Using Rough Set Theory to Find Minimal Log with Rule Generation

Tahani Nawaf E`layan Alawneh

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
January 2022
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Applied Mathematics and Computer Science.

Asst. Prof. Dr. Mehmet Ali Tut
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev
2. Prof. Dr. Hamza Erol
3. Prof. Dr. Efendi Nasiboğlu
4. Asst. Prof. Dr. Müge Saadetoğlu
5. Asst. Prof. Dr. Mehmet Ali Tut

ABSTRACT

The current research aims to enhance real-time analysis and speed of security algorithms by focusing on creating a minimal security log with the same efficiency as the original one through keeping core knowledge. Hence, manipulating this created minimal log with any security algorithm will surely enhance execution time and accordingly push towards real-time analysis. Rough set concepts are being used to reduce log size by using algorithms of the RoughSets package in R language, these algorithms guarantee to find a minimum log at least, and to find a minimal log at most depending on hardware resources available. Embedded knowledge is being extracted from minimum and minimal logs in the form of IF...THEN statements, and finally conclusions were discussed.

Keywords: hate speech, text classification, classifier, classifier ensembles, stacking ensemble, text mining, genetic programming, pattern classification.

ÖZ

Mevcut araştırma, temel bilgileri koruyarak orijinali ile aynı verimliliğe sahip minimum bir güvenlik günlüğü oluşturmaya odaklanarak güvenlik algoritmalarının gerçek zamanlı analizini ve hızını artırmayı amaçlamaktadır. Bu nedenle, oluşturulan bu minimum günlüğü herhangi bir güvenlik algoritmasıyla manipüle etmek, yürütme süresini kesinlikle artıracak ve buna bağlı olarak gerçek zamanlı analize doğru itecektir. RoughSets paketinin algoritmalarını R dilinde kullanarak log boyutunu küçültmek için kaba küme konseptleri kullanılmaktadır, bu algoritmalar mevcut donanım kaynaklarına bağlı olarak en az minimum log, en fazla minimum log bulmayı garanti etmektedir. Gömülü bilgi, IF...THEN ifadeleri biçiminde minimum ve minimum günlüklerden çıkarılıyor ve son olarak sonuçlar tartışıldı.

Anahtar Kelimeler: kaba küme teorisi; denklik sınıfları; daha düşük yaklaşım; üst yaklaşım; R dili.

DEDICATION

This is in memory of my beloved father Nawaf, who had always loved me unconditionally. Although he was unable to see my graduation, this is for him. A special feeling of gratitude to my loving mother Sameera whose words of encouragement and push for tenacity ring in my ears:" Nothing like the happiness of graduation".

To my very special sisters Amani and Sally who have never left my side and stand by me when things look black. To the greatest brother ever Rami for having the heart of my father. To my beloved brothers Sami, Hani and Gazi who have supported me always.

To my husband Osama, and my precious kids Abdulrahman, Maria and Juman who have been a constant source of support and encouragement during the challenges of graduate school and life.

To the light of hope Amal and her family Anas, Fatema and Aysha, for being my family in Cyprus. To the kindest friends ever Mona and Chima, I was lucky for being your friend. To my friends who supported and encouraged me through this journey Bashar and Laith.

To all of you, I am truly thankful for having you in my life, I love you all beyond words.

ACKNOWLEDGMENT

To my supervisor Assist. Prof. Dr. Mehmet Ali TUT and Prof. Dr. Rashad Aliyev who helped and guided me to successfully complete this thesis.

To all Academic Staff in mathematics department for their dedication in teaching us.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Information system security has been achieved using several security solutions such as IDS, IPS, anti-viruses and firewalls, etc. Each device will work independently to guarantee appropriate access to network resources [1–4], and will generate its own alert logs, bearing in mind that these logs are growing too rapidly to be covered under the terminology big data. High repetitions and false alerts are common in such logs. As a result, this may mislead the process of identifying real threats. This poses a difficulty in analyzing large logs and detecting serious security issues and intrusions, as well as reacting at the right time.

In this Chapter the basic concepts of log data generation, collection, and analysis will be explained.

1.1 Log Data

At the beginning log data has to be defined, it is the output of any computer system, hardware, program, or other device in reaction to external stimuli [1, p. 14-16]. What the stimuli are relies a lot on where the log message came from. Unix systems, for example, have user login/ logout messages, firewalls have Access Control Lists (**ACL**) accept/deny messages, while disk storage systems create log messages whenever they fail or, in some circumstances, when systems fail.

A log message's fundamental meaning is referred to as log data. Alternatively, log data refers to information extracted from a log message that explains the reason a log message was created. When someone accesses any resource (picture, file, etc.) on a Web page, for example, a Web server will frequently log it. The user's name would appear in the log message if he had to authenticate himself to access the page.

The phrase "logs" refers to a group of log messages which will be utilized to create the picture of the event occurrence. The following are the broad categories that log messages fall into:

- Informational messages: are intended to inform users and system administrators that something good has happened. When the operating system (OS) is rebooted, for example, Cisco IOS will directly send out notifications. However, caution is needed. If a reboot occurs outside of typical maintenance or normal working hours, for example, you may be concerned.
- Debug messages: are usually created by software systems to assist software developers in debugging and identifying issues with executing application code.
- Warning messages: are often used in instances where something is missing or required for a system, but its absence has no influence on the system's operation. For instance, if a certain program isn't provided the correct number of parameters related to a programmed command line but can still execute without them, the program may generate a warning message to the operator or user.
- Error messages: are used to communicate errors which occur at different computer system levels. When an operating system can't synchronize buffers to disk, for example, an error message is generated. Most error messages,

unfortunately, simply provide a reference point for determining why they happened. In attempt to get to the source of the problem, more investigations are frequently necessary.

- Alert messages: is a notification that something interesting has occurred. Security devices as well as security systems are the core of such alerts in general, although this is not a hard rule. On a network, an Intrusion Prevention System (**IPS**) may be installed in the middle, examining all traffic coming. depending on the details of the data packets, it will determine whether a specific network connection is permitted or not. If **IPS** detects a potentially harmful connection, it can execute any of a group of predefined actions. The detection, as well as the action done, will be recorded as a log.

1.2 Log Data Collection and Transmission

The transmission and collection of log data are theoretically straightforward. A logging subsystem is implemented in a device or computer so that it can create a message whenever it sees fit [2, p.49-51] . The method used to create such messages varies depending on the device. You may, for example, be able to customize the device itself or it may be programmed to send a pre-determined message list. This location is generally called a log-host. A log-host is a system that collects log messages in a central place, usually a Unix or Windows server. The following are some of the benefits of utilizing a central log message collector:

- Having a centralized repository for storing log messages collected from several sites and locations.
- Having a place for storing logs backup copies.
- Having a location where analysis on log data is being performed.

