark from the CTU-13 Dataset

Type	Training Dataset	Testing Dataset
Normal	923	834
Attack	3376	3465

4.1 Experimental Results and Performance Analysis

The training dataset contains normal data and types of attacks, which are given to the proposed system for identifying result of the system. First, we using 10 fold cross validation to finding the best performance of the both dataset. After that using those for training and testing data in the anomaly detection system which include generate the rule for fuzzy logic then, using the fuzzy rule learning strategy and Decision Tree, the system generates definite and indefinite rules. Finally, decision tree and fuzzy rules are generated from the definite rules and predicted the label of test data.

In the testing phase, the testing dataset is given to the proposed system, which classifies the input as a normal or attack. The obtained result is then used to compute overall accuracy of the proposed system. The overall accuracy of the proposed system is computed based on the definitions, namely precision, recall and F-measure which are normally used to estimate the rare class prediction. It is advantageous to accomplish a high recall devoid of loss of precision. F-measure is a weighted harmonic mean which evaluates the trade-off between them.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (4.1)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4.2)$$

$$\text{F} - \text{measure} = \frac{(\beta^2 + 1)(\text{Precision} \cdot \text{Recall})}{\beta^2 \cdot (\text{Precision} + \text{Recall})} \quad \text{where}, \beta = 1 \quad (4.3)$$

$$\text{Overall accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \quad (4.4)$$

Where

$\text{True Positives(TP)}$ - Number of normal connections classified as normal.

$\text{True Negatives(TN)}$ - Number of attack connections classified as attacks.

$\text{False Negatives(FN)}$ - Number of attack connections classified as normal.

$\text{False Positives(FP)}$ - Number of normal connections classified as attacks.

4.2 Results of decision tree classifier

4.2.1 Results from KDD Cup-99 Dataset by using decision tree classifier.

Table 4.3 Results of validation set of KDD Cup-99 dataset by using decision tree classifier.

Each 10% of dataset	Results				
	Normal	Attack	Precision	Recall	Overall Accuracy
1	75521	369098	0.860274129	0.997298765	0.972027736
2	83830	360789	0.931422828	0.972766313	0.98136157
3	203532	241087	0.999543123	0.429961873	0.738965271
4	202483	242136	0.999748718	0.432278265	0.741405563
5	202560	242058	0.999840091	0.432148499	0.741265536
6	212312	281710	0.999866364	0.458127661	0.767097417
7	407201	86821	0.999969161	0.238889394	0.372643324
8	122400	371622	0.993976089	0.78997549	0.946777674
9	123763	370259	0.998961749	0.785194283	0.945982163
10	92183	401839	0.93562845	0.987351247	0.984964232

Table 4.4 Results of train set of KDD Cup-99 dataset by using decision tree classifier.

Each 90% of dataset	Results				
	Normal	Attack	Precision	Recall	Overall Accuracy
1	82751	361868	0.93396916	0.988133074	0.984789224
2	81384	363235	0.928418864	0.998771257	0.985679874

3	88103	356516	0.991639159	0.985426149	0.995465781
4	87969	356650	0.99206177	0.987347816	0.995933597
5	88070	356548	0.992484295	0.986624276	0.995870613
6	97102	396920	0.987304557	0.989104241	0.995358506
7	92217	401805	0.935392017	0.986737803	0.984802296
8	97681	396341	0.992125741	0.98804271	0.996085195
9	97605	396417	0.991087491	0.987777266	0.995830145
10	97399	396623	0.98720176	0.985985482	0.994716834

From the cross-validation of the KDD Cup-99 dataset for classifying by Decision Tree, Model 10 is the best model that provides the most accurate results in anomaly detection which was taken as training file for experiment and the results as show in Table 4.7

4.2.2 Results of capture Wireshark from the CTU-13 Dataset by using decision tree classifier.

Table 4.5 Results of validation set of the CTU-13 Dataset by using decision tree classifier.

Each 10% of dataset	Results				
	Normal	Attack	Precision	Recall	Overall Accuracy
1	87	334	0.85555556	0.88505747	0.94536817
2	65	351	0.71428571	0.69230769	0.90865385
3	78	356	0.40601504	0.69230769	0.76267281
4	69	358	0.55339806	0.82608696	0.86416862
5	81	311	0.97530864	0.97530864	0.98979592

6	77	396	0.52985075	0.92207792	0.85412262
7	92	298	0.30909091	0.36956522	0.65641026
8	97	396	0.96875	0.31958763	0.86409736
9	61	387	0.33774834	0.83606557	0.75446429
10	95	396	0.56779661	0.70526316	0.83910387

Table 4.6 Results of train set of the CTU-13 Dataset by using decision tree classifier.

Each 90% of dataset	Results				
	Normal	Attack	Precision	Recall	Overall Accuracy
1	897	3343	0.885892116	0.95206243	0.963915094
2	838	3607	0.88589212	0.9725537	0.981327334
3	532	3608	0.93142857	0.89285714	0.6968599
4	483	3636	0.4839211	0.68115942	0.59310512
5	560	3658	0.37774176	0.95714286	0.96728307
6	312	3710	0.82461538	0.8525641	0.98533068
7	201	3821	0.95340502	0.87562189	0.99303829
8	400	3622	0.98324022	0.8332423	0.88065639
9	763	3159	0.57411168	0.79541284	0.64660887
10	888	3431	0.87236025	0.97522523	0.89719843

From the cross-validation of the CTU-13 Dataset for classifying by Decision Tree, Model 5 is the best model that provides the most accurate results in anomaly detection which was taken as training file for experiment and the results as show in Table 4.7

Table 4.7 Result of testing for capture Wireshark from the CTU-13 dataset and KDD Cup-99 dataset by using decision tree classifier.

Classifies	Dataset	Results				
		Normal	Attack	Precision	Recall	Overall Accuracy
Decision tree	KDDcup'99	80386	230643	0.969484924	0.730774015	0.924473281

4.3 Results of Fuzzy logic classifier

4.3.1 Results from KDD Cup-99 Dataset by using fuzzy logic classifier.

Table 4.8 Results of validation set of KDD Cup-99 dataset by using fuzzy logic classifier.

Each 10% of dataset	Results				
	Normal	Attack	Precision	Recall	Overall Accuracy
1	87529	242114	0.929748718	0.850192323	0.89552433
2	87536	242044	0.979840091	0.635483894	0.867761088
3	97266	281697	0.929866364	0.458127661	0.767097417
4	97276	86818	0.936925473	0.800572802	0.882411588
5	96693	371036	0.953976089	0.78997549	0.866777674
6	97178	370158	0.968961749	0.785194283	0.845982163
7	91017	395577	0.93562845	0.937351247	0.914964232
8	75317	356865	0.860274129	0.957298765	0.922027736
9	81547	354785	0.931422828	0.922766313	0.92136157
10	87511	241047	0.919543123	0.876380752	0.913677106

Table 4.9 Results of train set of KDD Cup-99 dataset by using fuzzy logic classifier.

Each 90% of dataset	Results				
	Normal	Attack	Precision	Recall	Overall Accuracy
1	86819	355784	0.931639159	0.925426149	0.935465781
2	86856	355955	0.93206177	0.927347816	0.935933597
3	86892	355890	0.932484295	0.926624276	0.935870613
4	96044	395685	0.927304557	0.929104241	0.935358506
5	90994	395520	0.925392017	0.926737803	0.924802296
6	96513	395575	0.922125741	0.92804271	0.956085195
7	96412	395550	0.921087491	0.927777266	0.935830145
8	96034	395378	0.93720176	0.925985482	0.934716834
9	81769	356087	0.92396916	0.928133074	0.924789224
10	81284	356968	0.878418864	0.938771257	0.925679874

From the cross-validation of KDD Cup-99 dataset for classifying by Fuzzy logic, Model 8 is the best model that provides the most accurate results in anomaly detection which was taken as training file for experiment and the results as show in Table 4.12 and results of membership and rules as show in Appendix

4.3.2 Results of capture Wireshark from the CTU-13 Dataset by using fuzzy logic classifier.

Table 4.10 Results of validation set of the CTU-13 Dataset by using fuzzy logic classifier.

Each 10% of dataset	Results				
	Normal	Attack	Precision	Recall	Overall Accuracy
1	40	379	0.40206186	0.975	0.85918854

2	65	343	0.81818182	0.69230769	0.92647059
3	87	375	0.35526316	0.62068966	0.71645022
4	69	401	0.98507463	0.95652174	0.99148936
5	81	356	0.62698413	0.97530864	0.88787185
6	77	334	0.98611111	0.92207792	0.98296837
7	78	333	0.62962963	0.43589744	0.84428224
8	87	349	0.65957447	0.35632184	0.83486239
9	61	487	0.47663551	0.83606557	0.87956204
10	95	320	0.90540541	0.70526316	0.91566265

Table 4.11 Results of train set of the CTU-13 Dataset by using fuzzy logic classifier.

Each 90% of dataset	Results				
	Normal	Attack	Precision	Recall	Overall Accuracy
1	511	3471	0.57423971	0.62818004	0.89251632
2	458	3521	0.52662037	0.99344978	0.8964564
3	697	3125	0.95754717	0.87374462	0.96991104
4	487	3319	0.61610487	0.67556468	0.90462428
5	464	3387	0.52146465	0.89008621	0.88834069
6	568	3289	0.66334165	0.46830986	0.88669951
7	552	3310	0.64054514	0.68115942	0.89979285
8	668	3098	0.87287025	0.99700599	0.97371216
9	601	3109	0.88913043	0.68053245	0.93450135
10	489	3267	0.57071547	0.70143149	0.89243876

From the cross-validation of the CTU-13 Dataset for classifying by Fuzzy logic, Model 4 is the best model that provides the most accurate results in anomaly detection which was taken as training file for experiment and the results as show in Table 4.12 and results of membership and rules as show in Append

Table 4.12 Result of testing by using fuzzy logic classifier.

