STRENGTHENING INTRUSION DETECTION SYSTEM FOR ADVERSARIAL ATTACKS: IMPROVED HANDLING OF IMBALANCE CLASSIFICATION PROBLEM

CHUTIPON PIMSARN

MASTER OF SCIENCE
IN
INFORMATION TECHNOLOGY

SCHOOL OF INFORMATION TECHNOLOGY
MAE FAH LUANG UNIVERSITY

2021

©COPYRIGHT BY MAE FAH LUANG UNIVERSITY

STRENGTHENING INTRUSION DETECTION SYSTEM FOR ADVERSARIAL ATTACKS: IMPROVED HANDLING OF IMBALANCE CLASSIFICATION PROBLEM

CHUTIPON PIMSARN

THIS THESIS IS A PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
INFORMATION TECHNOLOGY

SCHOOL OF INFORMATION TECHNOLOGY
MAE FAH LUANG UNIVERSITY
2021
©COPYRIGHT BY MAE FAH LUANG UNIVERSITY

STRENGTHENING INTRUSION DETECTION SYSTEM FOR ADVERSARIAL ATTACKS: IMPROVED HANDLING OF IMBALANCE CLASSIFICATION PROBLEM

CHUTIPON PIMSARN

THIS THESIS HAS BEEN APPROVED
TO BE A PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE
IN
INFORMATION TECHNOLOGY
2021

EXAMINATION COMMITTEE

.....CHAIRPERSON
(Asst. Prof. Santichai Wicha, Ph. D.)

.....ADVISOR
(Assoc. Prof. Wg. Cdr. Tossapon Boongoen, Ph. D.)

.....CO-ADVISOR
(Assoc. Prof. Natthakan Iam-on, Ph. D.)

.....EXAMINER
(Sujitra Arwatchananukul, Ph. D.)

.....EXTERNAL EXAMINER
(Assoc. Prof. Wg. Cdr. Thiansiri Luangwilai, Ph. D.)

©COPYRIGHT BY MAE FAH LUANG UNIVERSITY

ACKNOWLEDGEMENT

This thesis was made possible by the kindness of many persons that I would like to thank. I would like to thank to my advisor and co-advisor, Assoc.Prof. Wg.Cdr. Dr. Tossapon Boongoen and Asst. Prof. Dr. Natthakan Iam-on, respectively. The study of this work is a part of the Newton Fund IAPP2017 project, which has been jointly funded by Thai Research Fund and Royal Academy of Engineering.

Finally, I would like to acknowledge Mae Fah Luang University for financial support in writing my thesis journal and publication grant. Furthermore, I would like to extend my gratitude to the various Mae Fah Luang University members, lecturers of the school of Information Technology, and the Postgraduate office.

Chutipon Pimsarn

Thesis Title	Strengthening Intrusion Detection System for Adversarial Attacks: Improved Handling of Imbalance Classification Problem
Author	Chutipon Pimsarn
Degree	Master of Science (Information Technology)
Advisor	Assoc. Prof. Wg. Cdr. Tossapon Boongoen, Ph. D.
Co-Advisor	Assoc. Prof. Natthakan Iam-on, Ph. D.

ABSTRACT

A network-based intrusion system or NIDS is the best defense mechanism, often sup-optimal for detecting an unseen malicious pattern. In response, many studies have attempted to promote NIDS that uses machine learning to improve their ability to recognize an opponent's attack. According to this research line, the current work focuses on non-payload connections at the TCP stack level, which is generalized and applicable to different network applications. As a complement to the recently published investigation that searches for the most informative feature space for classifying obfuscated connections, the problem of class imbalance is examined herein. Especially, a multiple-clustering-based undersampling framework is proposed to determine the set of cluster centroids that best represent the majority class to reduce par size to be equal to the minority. Initially, a pool of centroids is created using ensemble clustering to obtain a group of precise and diverse groupings. Then select the last representative from this group. Three different objective functions are formed for this optimization-driven process, thus leading to three variants of FF-Majority, FF-Minority, and FF-Overall. Based on a detailed assessment of the published dataset, four classification models, and different settings, these new methods show better predictive performance than baseline

data: the single-clustering undersampling counterpart and state-of-the-art techniques. Parameter analysis and implications for analyzing an extreme case are also guidelines for future applications.

Keywords: Intrusion detection system, Adversarial attack, Machine learning, Imbalance classification, Data clustering

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES (continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Background and Rational

As the world becomes more interconnected web by web, web-based applications like personalized online banking [1], industrial control systems [2], Internet of things [3], and wireless sensor networks [4] are subject to various security vulnerabilities, thus raising an urgent need for effective network and information security measures [5 - 6]. For this, some attempts have developed different defense mechanisms such as a hardware/software firewall and an intrusion detection system (IDS) to safeguard assets and system/device control in cyberspace [7 - 9]. However, using unpatched services is a typical instance of intrusive attacks that remain one of the crucial threats to both individuals and organizations [10]; this is highly critical to the Internet-based monitoring of engineering systems in general, especially the case in the case of critical national infrastructures such as health care., manufacturing, power grid, gas, and oil refineries [11]. A similar approach has recently been proposed to detect intrusive attacks on in-vehicle communication systems, particularly the controller area-network bus protocol [12].