The question here, how to transmit log messages? Syslog protocol is the most often used method, it forms a standard method for exchanging log messages. It is most often seen on Unix computers, although it is also available for Windows as well as other non-Unix environments. However, there is essentially a client - server component that is implemented via the User Datagram Protocol (**UDP**), but it worth to mention that many commercial Syslog and open-source applications support the Transmission Control Protocol (**TCP**) for assured delivery. The physical computer system or device that creates and sends log messages is referred to as the client part. Typically, the server part is found on log collecting server, its primary function is to accept Syslog-based log messages and save them on its disk storage so they can be backed up, examined, and kept for long-term usage.

Moreover, Syslog is not really the only mean used for transmitting and collecting log data. Microsoft, for example, has their very own Windows logging system. It is known as the Windows Event Log. Things like application messages, user logins and logoffs, and other data are saved in a private storage format. There are commercial and open-source applications that operates depending on Event Log that converts event log records to Syslog, and hence transmits them to the syslog server.

Now, there exist a protocol called Simple Network Management Protocol (**SNMP**) it is used to manage devices all over the network. This protocol is standard, and it is built depending on traps/polling concepts. Traps are simply sort of log messages a computer system or device produces when something unusual occurs. Traps are transmitted to a host system, which functions similarly to a loghost. Polling occurs when ever a management station can use **SNMP** for asking a device for variables which are pre-defined

such as bytes transferred through an interface, interface statistics, etc. One important distinction between Syslog and **SNMP** will be that **SNMP** is intended to be organized in terms of data format. However, this has not been the case in practice, meanwhile it is strict in Syslog.

Recently, a convenient way to save applications log messages was through using databases. Simply, applications will write the generated log messages to a database server rather than creating Syslog messages. This has several advantages, particularly in terms of providing a systematic method for saving, analyzing, and creating reports on the stored log messages.

Lastly, there are another loggings format to consider. These formats come from third-party programs and devices that use their own private methods to create and collect log messages. In this case, the provider either offers an Application Programming Interface (**API**) in the format of Java/C libraries, or you must build the protocol your own self. Event Logs for Windows is a proprietary format, it is frequently considered as informal logging standard as Syslog, because it is widely adopted. Protocols discussed so fare:

- Syslog: client/server protocol runs on top of **UDP**. This has been the most popular and widely used logging method.
- **SNMP**: was initially designed to manage networked devices. However, various non-networked devices have used **SNMP** as a means of emitting log messages and status data over time.
- Windows Event Log: Microsoft's own message log format.
- Database: structured method of storing and retrieving log messages.

- Common Private Protocols:
    - **LEA**: Log Extraction API (**LEA**) collects logs from security solutions and appliances such as firewalls.
    - **SDEE**: Security Device Event Exchange (**SDEE**) protocol is based on eXtensible Markup Language (**XML**) and is used by Cisco to collect log messages from its **IPS** devices.
    - E-Streamer: E-Streamer is a protocol owned by Sourcefire and used by their Intrusion Prevention System (**IPS**).

1.3 Log Message

As previously stated, a log message is anything created by a system or device to indicate something has occurred. But what exactly the structure of a log message? To begin, the following are the typical fundamental contents of any log message [3,p.97-101]:

- Data.
- Source.
- Timestamp.

It makes no difference whether the message is transmitted through Syslog, collected by Microsoft Event Log, or saved in any database. These fundamental elements are always included in the message. A log message's heart is its data. Unfortunately, there is no common template for representing data in such a logging message. Some of the most typical data elements seen in a log message are source/destination ports, source/destination IPs, program names, bytes transferred both in or out, resource object (such as directories, files, and so on), user names, etc. A system which created a

log message is a source. This is usually provided in the form of an IP address or a host name. Finally, the timestamp indicates when the log message was issued.

The precise format of any log message is determined by the way the log source has configured its log system. As previously said, Syslog is the widely used format by computer systems and devices. As an example, consider the following Syslog message [4, p.53]:

Jul 13 19:17:23 10.240.46.16 15: *Mar 1 00: 16:17 %LINEPROTO-5-UPDOWN: line protocol on Interface FastEathernet 0\0, changed state to up

This message was created by Cisco router then stored on Syslog server. “Jul 13 19:17:23” indicates the timestamp indicating when the server received the message. While, “10.240.46.16 “indicates the Cisco router IP. Now, what comes after the colon is the data. “%LINEPROTO-5-UPDOWN” is the interpretation of the or event which depends on the vendor, and so a vendor should provide users with a suitable information manual to understand this part. This part of the log message has also a time stamp, but this for the router itself, and most probably indicating Time/date issue, either the server or the router clock has a problem. Absence of time synchronization between different network devices and computer systems will generate data inconsistency problem regarding the log, that will cause a problem in analyzing the data log.

1.4 Log Ecosystem

Log ecosystem, also known as the log infrastructure, is comprised of all the pieces and components that work together to enable creating, normalizing, filtering, analyzing, and storing log data for long-term. Eventually, the objective of Log ecosystem is to assist you to solve issues using logs. As mentioned so far, logs are generated by a variety of sources, including:

- Unix/Windows operating systems.
- Switches.
- Firewalls.
- Routers.
- Virtual Private Network (**VPN**) Server.
- Anti-virus solutions.
- Wireless access points.
- Printers.

The list could go infinity. The lesson is that almost every application, computer system, and devices on the network is able to record logs. You only need to figure out where to search. Aside from that, you have to configure your network devices to generate logs. Now it is the time to treat log messages by filtering and normalizing. Filtering is the process of including or omitting log messages depending on their content. Some sources enable this natively, while others may need the deployment of an agent, which intercepts log data and filters them depending on predefined criteria. Normalization is the process of transforming disparately structured log messages to a consistent format [3,p.91]. A log message after normalization is generally referred to as an event. However, the final destination of an event is often relational databases, where analyzing and generating reports may be conducted. Here, the importance of having log data with common format shows up, this will facilitate data manipulation and deriving meanings from it. It should be emphasized that normalization occurs independently of the protocol or source adopted (i.e. Syslog, database, **SNMP**, etc.).