Classifies	Dataset	Results				
		Normal	Attack	Precision	Recall	Overall Accuracy
Fuzzy Logic	KDDcup'99	90258	201652	0.953483643	0.710482294	0.903842321

Chapter 5
Conclusions and Discussion

5.1 Conclusions

We have developed an anomaly based intrusion detection system in detecting the intrusion behavior within a network from KDD cup 99 and CTU dataset. A fuzzy decision-making module and Decision tree ware designed to build the system more accurate for attack detection, using the fuzzy inference approach and Decision tree classification. An effective set of fuzzy rules for inference approach were identified by dataset of the KDD cup 99 dataset and CTU dataset. First, the definite rules were generated by using attributes from KDD cup 99 dataset. Then, fuzzy rules were identified by sk-fuzzy the definite rules and these rules were given to fuzzy system, which classify the test data. We have used 10-fold cross validation for evaluating the performance of the proposed system and experimentation results showed that the proposed method is effective in detecting various intrusions in computer networks. Finally, proposed system can predict result from both system which as normal or attack and using cross-validation to find which one is the best performance for train dataset.

5.2 Social and Environmental Impacts (Outcome 7)

Result of the propose system may be affect to the social and environmental, such as if system predict result as normal but actually it is attack, this affect can making damage for the system. So, the accuracy is the main affect of this system, it can make affect to the social and environmental more or less depend on it.

5.3 Knowledge and Engineering Understanding (Outcome 11)

This project use knowledge about programming for coding and data mining to proceed the anomaly detection. In the knowledge that we using Python to run the system and knowledge from data mining, In the system, analysis big data is the important thing. Data mining is the main knowledge of the propose system, this knowledge can help system success.

5.4 Problem

Problem of the system is the fuzzy rules, in this problem we must analysis the dataset that it is very large scale to finding the appropriate for range scale for each attribute to generate the rule of fuzzy logic. The accuracy of the fuzzy logic is depend on the rules which them is very complex to complete the rules and can make the system have more accuracy.

5.5 Suggestions and further improvement

1. Increase accuracy of the system.
2. Can detect attack data on real time.

References

[1] KDD Cup 1999. Available on: http://kdd.ics.uci.edu/databases/kddcup 99/kddcup99.html

[2] Hyafil, Laurent; Rivest, RL (1976). “Constructing Optimal Decision Trees”.Information Processing Letters. 5,pp. 15–17.

[3] J. Quinlan, C4.5:Programs for Machine Learning. Morgan Kaufmann, 1993.

[4] Auephanwiriyakul, S. (2014) Intro CI with watermark.

[5] D.Md.Farid, M. Z. Rahman, “Anomaly Network Intrusion Detection Based on Improved Self Adaptive Bayesian Algorithm”, in Journal of Computers, Vol.

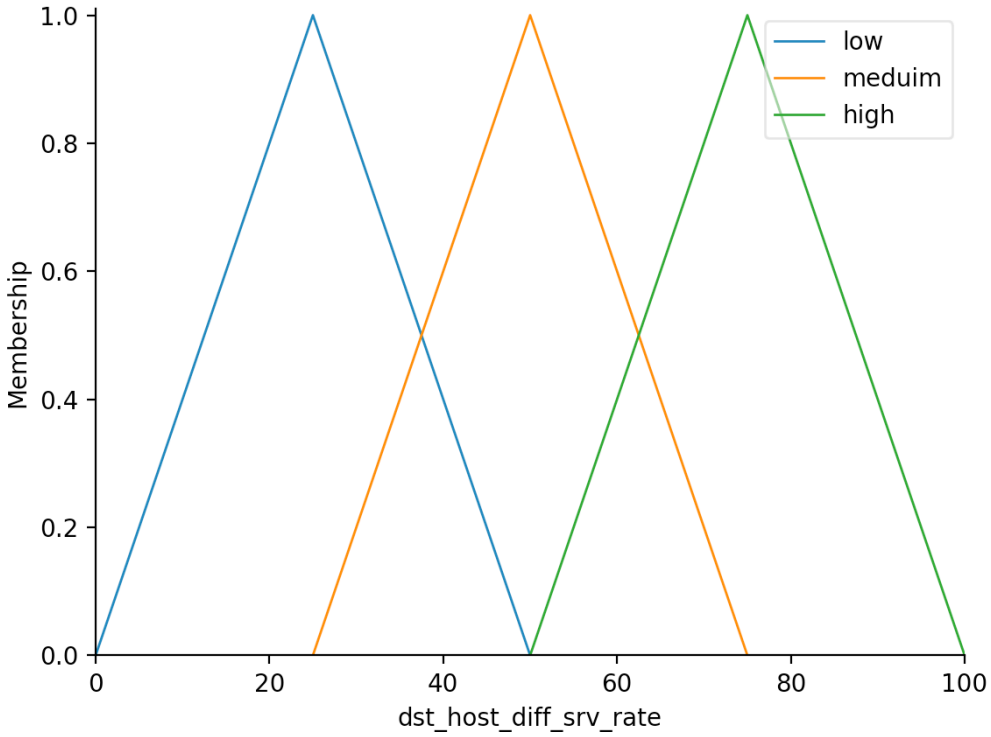
[6] Y.Dhanalakshmi and Dr.I. Ramesh Babu. Intrusion Detection Using Data Mining Along Fuzzy Logic and Genetic Algorithms in IJCSNS International Journal of Computer Science and Network Security, Vol.8

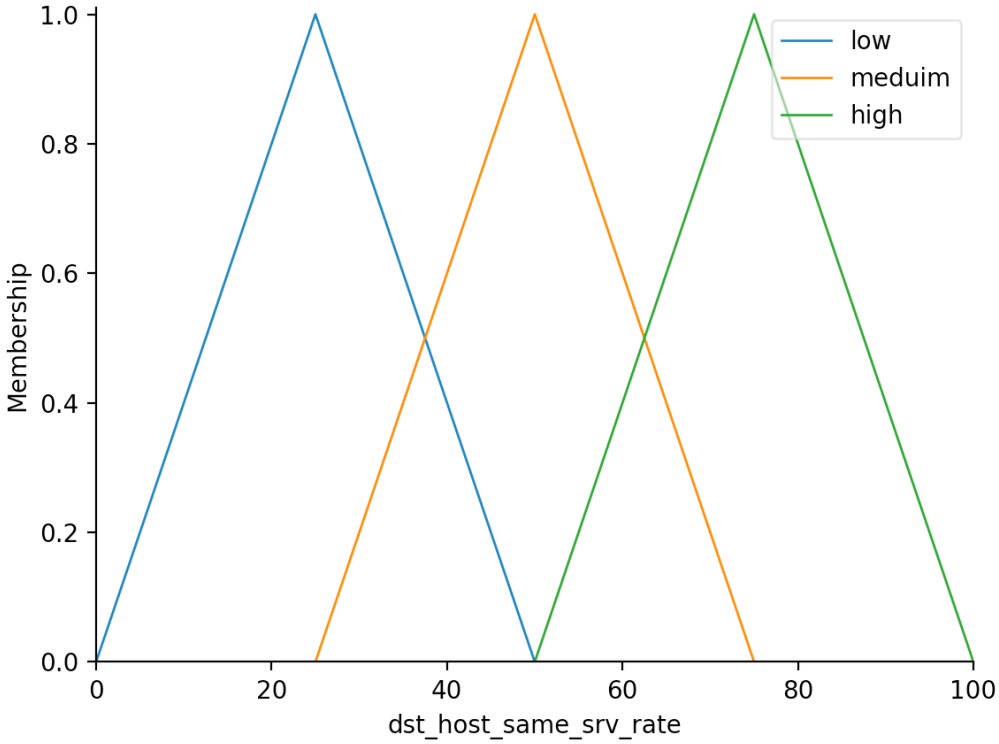
[7] Dernoncourt, F. (2013) ‘Introduction to fuzzy logic’, in Computational Intelligence. Wiley-Blackwell, pp. 19–63.