Now we should mention her that the data part of the log message contains a priority entry, this notion is important for normalization process. A fundamental technique is

to transfer the direct and indirect priority of a log message to a certain common scheme. As an example, consider the low, middle, and high scales, as described below:

- Low: informative messages in nature and do not require immediate attention.
- Medium: those messages that have to be addressed in such a timely basis, but not absolutely immediately. Whenever the engine recognizes malicious intent, IPSs, for example, have the ability to prohibit network traffic. This is called an action. If the IPS sends a logging message to this action, you'll know the traffic was banned and may investigate into it when you can.
- High: High priority situations necessitate quick response. For example, router restarts outside of authorized maintenance intervals, alerts of IPS engine for possible data theft, network devices dropping off the infrastructure for a longer duration, and so forth.

But how does normalization take place? let's show a basic example. The below is a Sourcefire **IPS** message in a Syslog format [4,p.58]:

Jul 17 11:54:38 Source-Fire SFIMS: [1:479:2] ICMP -PING -NMAP [Classification: Attempted Information Leak] [Priority: 3] {ICMP} 210.22.216.78 -> 68.126.152.137

This message contains a lot of information. We employ a method known as parsing to normalize its information. Parsing includes reading the logging message from beginning to end to extract information of interest and inserting it into normalized attributes within the event. The following are some of the most frequent fields used in the normalization:

Timestamp: July 17 2017, 11:54:38

category: Attempted Information Leak

Protocol: ICMP

Priority: High

Source Port: NULL

Destination Port: NULL

Source IP : 210.22.216.78

Destination IP : 68.126.152.137

Chapter 2

PRELIMINARIES

Generally, the process of extracting knowledge from event record files in computer systems is called log analysis. However, computer systems generate a large number of log events with a high rate of repetition. As a consequence, the goal of the log analysis process is detecting serious security issues and intrusions out of repetitive false ones to protect such systems [1].

Information system security is a critical and a wide concept, as it governs and controls all processes, tools, and appliances used to protect critical information from unauthorized actions or access, precisely as the new definition 'Appropriate Access' aims [2]. Hence, and depending on the concepts of cloud technology, web systems, Internet of Things (IoT), and the new demands and challenges forced by the Corona Pandemic; where the adoption of online concepts spread in almost all our daily activities, tremendously huge amount of security logs are being generated and need to be analyzed and understood. In addition, proper remediation actions must be performed on time to guarantee efficient risk elimination, especially that we are moving recently to real-time Intrusion Detection Systems (IDS).

Such a task forces great pressure on information security officers and system administrators, bearing in mind that detecting real alerts out of repetitive false ones is a complex and even an expensive task [3-4]. But if a minimal log set can be created, the

problems of repetition and real-time detection, even storage optimization can be solved.

Rough set is able to discover structural relationships in data sets because it has a mathematical approach to treat and process imperfect knowledge. Rough set has an algorithm that can identify the smallest set of data while keeping the original data set's knowledge [5-6], herein information security officers can do a better job by concentrating on real alerts and eliminating redundant ones and so taking right security actions in better timing base.

2.1 Rough Set Theory

Rough set theory is a novel mathematical technique for analyzing data; it deals with imperfect knowledge such as imprecision, vagueness, and uncertainty. This is very important since a frequent feature across the various domains of decision analysis, machine learning, pattern recognition, and data mining is processing incomplete and imprecise knowledge [23].

Rough set was proposed by Z. Pawlak back in 1980, it offers efficient algorithms, methods, and tools for recognizing hidden patterns in data [25]. Generally, rough set approach can solve the following issues:

- A group of objects is described depending on an attribute or feature values.
- Full or partial dependencies between features or attributes.
- Attributes or features reduction.
- Attributes or features significance.
- Generation of decision rules.
- plus others [26] [27].

Decision support systems, pattern recognition, knowledge discovery, expert systems, machine learning, decision analysis, and other real-life systems have all used rough set techniques. [28] [29].

Philosophy of rough set assumes that objects are associated with some data or details. Hence, according to the similarity in data, objects can be classified into similar or in-discernible groups or classes. This indiscernibility relation provides the mathematical foundation for rough set theory. Previously mentioned indiscernible classes form what is called knowledge granule, all knowledge granule union is called crisp set, or else the set is rough [27].

As a result, a rough set has boundary elements or elements that cannot be identified as members of the set or even as the complement of the set with certainty. Apart from that, crisp set is the opposite, where no boundary elements exist. Hence, a rough set can be used to mathematically model ambiguous concepts. A pair of notions known as the lower and upper approximations are required to express such ambiguous conceptions. The lower approximation includes all items that are certain to belong to the indiscernible class, while the upper approximation includes all objects that may or may not belong to the in-discernible class. [30] [31].

Figure 1 explains the previous concepts, an approximation for the given set of objects X can be computed depending on a coefficient named accuracy of approximation $\alpha(X)$ [32], which is calculated by dividing lower approximation by upper approximation. As mentioned before, lower approximation includes elements that surely belong to the considered set X denoted by $L(X)$, on the other hand, upper approximation

includes elements that maybe belong to X denoted by U(X) [33], clearly, $0 \leq \alpha(X) \leq 1$, considering the value $\alpha(X) = 1$ then the set X is called crisp, which means it is the normal set theory, but considering $\alpha(X) < 1$, then the set X is rough [34]. Thus, the concept of indiscernibility appears; once again objects are being indiscernible from each other if they were categorized within the same class referring to their related information [35].

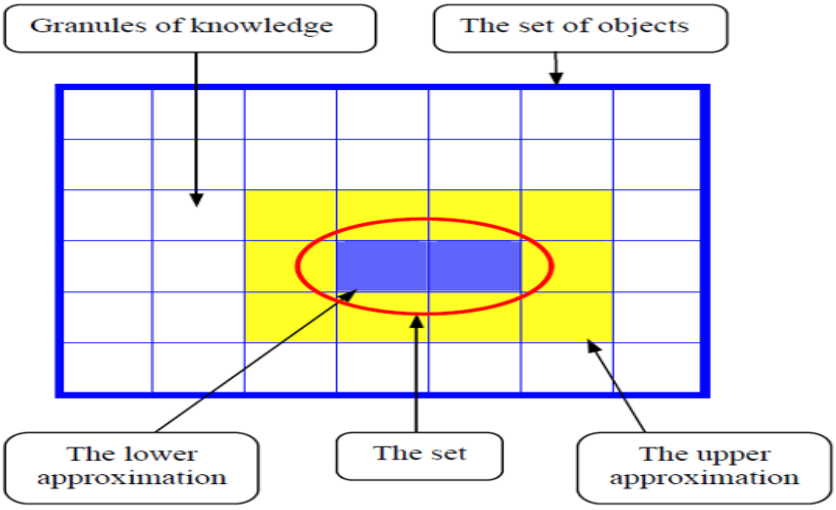

Figure 2.1: Set approximation.