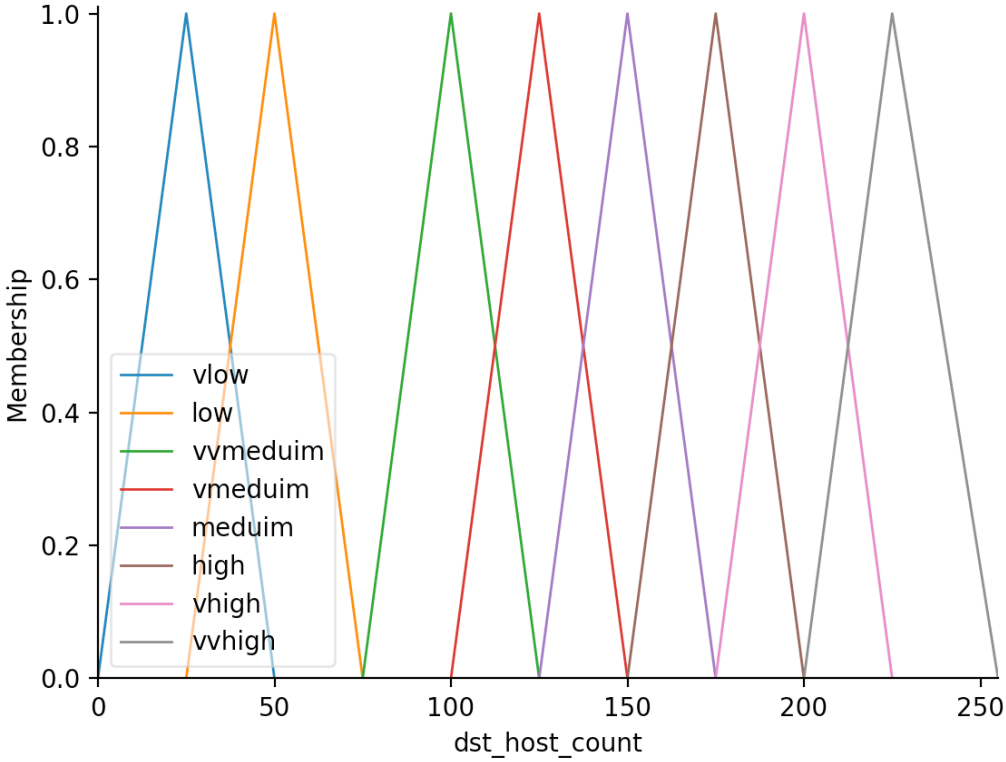
Appendix

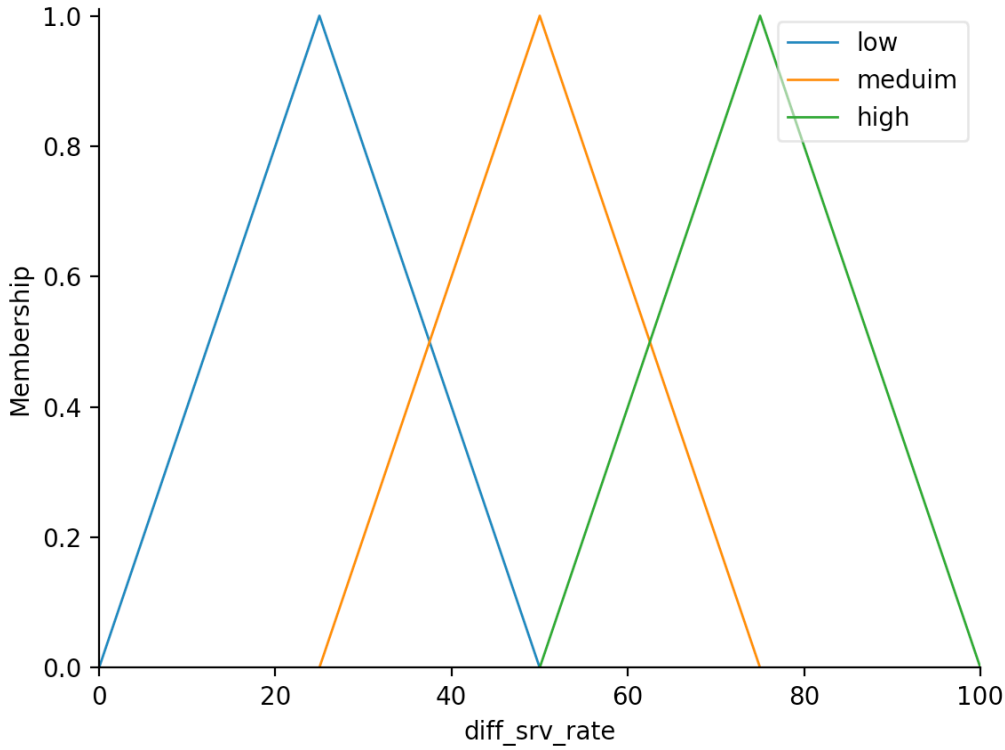
Appendix A

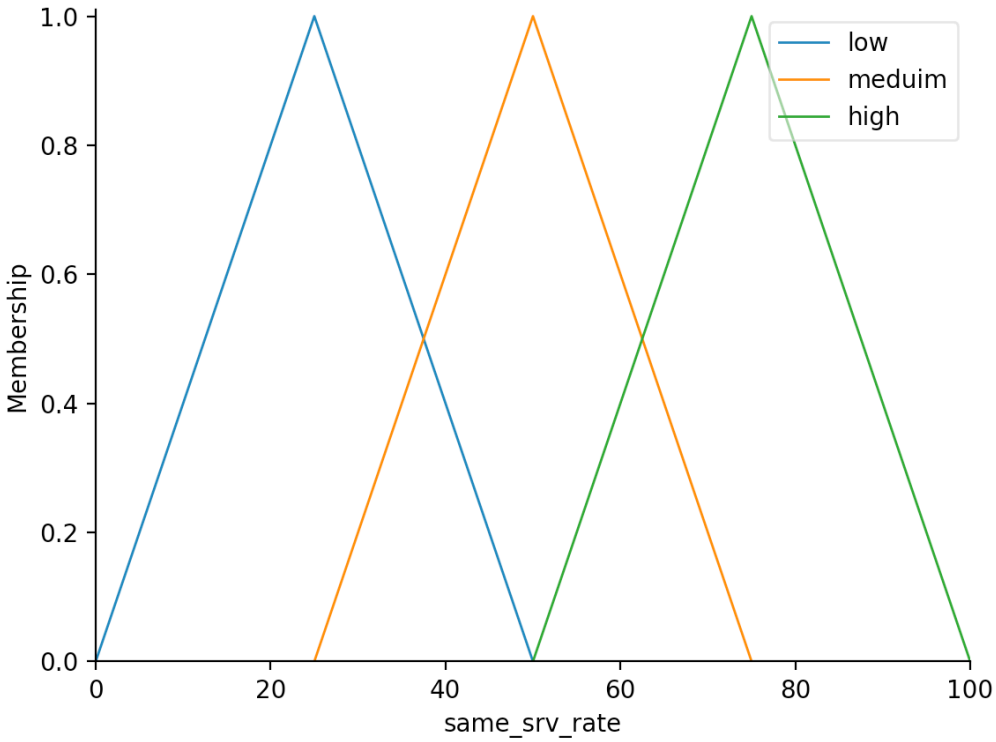
Membership of KDD Cup-99 dataset

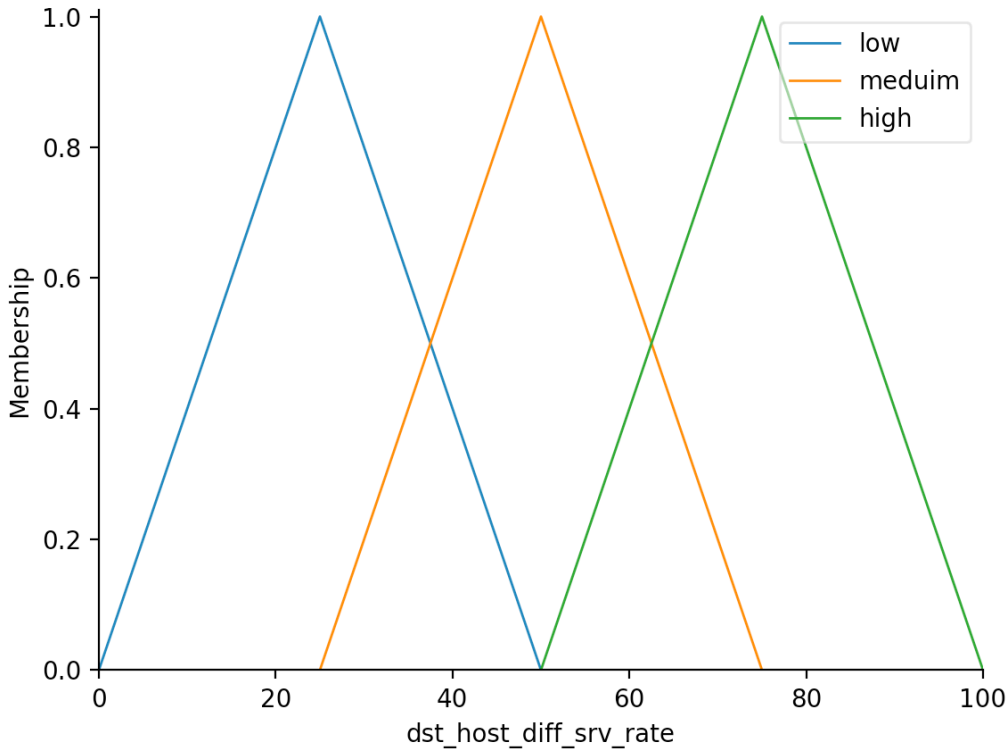


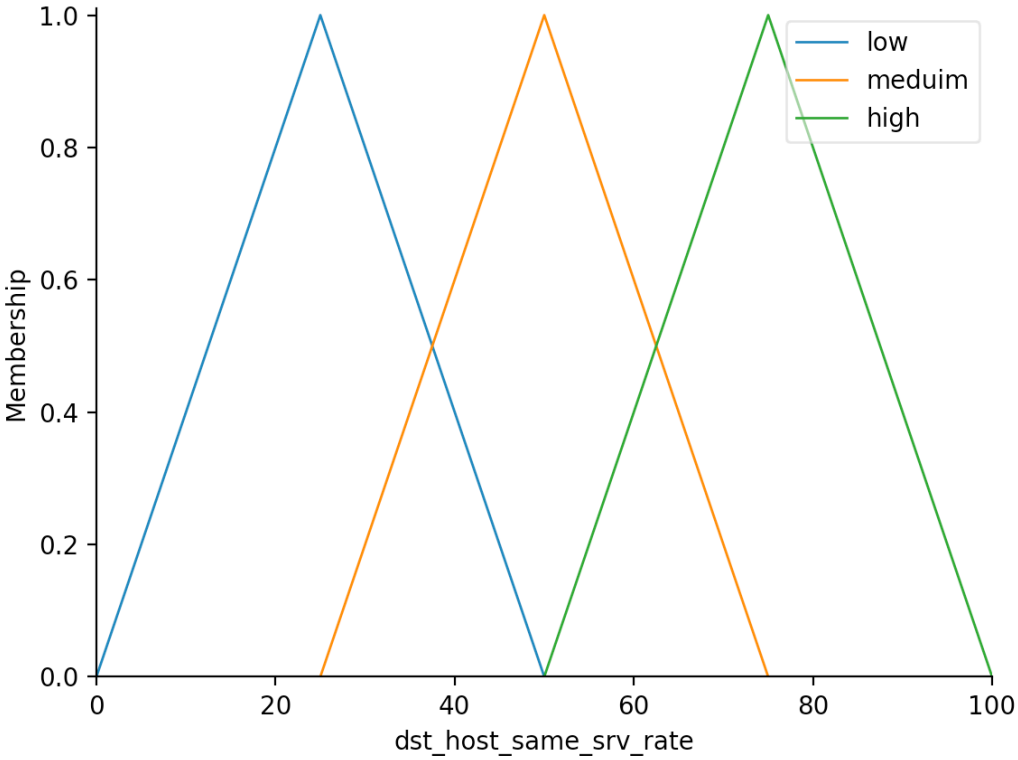


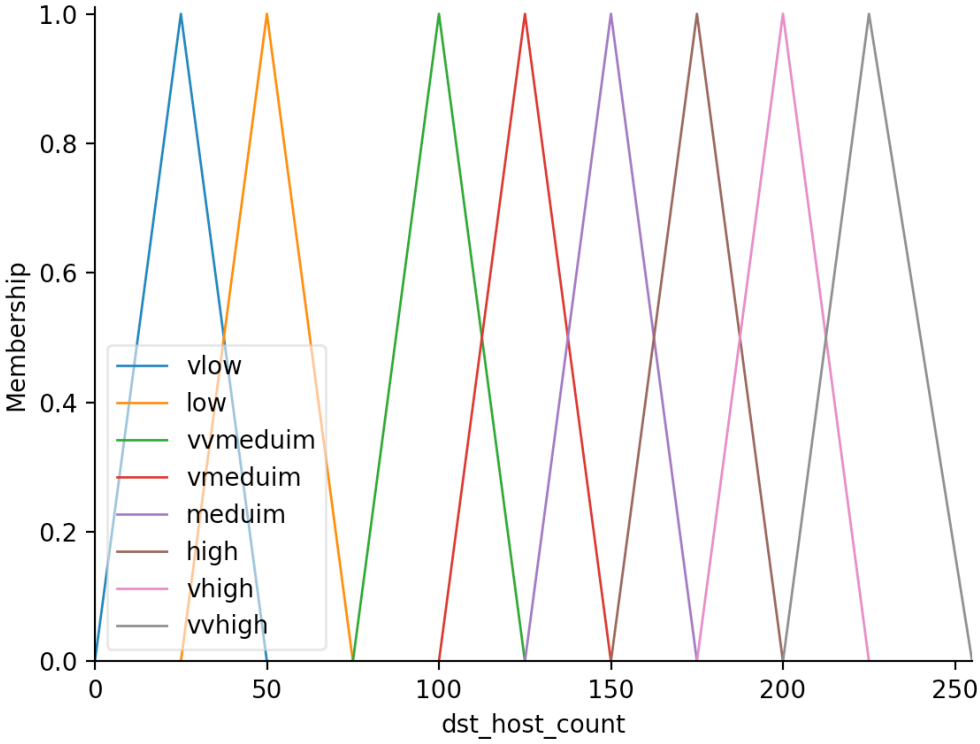

1.0
0.8
0.6
0.4
0.2
0.0
Membership
vlow