2.2 R Language

R is a computer language that is designed for data analysis and data visualization. R was created by Robert Gentleman and Ross Ihaka at the University of Auckland, the purpose was to teach student statistics, but later in 1997, the source code has been freely dis-tributed. R has the benefit of having continuous enhancements and new source code being downloaded in the form of packages [18].

R also works across different operating systems like MS-Windows, Linux, and Unix [19]. In general, R language contains more than 5,000 algorithms in its library to analyze data [20], but in particular, R has 96 add-on packages related to the field of machine learning, their ideas and methods were implemented and developed mixing computer science and statistics. One of these packages is RoughSets package that contains algorithms developed according to rough set and fuzzy rough set theories [21]. Moreover, in addition to R tools, built-in data sets exist to simplify the process of machine learning and data analysis [22].

Z. Abbas and A. Burney [23] provided a survey of frequently used data processing tools that automate the application of rough set theory, these tools were R, Rose2, Rosetta, RSES, and WEKA. In the comparison, the following items were used:

- Package techniques.
- The language in which the package was created.
- Supported operating systems.
- Existence of user interface.
- Existence of rough set basic concepts: lower, upper approximation, boundary sets, etc.
- Existence of feature/instance selection.
- Ability to separate data to a training set and testing set.
- Ability to generate decision rules induced from reducts.
- Existence of nearest-neighbor-based algorithms to classify data.

Table 2.1 : Rough set packages functionalities comparison.

Components	**Rough Sets**	**Rose2**	**Rosetta**	**RSES**	**WEKA**
Technique	RST FRST	RST	RST	RST	RST
Programming Language	R	C++	C++	Java/ C++	Java
Operating System	Win./ Linux/ Mac	Win.	Win	Win./ Linux	Win./ Linux/ Mac
User Interface	Script	GUI	GUI	GUI	GUI
Basic Concepts	Yes	Yes	No	No	No
Discretization	Yes	Yes	Yes	Yes	Yes
Feature Selection	Yes	Yes	No	Yes	Yes
Instance Selection	Yes	No	Yes	No	Yes
Missing Value Completion	No	Yes	Yes	Yes	Yes
Decomposition	No	No	No	No	No
Rule Based Classifiers	Yes	Yes	Yes	Yes	Yes
Nearest Neighbour Based Classifiers	Yes	No	Yes	No	Yes
Cross Validation	No	Yes	Yes	Yes	Yes

Hence, according to the functionalities mentioned in Table 1, R was chosen as a tool for analyzing data in this research, bearing in mind that according to Craner r.project [24], the missing value completion functionality mentioned in Table 1 above is now part of RoughSet package in R.

Chapter 3

LITTRETURE REVIEW

The motivation for carrying out the work in this thesis was triggered after reading works in paper [7], this paper depending on 34 relevant publications selected from 1374 papers published within the past decade relevant to machine learning, cloud computing, clustering, web systems, and Internet of Things (IoT) discuses currently available trends in log analysis, their advancements and future tendencies. As mentioned in the work, very few of these studies relate both the process of analysis with security aspects, moreover, up-to-date studies are needed.

Algorithms of machine learning are being used in log analysis, these algorithms are slow and resource-intensive [7], which in turn poses a limitation in using such strategy. Reducing the size of targeted security log files by applying the proposed rough set approach will reduce the effect of limitations previously mentioned.

In general, paper [7] identifies current problems and limitations related to log analysis, these problems have been discussed but are in need of further research. Such issues were real-time analysis, speed and security of the algorithms, and multi-source analysis [8–10], in all cases mentioned log size will influence the treatment process through log analysis in terms of speed, accuracy, and resource consumption, so the work in this paper presents a solution for such problems by reducing log size but preserving core knowledge as explained below:

Papers [11–13] concentrated on moving intrusion detection to real-time approach so that system administrators can pinpoint the attacks and restore the normal system conditions on spot. But managing huge data logs is a bottleneck here in terms of accelerating the whole process. Using the proposed rough set concepts to process such logs will cause re-duction in size without losing the core knowledge, and as a consequence will speed up real-time intrusion detection, hence the work in this research can be used as a previous step to prepare log data.

The work in [9] proposed a future enhancement to their HERCULE intrusion detection system by using log files distributed across multiple hosts, but the huge growth of the number and size of log files was an obstacle, which can be solved if log files were processed using rough set as explained in this research.

The study performed by Y. Yao, et al [3] was similar in general terms of using the concepts of rough set theory, but it had a different purpose as well as different methodology. To identify security semantics, the researchers employed rough set theory to analyze alert data collected from multi-sources. This was performed through collecting security data from several resources, then by applying rough set theory concepts, a weight was being calculated for classifications of alerts, then alert aggregation was performed to eliminate repetitive and false alerts, and finally, a reliability metric was introduced according to background information to measure the credibility. Therefore, our research would be a complement to this work as an extra stage to enhance both the classification of alerts and credibility measuring.

M.R. Gauthama Raman et al. [14] presented a selection method to determine the optimal attribute subset of Intrusion Detection System (IDS). This technique used rough set theory and some properties of the hypergraph to enhance the accuracy of classification and time complexity of IDS. Dutta, S., Ghatak, S., Dey, R. et al. [15] proposed an attribute selection methodology that improves spam classification for Online Social Network (OSN). Rough set theory concepts were applied to develop an attribute selection algorithm to identify a smaller group of features that leads to improve classification performance.

Anitha, A., and D. P. Acharjya [16] proposed a feature selection technique based on a novel filter stands on Rough Set Theory approach and Hyper-Clique based Binary Whale Optimization Algorithm (RST-HCBWoB), the technique identifies informative features. This is necessary for an effective feature selection algorithm used in supervisory control and data acquisition intrusion detection system to protect critical infrastructure from cyber attacks.

The work in Nanda, N.B., Parikh, A. [17] proposed a hybrid technique that works on identified risks of the network-attached intrusion detection system in attempt to determine the minimum rules set that could represent the knowledge offered by the data set under consideration. Two models are used in this procedure: random forest classifier to select attributes, and rough set theory to generate rules.

Hence, regarding the work in papers [14–17] mentioned above, our study proposes more relevant research, it provides an algorithm by using rough set package in R language to find the optimal minimal subset of attributes rather than a smaller one, and the technique is general where it could be used for any type of system logs.