low
vvmeduim
vmeduim
meduim
high
vhigh
vvhigh
0
50
100
150
200
250
dst_host_count

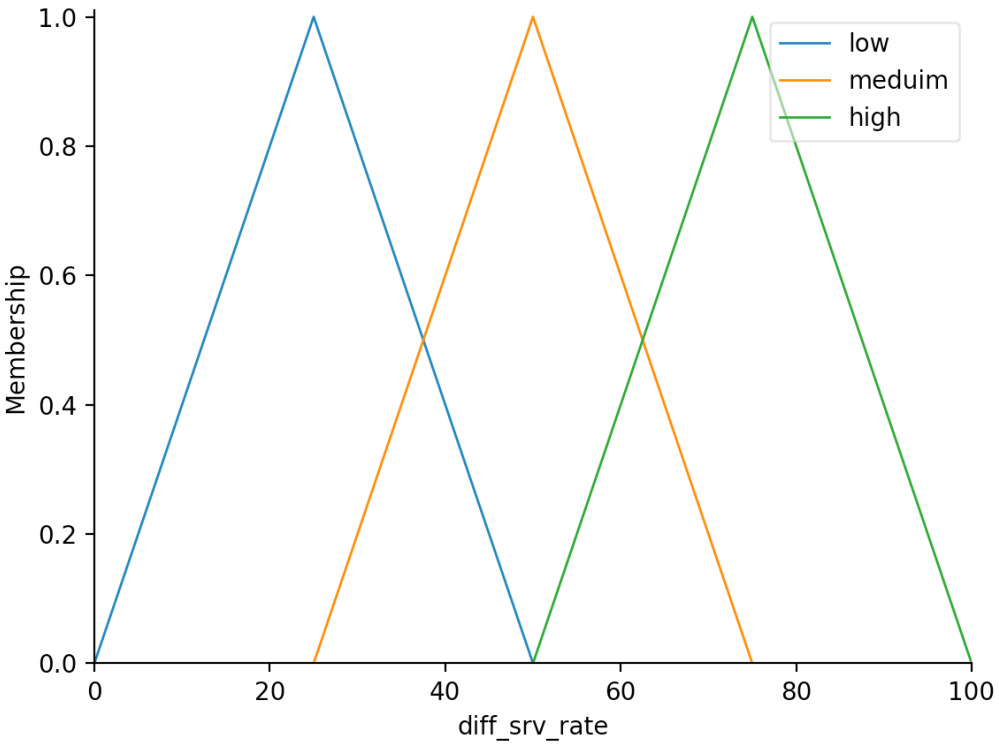

1.0
0.8
0.6
0.4
0.2
0.0
Membership
low
meduim
high
0
20
40
60
80
100
diff_srv_rate

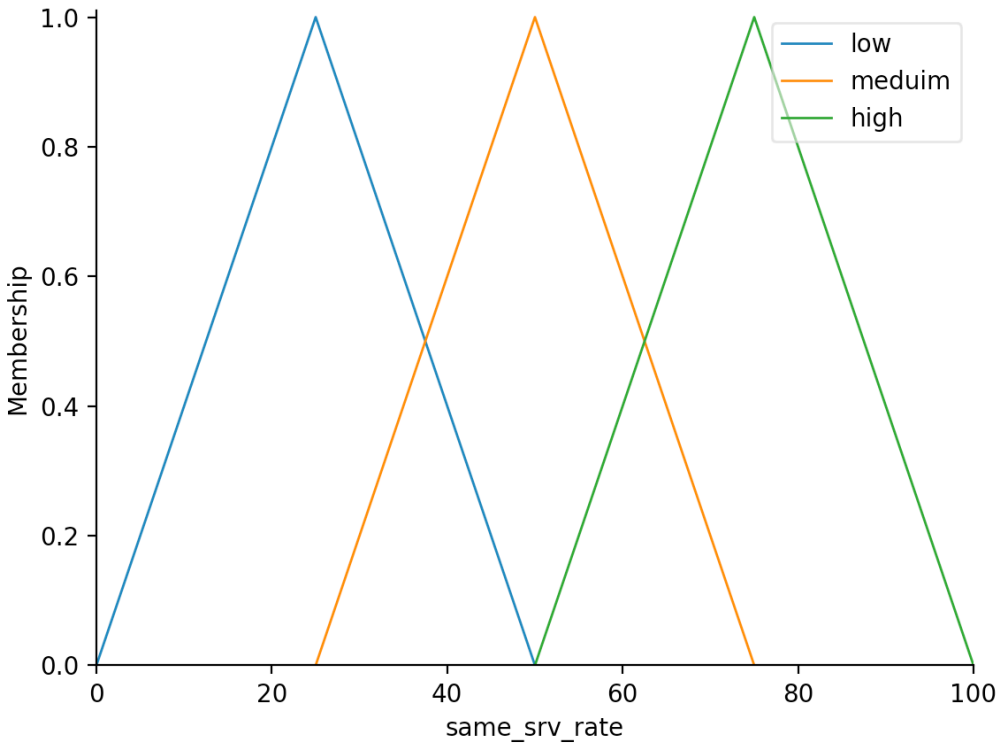

low
meduim
high
Membership
same_srv_rate
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