Chapter 4

REPRESENTATION OF SOLUTION

This chapter presents an overview of the datasets used throughout this dissertation. The proposed models were developed and tested using four publicly available English Twitter datasets retrieved from different repositories as shown in Table 4.1. The datasets were selected with different sizes.

In this section, we will present our proposed iterated rough set based algorithm, which we name IRS. IRS is proposed to be scalable for big data pre-processing for feature selection. The algorithm generates a minimal security log from any given big data set. The proposed steps are employed to accelerate the run time. Then, as a result of generating a minimal log, a minimal decision rules database is generated; this decision set maintains the data consistency embedded in the original dataset. In this section, we will also clarify our IRS algorithm as an efficient solution able to perform big data feature selection with less execution time. It will be compared with an existing novel algorithm using three benchmark datasets to prove its effectiveness. However, we will first explain the motivation for proposing IRS by discussing the computational complexity of the traditional rough set theory when working with high dimensional datasets.

4.1 Problem Statement and Motivation

Performing feature selection when using RST will force the theory to compute each possible attribute combination. The number of attribute subsets that maybe created

using *m* attributes from a set of *N* attributes is $\binom{N}{m} = \frac{N!}{m!(N-m)!}$ [41]. Hence, the number of generated feature subsets as a total, is $\sum_{i=1}^{N}\binom{N}{i} = 2^N - 1$. For instance, if N = 30, there will be around one billion possible combinations. This prevents the use of RST with high dimensional datasets. Moreover, hardware limitations exist, and in particular, memory capacity will not be able to store and calculate a huge number of entities. The RAM will need to allocate the entire dataset, its computations, and results. For big data this can exceed the physical memory. Our proposed algorithm was motivated by all of these reasons.

4.2 Datasets

In our research, we used four datasets, three of which were benchmark datasets taken from UCI [42]. The purpose was to examine the proposed algorithm's effectiveness. This will be discussed in more detail in Section 4.5 The fourth dataset was used to execute the proposed algorithm on real-life huge datasets.

Our real-life huge datasets were collected from a government enterprise that uses IBM security Qradar. Qradar is a Security Information and Event Management (SIEM) solution that collects and analyzes log data from security systems [43]. Three datasets were taken from Qradar, each containing 63,000 objects (Instances) with 10 attributes of unprocessed security events. This enterprise considers the cloud technology in its structure and virtual machine concepts for more than 40 servers that have both Microsoft and Linux operating systems. Part of the servers provide about 100 online services for citizens.

Table 4.1 shows the general structure of each SIEM dataset. Every dataset has 10 attrib-utes, A = {Event Name, Log Source, Event Count, Low-Level Category, Source

IP, Source Port, Destination IP, Destination Port, User Name, Magnitude}. The first 9 attributes are the condition attributes (C), while the last one, Magnitude, is the decision attribute {D}. Magnitude indicates the importance of the offense and has an integer value ranging from one to eight, being from least severe to most severe. Hence, each dataset forms a decision table, T = (U, A ∪ D). The data populated in table T contain no real-valued attributes, meaning that concepts of RST can be applied directly, without the need to perform extra pre-processing steps such as discretization [44]

Table 4.1: Decision table

Event Name	Log Source	Event Count	Low Level Category	Source IP	Source Port	Destination IP	Destination Port	User Name	Magnitude
Tear down UDP connection	ASA @ 172. 17.0 .1	1	Fire wall Session Closed	8.8. 8.8	53	172. 18. 12. 10	53,657	N/A	7
Deny protocol src	R	1	Fire wall Deny	172. 20. 12. 142	56,511	172. 217. 23. 174	443	N/A	8
Deny protocol src	ASA @ 172. 17.0 .1	1	Fire wall Deny	172. 20. 18. 54	52,976	213. 139. 38. 18	80	N/A	8
Deny protocol src	ASA @ 172. 17.0 .1	1	Fire wall Deny	172. 20. 15. 71	53,722	52. 114. 75. 79	443	N/A	8
Deny protocol src	ASA @ 172. 17.0 .1	1	Fire wall Deny	192. 168. 180. 131	55,091	40. 90. 22. 184	443	N/A	8
Built TCP connection	ASA @ 172. 17.0 .1	1	Fire wall Deny	172. 18. 12. 19	59,201	163. 172. 21. 225	443	N/A	8

4.3 Building a Minimal Log Size (Reduct)

Considering [46,45], both papers discussed the concept of using maximal or minimal pairs in discernibility matrix to overcome the complexity of feature selection. We will use this concept in our methodology inside iterations calculation for the same reason. Later, the results of [45] will be used to prove the efficiency of our algorithm.

To compute any minimal subset, two mathematical foundations are needed: discernibility matrix, and reduct. It was noted in Section 4.1 that this process is computationally expensive. The proposed IRS algorithm aims to overcome this limitation and reduce the execution time of minimal log generation by redesigning the calculations using two concepts: iteration calculations and minimal elements in the discernibility matrix calculations.

The iteration step divides the big dataset into N subsets and calculates the iterated minimal reduct for each, where finally the intersection of all previously calculated iterated minimal reducts will generate the core minimal feature subset. The second step focuses on reducing the calculation complexity in each iteration by passing only the minimal element in a discernibility matrix to reduce calculations. Working in such design will contribute to solving the problem in the following way:

- Splitting the dataset into N subsets and performing the proposed algorithm on each subset will overcome hardware limitations, since fewer entries means less memory space to upload the data, perform computations, and store the results. Keeping the whole high dimensional dataset in memory and performing all the previous steps, is mostly impossible.

Reducing the number of calculations, since passing only the minimal elements in the discernability matrix to reducts calculation will not cause the computation of each possible attribute combination, and hence the equation $\sum_{i=1}^{N}\binom{N}{i} = 2^N - 1$ is no longer valid. This will certainly reduce the execution time. The proposed code is given in Table 4.2.

Reducing the number of calculations, since passing only the minimal elements in the discernability matrix to reducts calculation will not cause the computation of each possible attribute combination, and hence the equation $\sum_{i=1}^{N}\binom{N}{i} = 2^N - 1$ is no longer valid. This will certainly reduce the execution time. The proposed code is given in Table 4.2.

Table 4.2: IRS Algorithm

- Input: T = (U,A∪D): information table, N: number of iterations,
- M: number of datasets
- Output: Core–Reduct,
- 1: For each dataset M do
- 2: For each iteration N do
- 3: Calculate $IND_N(D)$
- 4: Compute $DISC.Matrix_N(T)$
- 5: Do while ($DISC.Matrix_N(T) \neq \emptyset$) and $i \leq j$
- (RST discernibility matrix is symetric)
- 6: $S_{i0,j0}$ = Sort $(x_i,x_j) \in DISC.Matrix_N(T)$
- according to number of conditional attributes A
- 7: End while
- 8: Compute $Reduct_N(S_{i0,j0})$
- (calculating reducts for minimal condition atrridutes)
- 9: $Reduct_N = Reduct_N \cap Reduct_N(S_{i0,j0})$
- 10: End For N
- 11: Core–Reduct = Core–Reduct ∩ $Reduct_N$
- minimal optimal reduct
- 12: End For M

Table 4.3 shows the output after performing the algorithm using our datasets. The three datasets are labelled as S1, S2, and S3, respectively. It was found that the server cannot

run the whole data set with 6300 objects and 10 attributes at once, so each dataset was split into three parts and processed on three iterations (M = 3, N = 3). The table also shows the calculated degree of dependency for each iterated reduct, being how much the generated iterated reduct (attributes set) depends on the decision attribute(s), with a maximum value of 1. The methodology was performed under hardware specifications of Intel(R) Xeon(R) Gold 6148 CPU @2.40 GHz 2.39 GHz, RAM 48.3 GB.

The intersection of the three iterations of the first data set S1 produced a minimal iterated reduct of 5 attributes $|Reduct_{N} = 1| = 5$, while the original set S1 had 10 attributes.

Table 4.3: Minimal reduct output for N = 3, M = 3

Training Data Set	**Minimal Attribute**	**Degree of Dependency [1]**
First Training Set S1 (∩ three iterations) $Reduct_{N = 1}$	A1 = {Event Name, Source IP, Source Port, Destination IP, Magnitude } |A1| = 5	1
Second Training Set S2 (∩ three iterations) $Reduct_{N = 2}$	A2 = { Event Name, Source IP, Destination IP, Magnitude }|A2| = 4	0.9992941
Third Training Set S3 (∩ three iterations) $Reduct_{N = 3}$	A3 = {Event Name, Source IP, Source Port, Destination IP, Magnitude } |A3| = 5	1
Core-Reduct (A1∩ A2∩ A3)	A2 = { Event Name, Source IP, Destination IP, Magnitude }|A2| = 4	0.9992941

[1]: a decision attribute, d, totally depends on a set of attributes A, written as $A \Rightarrow d$ if all attribute values from d are distinctly identified by attribute values from A

In this reduct, the degree of the dependency = 1, which means the decision attribute {Magnitude} was completely identified by the values of the 5 attributes of the reduct

set A1. This reduct omitted 50% of the attributes of the original set and retained information content 100%. For the second iteration S2, the reduct was even better, producing 4 attributes, $|Reduct_N = 2| = 4$, with a dependency degree of 0.9992941, while the last iteration for S3 had the same output as S1.

Following step 11 in the algorithm, Core-Reduct was generated by taking the intersection of all previous iterations' outputs. This means that Core-Reduct = {Event Name, Source IP, Destination IP, Magnitude} was the minimal reduct for all datasets S1, S2, and S3. It had 4 attributes, rather than the 10 attributes of the original sets. Despite this reduct, the information content of the original datasets was retained with 99.9% accuracy.

This proves that the proposed solution was able to create a minimal reduct for the security log, this optimal reduct used only 40% of the attributes of the original dataset (4 instead of 10), and still offered the same information covered by the original dataset with 99.9% accuracy degree. The next section will use this minimal dataset to create a minimal decision rules database. The effectiveness of step 8, which passes only the minimal ele-ments of discernibility matrix for reduct calculations inside each iteration, will be proved in Section 4.5, by comparing our algorithm with a similar approach using the same benchmark datasets in terms of runtime.

4.4 Generating Minimal Decision Rules

It is significant to understand that the derivation of rule structure, using learning procedures from training cases, is being employed in rule-based expert systems. Fortunately, these rules are more accurate than information included in the original input

data set, because new examples that do not match examples taken from the original data, are being properly classified by such rules [23].

Te RoughSets package in the R language has different algorithms to extract knowledge hidden in any given data set in the form of an IF...THEN structure. This paper uses the CN2Rules algorithm, which is designed to work even with the existence of imperfect data. The CN2Rules algorithm was deployed on each of the reduct sets A1, A2, and A3, produced in the previous section. The algorithm in Table 4.4 generates the decision rules in the form of an IF...THEN structure, with each set divided into a training set with 60%, and a test set with 40%, as shown in step 3, because this will be used later in steps 5 and 6 to validate the accuracy of the prediction using the 40% test part.

Table 4.4: Rule generation algorithm

Input: $Reduct_N$ (T): minimal reduct information table, M: number of datasets
Output: $Set\text{-}Rule_{Min}$
1: For each dataset M do
2: read.table($Reduct_N$ (T))
3: Splitting $Reduct_N$ (T)
training set 60% and a test set 40%.
4: RI.LEM2Rules.RST() function
Create rules depending on training set of $Reduct_N$ (T)
5: predict() function
Testing the quality of prediction depending on the test set of $Reduct_N$ (T)
6: mean() function.
Checking the accuracy of predictions
7: End For M

Table 4.5 shows the total number of rules generated for each dataset before minimizing (S1, S2, S3), and after minimizing (A1, A2, A3). It also calculates the prediction accuracy of each minimal iterated reduct set (A1, A2, A3).

Table 4.5: Minimal reduct output for N = 3, M = 3.

Training Data Set	Number of Decision Rules before Reduct	Number of Deccision Rules after Reduct	Prediction Accuracy
First Training Set	S1 = 905	A1 = 596	0.9552733
Second Training Set	S2 = 878	A2 = 509	0.9535073
Third Training Set	S3 = 813	A3 = 481	0.9741291

Examining the first row in the table and comparing the number of rules generated from the first original dataset S1 and its minimal reduct A1, the number of rules decreased to about 66% with a prediction accuracy of around 96%. A similar result occurred for datasets S1 and S2.

A minimal decision rules dataset was successfully created for each original dataset (S1, S2, S3). Each minimal decision rules dataset (A1, A2, A3) reduced the number of the rules (by 50% to 65%) with high accuracy prediction (from 95% to 97%). In addition, we know from the previous section that each minimal iterated set (A1, A2, A3) strongly represents the knowledge in its original dataset (S1, S2, S3) with a high degree of dependency (ranging from 1 to 0.99). We conclude that the same knowledge is being presented in the form of decision rules, with a smaller number of attributes and high accuracy prediction.