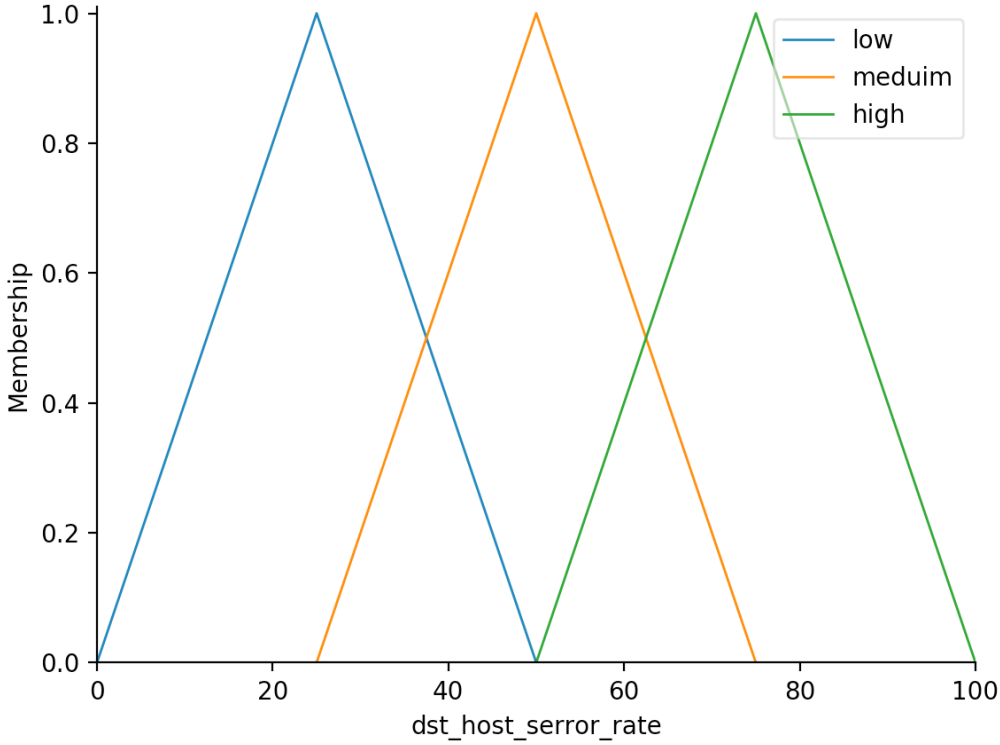

low
meduim
high
Membership
dst_host_diff_srv_rate
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

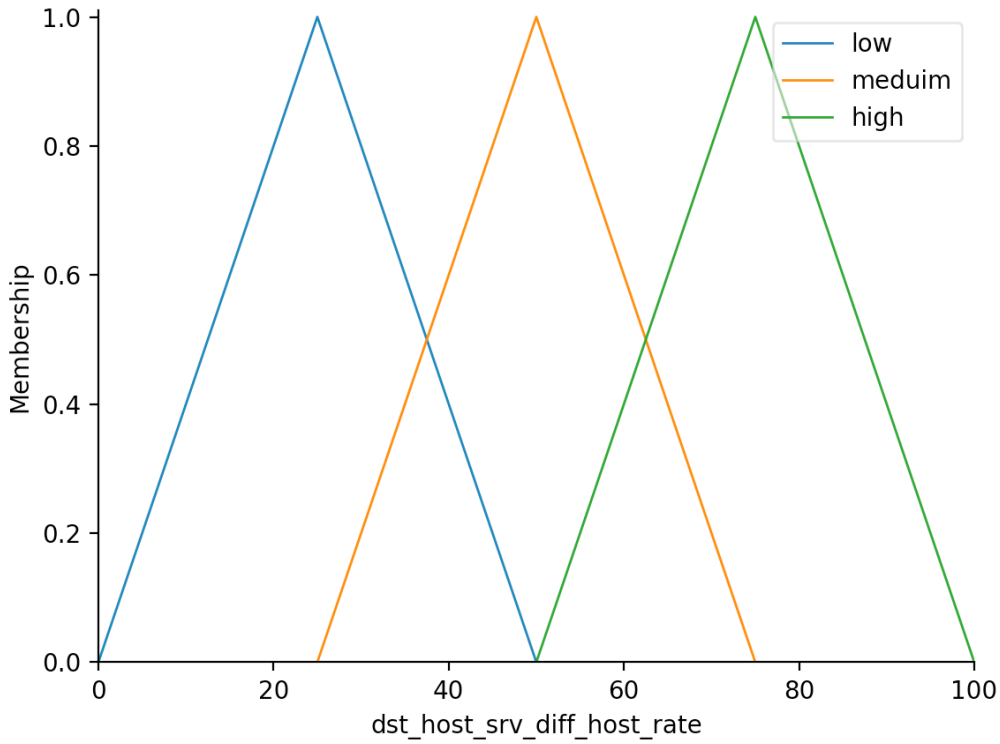

low
meduim
high
Membership
dst_host_same_srv_rate
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

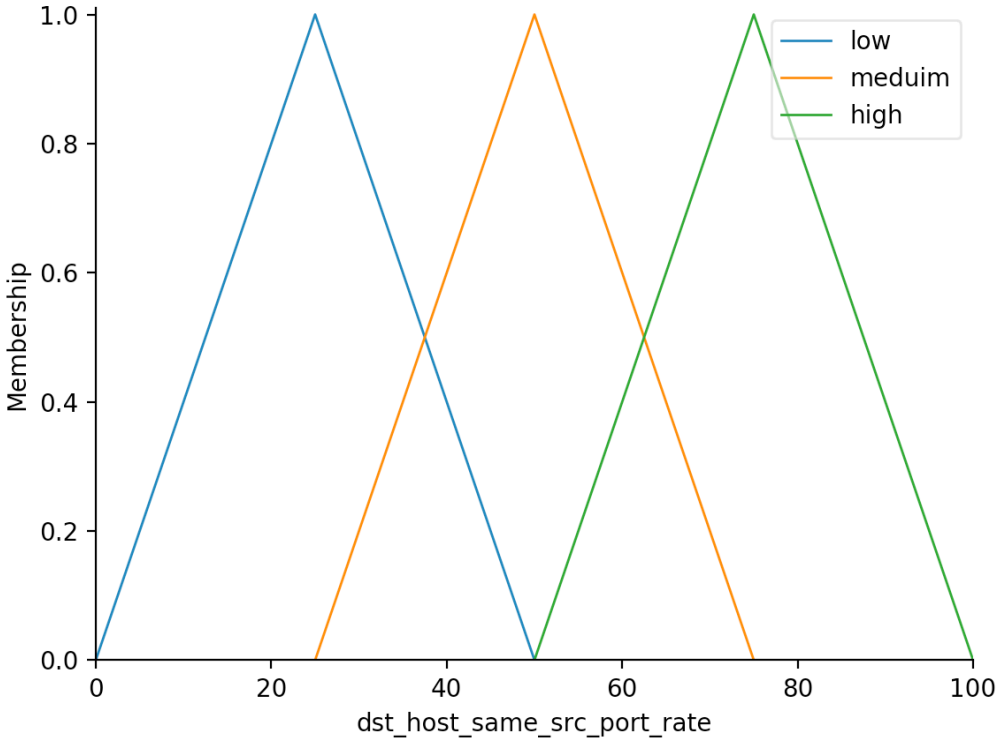

vlow
low
vvmeduim
vmeduim
meduim
high
vhigh
vvhigh
Membership
dst_host_count
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
250

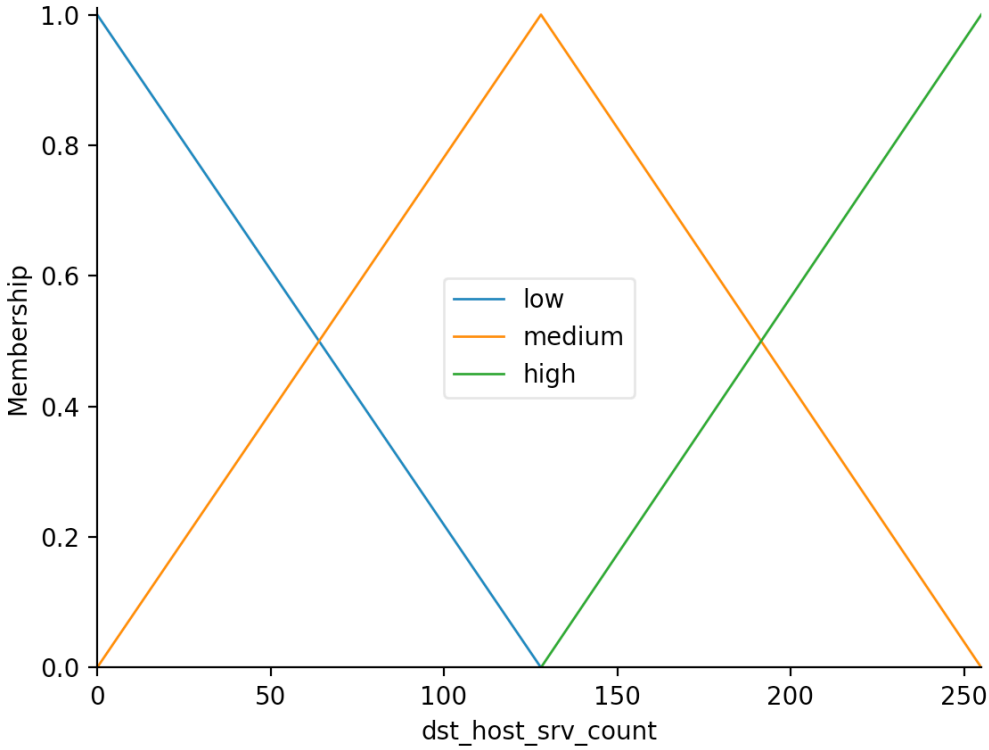

1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
20
40
60
80
100
diff_srv_rate
low
meduim
high

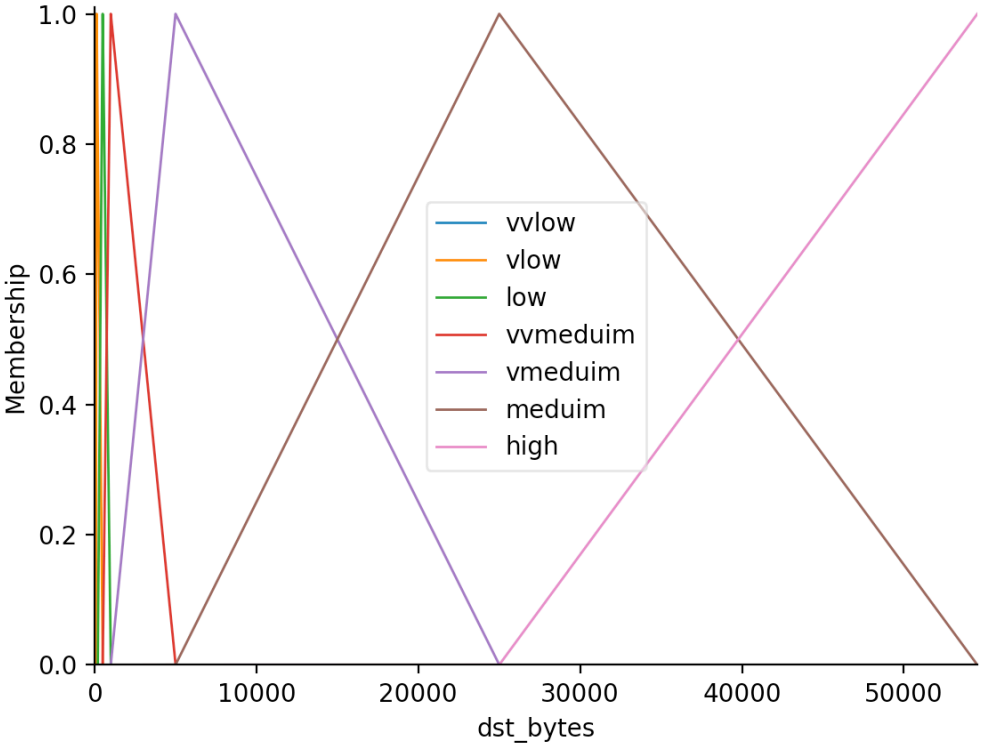

1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
20
40
60
80
100
same_srv_rate
low
meduim
high

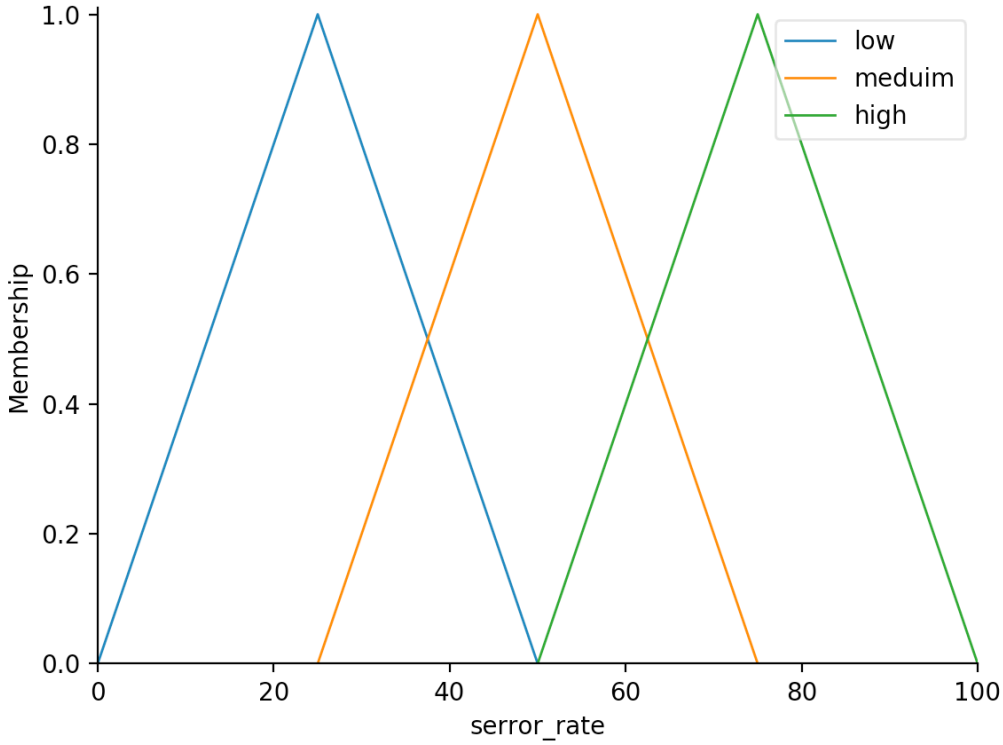

low
meduim
high
Membership
dst_host_serror_rate
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

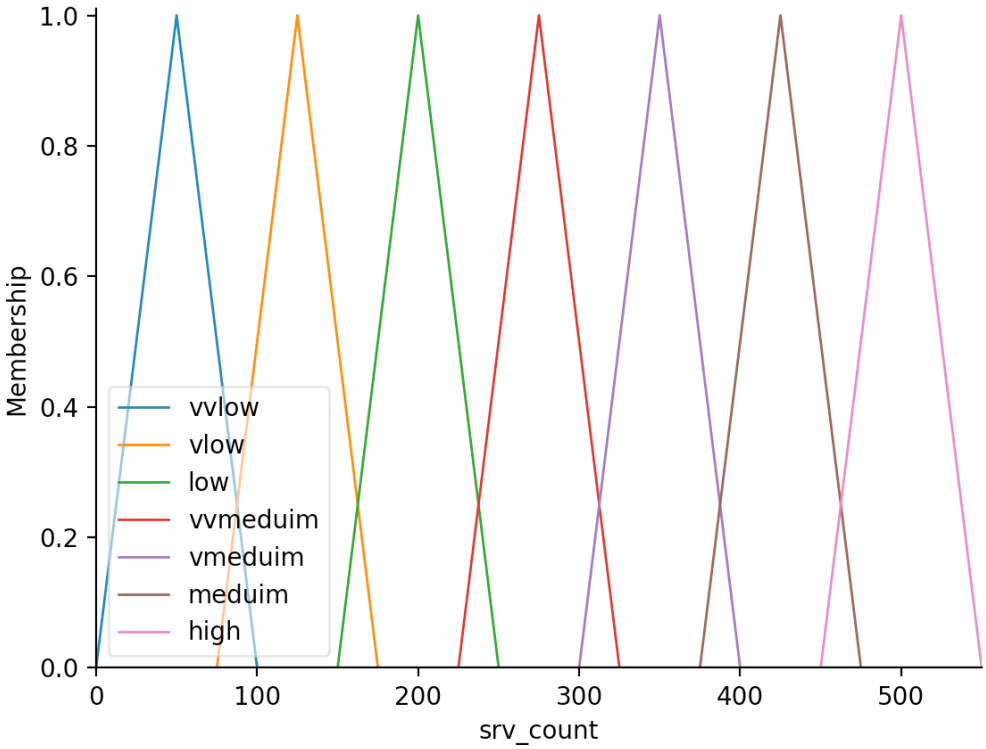

low
meduim
high
Membership
dst_host_srv_diff_host_rate
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

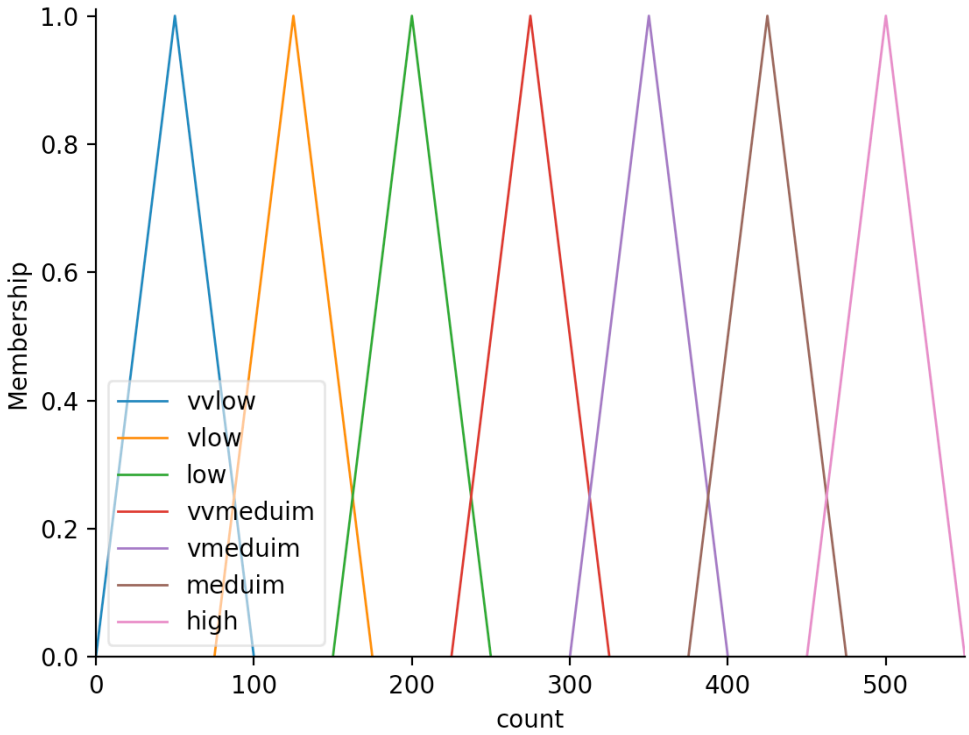

1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
20
40
60
80
100
dst_host_same_src_port_rate
low
meduim
high