4.5 Execution Time Comparison with Existing Methods

The current section evaluates the efficiency of our IRS algorithm. Our technique for generating a minimal subset will be compared with two other techniques: one using classical discernibility matrix [47], while the other uses its own proposed novel algorithm, named Sample pair selection SPS [45]. The experiments used the same hardware environment specifications mentioned in [45], being Intet (R) i5 CPU 2.40 GHz M450.

The comparison will measure the runtime needed to calculate the minimal reduct, using the three algorithms on the same datasets. It is worth noting that the comparison uses three benchmark datasets taken from the UCI machine learning repository [44]. Table 4.6 shows the description of the original dataset. These three datasets were previously used in [45,47] to compare the effectiveness of the SPS algorithm against the classical discernibility matrix.

Table 4.6: Original Datasets description.

Dataset	**Number of Attributes**	**Number of Instances**
Glass	9	100
Wiscon	9	699
Zoo	16	100

We executed our algorithm IRS using the three benchmark datasets and compared the runtime values with the previous statistical calculations from [45,47]. As shown in Table 4.7, our algorithm IRS generated the same number of all possible reducts for glass and Wiscon datasets 2 and 4 respectively. However, in the case of the zoo dataset, our algorithm created 35 reducts, while both compared algorithms created 33, yet our algorithm had the best runtime over the other two algorithms, at 0.9967 s. A general comparison of the runtime of all algorithms shows that our IRS algorithm had the best execution time over SPS and classical discernibility matrix, for all datasets. This proves that the IRS algorithm, which uses iteration calculations depending on minimal elements of discernibility matrix, decreased the complexity of calculations successfully.

Table 4.7: Computations of execution time in finding minimal reduct

Data	**Num. of Attributes of The Dataset**	**All reducts**		**Execution Time in seconds**		
		IRS	SPS & CDM	Classical DiscernibilityMatrix (CDM)	SPS	IRS
Wiscon	9	4	4	1362.1	24.0956	9.05
Glass	9	2	2	23.3268	0.7931	0.7
Zoo	16	35	35	106.6581	1.2574	0.9967

Chapter 5

CONCLUSION AND FUTUR WORKS

Following the proposed procedure, this research designed a new algorithm named IRS to create a minimal security log. The approach used RST basic concepts by adopting an iterated model. Inside each iteration, minimal discernability matrix elements were passed for reduct calculations. This design helped to overcome hardware limitations and prevent reduct calculation growing exponentially high, by decreasing the calculation needed to compute all possible attribute combinations to the minimal elements in a discernibility matrix.

We also computed a minimal decision rule database with a prediction accuracy of about 96%. This minimal subset used only 40% of the attributes of the original feature set, with a 99.9% degree of dependency (knowledge consistency).

We compared our methodology with another recent novel algorithm, using the same three benchmark datasets. Our comparison showed that the proposed methodology effectively calculated a minimal set without losing performance. The results showed that our methodology was even better in terms of execution time, which proved that calculation complexity, as well as search space, were reduced. This makes the proposed model relevant to huge datasets and will enhance real-time analyses.

In the future, we will apply the new concept of Sensitivity Analysis (SA) to this work, because SA can manage uncertainty in a real-world decision system, especially in

high-dimensional problems. This will surely offer a better solution for work in this field of research.

REFERENCES

[1] J.Kreps, IHeartLogs: EventData, StreamProcessing, and DataIntegration.O'ReillyMedia, 2014. [Online] Available: https://books.google.jo/books?id=gdiYBAAAQBAJ.

[2] M.Collins, Network Security Through Data Analysis: Building Situational Awarness. O'Reilly Media, 2014.[Online]. available: https://books.google.jo/books?id=pZvQAgAAQBAJ

[3] A. Buecker, S. Arunkumar, B. Blackshaw, M. Borrett, P.Brittenham, J.Flegr, J.Jacobs,V.Jeremic, M.Johnston, C.Marketal., UsingtheIBM Security Framework and IBM Security Blue print to Realize Business-Driven Security, ser. IBMredbooks. IBM Redbooks, 2014. [Online]. Available:https://books.google.jo/books?id=K3bJAgAAQBAJ

[4] Chuvakin, K. Schmidt, and C. Phillips, Logging and Log Management: The Authoritative Guide to Under-standing the Concepts Surrounding Logging and LogManagement. Elsevier Science, 2012. [Online]. Available:https://books.google.jo/books?id=Rf8MXYTUoC

[5] S. Yen and M. Moh, "Intelligent log analysis using machine and deep learning," in Research Anthology on Artificial Intelligence Applications in Security. IGI Global, 2021, pp. 1154–1182

[6] B. Lundgren and N. Moller, “Defining information security.” Science and Engineering Ethics, vol. 25, no. 2, pp. 419–441, 2019.

[7] Y. Yao, Z. Wang, C. Gan, Q. Kang, X. Liu, Y. Xia, and L. Zhang, “Multi-source alert data understanding for security semantic discovery based on rough set theory,” Neurocomputing, vol. 208, pp. 39–45, 2016.

[8] L. Bao, Q. Li, P. Lu, J. Lu, T. Ruan, and K. Zhang, “Execution anomaly detection in large-scale systems through console log analysis,” Journal of Systems and Software, vol. 143, pp. 172–186, 2018.

[9] S. Zbigniew, “An introduction to rough set theory and its applications-a tutorial,” Proceedings of ICENCO’2004, 2004.

[10] K. S. Ray, Soft Computing and Its Applications, Volume One: A Unified Engineering Concept. CRC Press, 2014, vol. 1.

[11] J. Svacina, J. Raffety, C. Woodahl, B. Stone, T. Cerny, M. Bures, D. Shin, K. Frajtak, and P. Tisnovsky, “On vulnerability and security log analysis: A systematic literature review on recent trends,” in Proceedings of the International Conference on Research in Adaptive and Convergent Systems, 2020, pp. 175–180.

[12] A. Ambre and N. Shekokar, “Insider threat detection using log analysis and event correlation,” Procedia Computer Science, vol. 45, pp. 436–445, 2015.

[13] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu, "Hercule: Attack story reconstruction via community discovery on correlated log graph," in Proceedings of the 32Nd Annual Conference on Computer Security Applications, 2016, pp. 583–595.

[14] D. Zou, H. Qin, H. Jin, W. Qiang, Z. Han, and X. Chen, "Improving log-based fault diagnosis by log classification," in IFIP International Conference on Network and Parallel Computing. Springer, 2014, pp. 446–458.