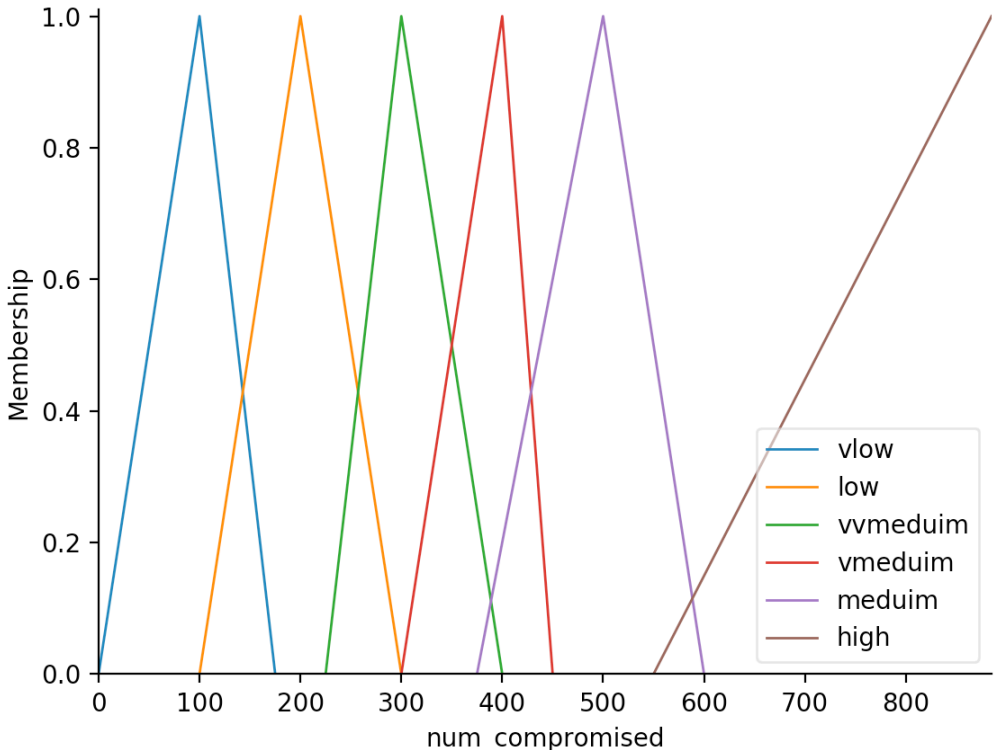

1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
50
100
150
200
250
dst_host_srv_count
low
medium
high

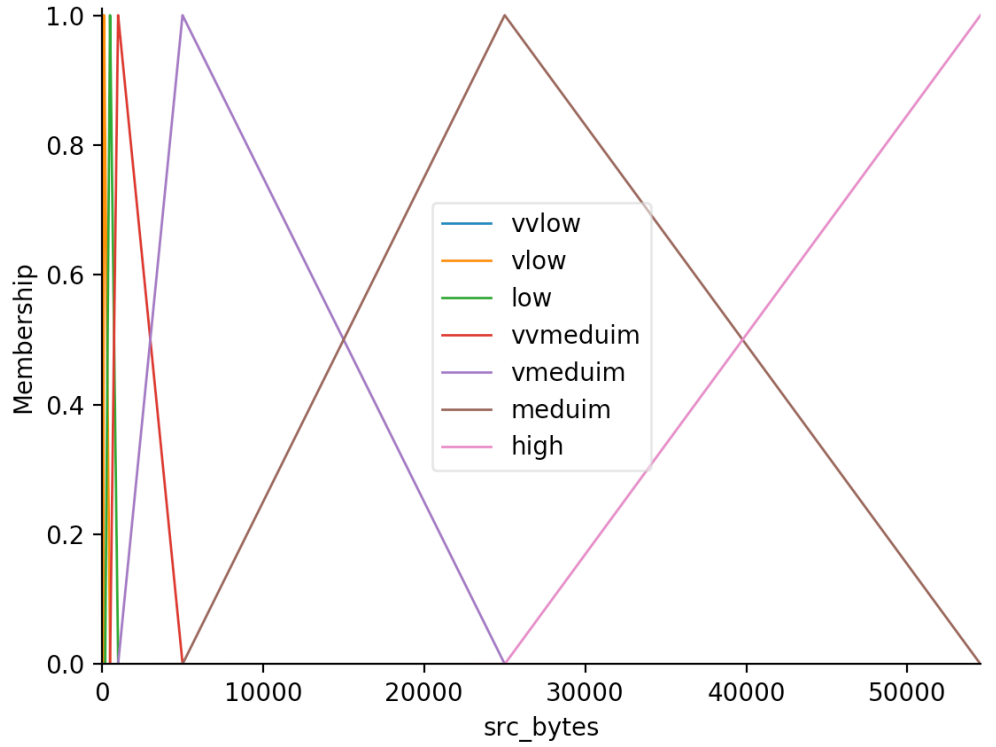

1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
10000
20000
30000
40000
50000
dst_bytes
vvlow
vlow
low
vvmeduim
vmeduim
meduim
high


1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
20
40
60
80
100
serror_rate
low
meduim
high


1.0
0.8
0.6
0.4
0.2
0.0
Membership
vvlow
vlow
low
vvmeduim
vmeduim
meduim
high
0
100
200
300
400
500
srv_count

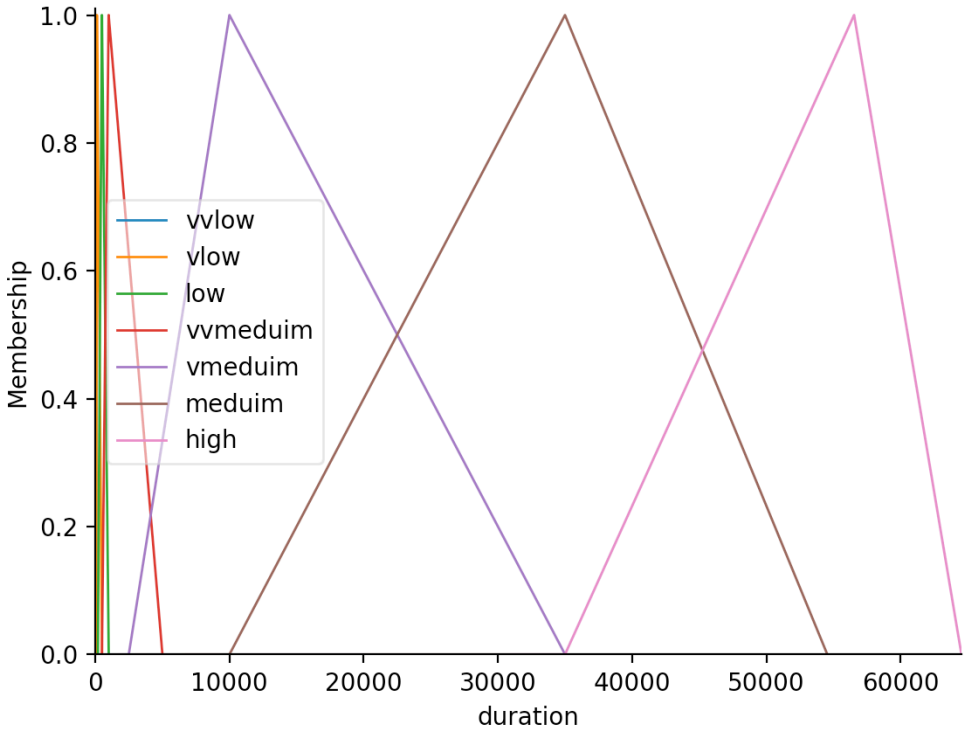

1.0
0.8
0.6
0.4
0.2
0.0
Membership
vvlow
vlow
low
vvmeduim
vmeduim
meduim
high
0
100
200
300
400
500
count

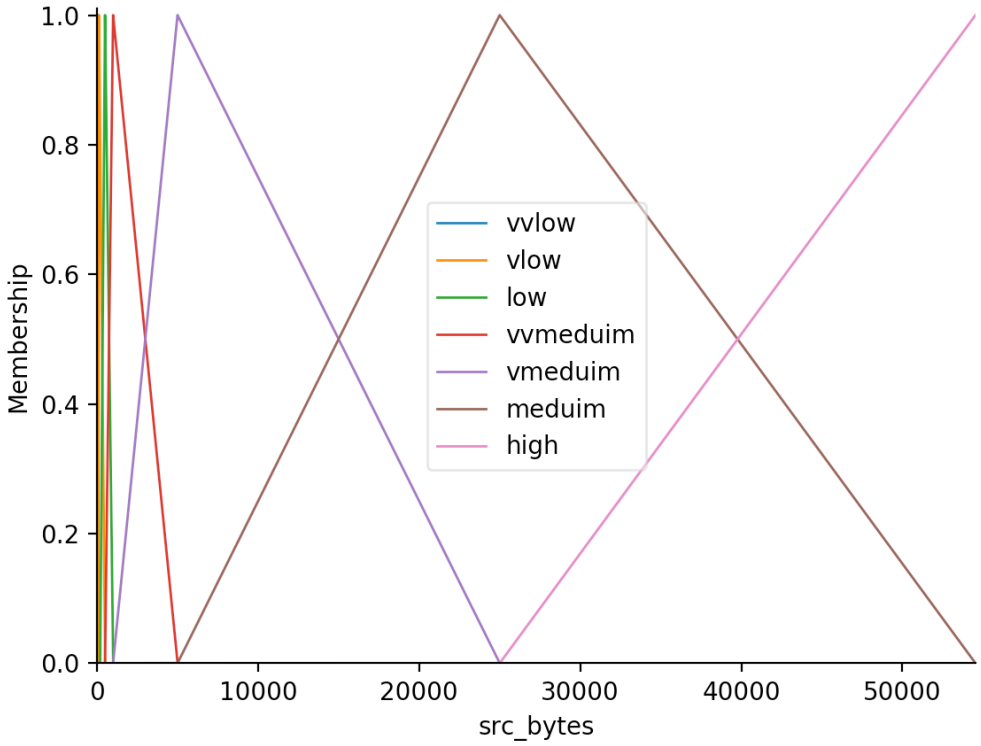

1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
100
200
300
400
500
600
700
800
num_compromised
vlow
low
vvmeduim
vmeduim
meduim
high

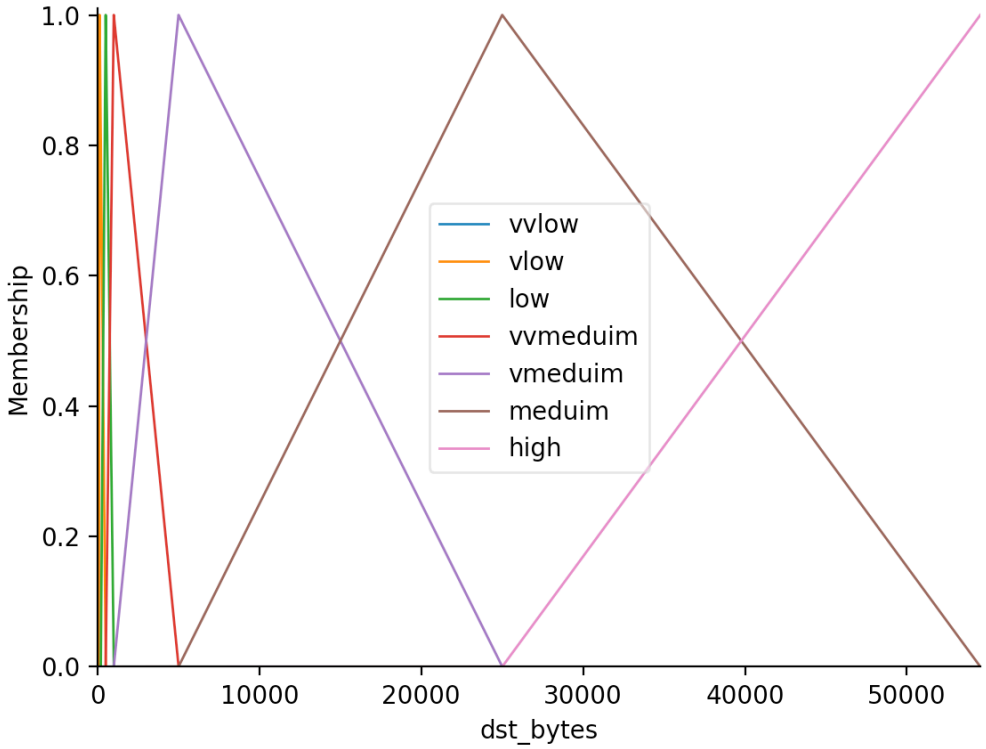

1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
10000
20000
30000
40000
50000
src_bytes
vvlow
vlow
low
vvmeduim
vmeduim
meduim
high

Appendix B

Membership of capture Wireshark from the CTU-13 dataset




1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
10000
20000
30000
40000
50000
src_bytes
vvlow
vlow
low
vvmeduim
vmeduim
meduim
high


1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
10000
20000
30000
40000
50000
dst_bytes
vvlow
vlow
low
vvmeduim
vmeduim
meduim
high

Appendix C
Rule of KDD Cup-99 dataset

Rule1 : If (dst_host_srv_count['low'] | dst_host_srv_count['high'] & dst_bytes['high'], types['Attack']

Rule2 : If (srv_count['high'] & src_bytes['low'] | src_bytes['high'], types['Attack'])

Rule3 : If (num_compromised['high'], types['Attack'])

Rule4 : If(count['vlow'] & dst_host_same_srv_rate['low'] & dst_host_diff_srv_rate['low'] & same_srv_rate['low'] & diff_srv_rate['low'] , types['Attack'])

Rule5 : If (count['low'] & dst_host_same_srv_rate['low'] & dst_host_diff_srv_rate['low'] & same_srv_rate['low'] & diff_srv_rate['low'] , types['Attack'])

Rule6 : If (count['meduim'] & same_srv_rate['high'] & diff_srv_rate['low'] & dst_host_same_srv_rate['low'] & dst_host_diff_srv_rate['low'], types['Attack'])

Rule7 : If (count['high'] & same_srv_rate['high'] & diff_srv_rate['low'] & dst_host_same_srv_rate['low'] & dst_host_diff_srv_rate['low'], types['Attack'])

Rule8 : If (count['meduim'] & same_srv_rate['high'] & diff_srv_rate['low'] & dst_host_same_srv_rate['meduim'] & dst_host_diff_srv_rate['low'], types['Attack'])

Rule9 : If (diff_srv_rate ['high'] & same_srv_rate['high'] & diff_srv_rate['low'] & dst_host_same_srv_rate['meduim'] & dst_host_diff_srv_rate['low'], types['Attack'])

Rule10 : If (dst_host_count['vvhigh'] , types['Attack'])

Rule11 : If (dst_host_same_srv_rate ['low'] | same_srv_rate ['high'] & dst_bytes['high'], types['Attack'])

Rule12 : If (dst_host_srv_count['low'] | same_srv_rate ['high'] & dst_bytes['high'], types['Attack'])

Rule13 : If (src_bytes['high'], types['Attack'])

Rule14 : If (dst_host_srv_count['low'] | dst_host_srv_count['high'] & dst_bytes['high'], types['Attack']

Rule15 : If (serror_rate['high'] & dst_host_srv_count['low'], types['Normal'])

Rule16 : If (dst_host_srv_count['high'] & dst_host_srv_count['low'] & count['low'] , types['Normal'])

Rule17 : If (src_bytes['low'] & dst_host_srv_count['low'], types['Normal'])

Rule18 : If (count['high'] & same_srv_rate['high'] & diff_srv_rate['high'] & dst_host_same_srv_rate['meduim'] & dst_host_diff_srv_rate['high'], types['Attack'])

Rule19 : If (diff_srv_rate ['high'] & same_srv_rate['high'] & diff_srv_rate['low'] & dst_host_same_srv_rate['meduim'] & dst_host_diff_srv_rate['high'], types['Attack'])

Rule20 : If (dst_host_count['vvhigh'] & dst_host_srv_count['low'] , types['Attack'])

Rule21 : If (dst_host_same_srv_rate ['low'] | same_srv_rate ['high'] & dst_bytes['high'] , types['Attack'])

Rule22 : If (dst_host_srv_count['high'] | same_srv_rate ['high'] & dst_bytes['high'] & dst_host_srv_count['low'], types['Attack'])

Rule23 : If (src_bytes['high'], types['Attack'])

Rule24 : If (dst_host_srv_count['high'] | dst_host_srv_count['high'] & dst_bytes['high'] & diff_srv_rate ['high'], types['Attack']

Appendix D
Rule of capture Wireshark from the CTU-13 dataset

Rule1 : If (duration ['low'] | src_bytes ['high'] & dst_bytes['high'], types['Attack']

Rule2 : If (src_bytes ['high'] & duration ['low'] | dst_bytes ['high'], types['Attack'])

Rule3 : If (duration ['high'], types['Attack'])

Rule4 : If(src_bytes ['vlow'] & duration ['low'] & dst_bytes ['low'] , types['Normal'])

Rule5 : If (src_bytes ['low'] & duration ['low'] & dst_bytes ['low'], types['Normal'])

Rule6 : If (src_bytes ['meduim'] & duration ['high'] & dst_bytes ['low'] ,types['Normal'])

Rule7 : If (duration ['high'] & dst_bytes ['high'],types['Attack'])

Rule8 : If (src_bytes ['meduim'] & duration ['high'] & dst_bytes ['low'] | dst_bytes ['meduim'] &, types['Normal'])

Rule9 : If (src_bytes ['high'] & duration ['high'] & dst_bytes ['meduim'] | dst_bytes ['high'], types['Attack'])

Rule10 : If (duration ['vvhigh'] & src_bytes ['high'], types['Attack'])

Rule11 : If (src_bytes ['low'] | src_bytes ['meduim'] & dst_bytes['high'], types['Attack'])

Rule12 : If (duration ['low'] | src_bytes ['high'] & dst_bytes['high'], types['Attack'])

Rule13 : If (duration ['high'] & dst_bytes ['high'], types['Attack'])