[15] M. Almgren and E. Jonsson, "Using active learning in intrusion detection," in Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004. IEEE, 2004, pp. 88–98.

[16] A. Goel, W.-c. Feng, W.-c. Feng, and D. Maier, "Automatic high-performance reconstruction and recovery," Computer Networks, vol. 51, no. 5, pp. 1361–1377, 2007.

[17] M. G. Kang, S. McCamant, P. Poosankam, and D. Song, "Dta++: dynamic taint analysis with targeted control-flow propagation." in NDSS, 2011.

[18] M. G. Raman, K. Kirthivasan, and V. S. Sriram, "Development of rough set–hypergraph technique for key feature identification in intrusion detection systems," Computers & Electrical Engineering, vol. 59, pp. 189–200, 2017.

[19] S. Dutta, S. Ghatak, R. Dey, A. K. Das, and S. Ghosh, "Attribute selection for improving spam classification in online social networks: a rough set theory-

based approach," Social Network Analysis and Mining, vol. 8, no. 1, pp. 1–16, 2018.

[20] A. Anitha and D. Acharjya, "Crop suitability prediction in vellore district using rough set on fuzzy approximation space and neural network," Neural Computing and Applications, vol. 30, no. 12, pp. 3633–3650, 2018.

[21] N. B. Nanda and A. Parikh, "Hybrid approach for network intrusion detection system using random forest classifier and rough set theory for rules generation," in International Conference on Advanced Informatics for Computing Research. Springer, 2019, pp. 274–287.

[22] S. Tuffery, ´ Data mining and statistics for decision making. John Wiley & Sons, 2011.

[23] P. J. Aphalo, Learn R: As a Language. CRC Press, 2020.

[24] B. Makhabel, Learning data mining with R. Packt Publishing Ltd, 2015.

[25] T. Hothorn, "Cran task view: Machine learning & statistical learning," 2021.

[26] H. I. Rhys, Machine Learning with R, the tidyverse, and mlr. Manning Publications, 2020.

[27] Z. Abbas and A. Burney, “A survey of software packages used for rough set analysis,” Journal of Computer and Communications, vol. 4, no. 9, pp. 10–18, 2016.

[28] S. S. Pal and S. Kar, “Time series forecasting for stock market prediction through data discretization by fuzzistics and rule generation by rough set theory,” Mathematics and Computers in Simulation, vol. 162, pp. 18–30, 2019.

[29] Z. Pawlak, J. Grzymala-Busse, R. Slowinski, and W. Ziarko, “Rough sets,” Communications of the ACM, vol. 38, no. 11, pp. 88–95, 1995.

[30] T. Y. Lin and N. Cercone, Rough Sets and Data Mining. Springer US, 1997.

[31] Z. Suraj, “An introduction to rough set theory and its applications,” ICENCO, Cairo, Egypt, vol. 3, p. 80, 2004.

[32] W. P. Ziarko, Rough Sets, Fuzzy Sets and Knowledge Discovery: Proceedings of the International Workshop on Rough Sets and Knowledge Discovery (RSKD’93), Banff, Alberta, Canada, 12–15 October 1993. Springer Science & Business Media, 2012.

[33] T. Lin and A. Wildberger, “The third international workshop on rough sets and soft computing proceedings (rssc’94),” San Jose State University, San Jose, California, USA, 1994.

[34] A. Skowron and Z. Suraj, Rough Sets and Intelligent Systems-Professor Zdzisław Pawlak in Memoriam: Volume 2. Springer Science & Business Media, 2012, vol. 43.

[35] Q. Yang, P.-a. Du, Y. Wang, and B. Liang, "A rough set approach for determining weights of decision makers in group decision making," PloS One, vol. 12, no. 2, p. e0172679, 2017.

[36] S. Narli, N. Yorek, M. Sahin, and M. Usak, "Can we make definite categorization of student attitudes? a rough set approach to investigate students' implicit attitudinal typologies toward living things," Journal of Science Education and Technology, vol. 19, no. 5, pp. 456–469, 2010.

[37] H.-C. Ho, W. Jann-Der Fann, H.-J. Chiang, P.-T. Nguyen, D.-H. Pham, P.-H. Nguyen, and M. Nagai, "Application of rough set, gsm and msm to analyze learning outcome—an example of introduction to education," Journal of Intelligent Learning Systems and Applications, vol. 8, no. 1, pp. 23–38, 2015.

[38] Z. Pawlak and A. Skowron, "Rough sets and conflict analysis," in E-Service Intelligence. Springer, 2007, pp. 35–74.

[39] V. Del Giudice, P. De Paola, and G. B. Cantisani, "Rough set theory for real estate appraisals: An application to directional district of naples," Buildings, vol. 7, no. 1, p. 12, 2017.

[40] A. Majeed, R. ur Rasool, F. Ahmad, M. Alam, and N. Javaid, “Near-miss situation based visual analysis of siem rules for real time network security monitoring,” Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 4, pp. 1509–1526, 2019.

[41] Razavi, S.; Jakeman, A.; Saltelli, A.; Prieur, C.; Iooss, B.; Borgonovo, E.; Plischke, E.; Piano, S.L.; Iwanaga, T.; Becker, W.; et al. The future of sensitivity analysis: An essential discipline for systems modeling and policy support. Environ. Model. Softw. 2021, 137, 104954.

[42] Beaubouef, T.; Petry, F.; Arora, G. Information-theoretic measures of uncertainty for rough sets and rough relational databases. Inf. Sci. 1998, 109, 185–195.

[43] Tang, J.; Wang, J.; Wu, C.; Ou, G. On uncertainty measure issues in rough set theory. IEEE Access 2020, 8, 91089–91102.

[44] Parthaláin, N.M.; Shen, Q.; Jensen, R. A distance measure approach to exploring the rough set boundary region for attribute reduction. IEEE Trans. Knowl. Data Eng. 2010, 22, 305–317.

[45] Dubois, D.; Prade, H. Rough fuzzy sets and fuzzy rough sets. Int. J. Gen. Syst. 1990, 17, 191–209.

[46] Tuffery, S. Data Mining and Statistics for Decision Making; John Wiley & Sons: Hoboken, NJ, USA, 2011.

[47] Rodriguez, J.D.; Perez, A.; Lozano, J.A. Sensitivity analysis of k-fold cross validation in prediction error estimation. IEEE Trans. Pattern Anal. Mach. Intell. 2009, 32, 569–575.