To respond to a network-based intrusion detection system (NIDS) has been continuously invented and improved to provide security by monitoring network traffic and identifying malicious connections [13]. However, to support timely intervention and valuable information for human operators, a NIDS must be resilient to new breeds of threats like polymorphism, which allows an exploit code to avoid positive signature matching. Overall, initial signature-based systems are ineffective for detecting mutated patterns and zero-day attacks, with a common problem of a high false-positive rate [14 - 15]. This lack has led to a family of machine learning-based NIDS that have proven more efficient, providing examples of up-to-date connections and corresponding expert-directed labels

[16 - 17]. Over the years, different classification models were examined for this task, sometimes called an anomaly-based approach [18]. These include support vector machine or SVM [19], k-means clustering [20], artificial neural networks or ANN [21], and classifier ensemble [22]. Despite reports of success in the literature, the methods mentioned earlier can detect methods as mentioned earlier some unknown intrusions that partly resemble those known to the NIDS. However, they are still prone to adversarial attacks that exploit various new obfuscation techniques [23 - 24]. These attacks are branded as adversarial machine learning, exploring blind spots of a machine learning-based system [2]. In particular, slight perturbations are introduced to new connections such that the pre-trained model may incorrectly justify a decision boundary, thus a predicted class [25].

Without knowledge of obfuscation methods, developing NIDS that are more flexible in recognizing modified traffic patterns is required. One way to achieve this is continually updating the underlying classifier with new training instances. At the same time, another focuses on learning more from the attack classes that are commonly a minority in training data collection [26 - 27]. Specific to the latter, many studies have used techniques developed within the data science community to handle this class imbalance issue [18]. For example, conventional oversampling techniques, such as oversampling synthetic minorities or SMOTE [28] and its variants, have been a common choice for different classification algorithms to cope with this problem [19, 21 - 22]. However, the resulting model shows overfitting as it is not generalized to unseen data [29].

On the other hand, an undersampling strategy has proven more robust to overfitting across different domain problems [30 - 31]. Following the work of [32] that introduces an initial model of clustering-based undersampling, the multiple clustering-based approaches proposed in this paper aim to explore a pool of representative candidates richer than that obtained from one clustering. Hence, it may better preserve functional data patterns presented in the majority class and improve predictive performance [33]

Assumption Following the initial research [23], this paper focuses on classification-based NIDS, which does not analyze payload data [24]. It considers the object of the TCP connection, not at the lower level of the individual packet. It can be assumed that the adversary knows the victim's system design. However, it can only mutate victim inputs to this system, so they must conform with the TCP/IP protocol specification. Based on the report [34], this is achieved by designing non-payload-based obfuscation techniques to work at network and transport layers, and This makes the attack look similar legitimate attack. This approach is on par with evasions in the phase of IDS measurement, which is defined in the taxonomy of adversarial attacks against IDS [35].

Problem and Scope: From data collection generated within the study [23], the current work aims to develop an effective clustering-based undersampling method, which is novel and more accurate than the baseline [32]. Other well-known methods are found in the literature [36 - 37] and the standard oversampling counterpart [28]. This new method selects representative samples from multiple clusterings generated with various settings to promote diversity between them. It is in line with the concepts and success reported in the field of consensus clustering [38 - 39]. Especially this selection is designed as an optimization problem. Hence, it is the best alternative added to the target sampling set iteratively. First, it is applied to reduce the cardinality of instances belonging to the majority class of the binary classification problem, i.e., legitimate and attack connections are majority and minority classes. Following that, a classification model developed from the balanced training set is employed to categorize unseen and obfuscated attacks. Intuitively, the chance of recognizing these modified patterns may be improved, as the classifier can learn more from the smaller class, while information loss due to undersampling of the majority class is minimized.

Contributions: The main contributions of this paper are summarized as follows

1. This work extends the previous research of [23] to develop an accurate classification-based NIDS that is robust to adversarial attacks. It presents new multiple clusterings-based methods to handle imbalance classification through undersampling the majority class. The proposed framework allows any classification algorithm to learn better with the minority class representing attack connections. As such, the resulting model becomes more effective in recognizing obfuscated intrusions, whose

appearances partly overlap with those known direct attacks. In addition, the resulting framework is generalized to a broader range of non-payload adversarial attacks.

2. The offered undersampling technique is based on an iterative and greedy-optimization method of selecting the most acceptable alternative from a pool of centroids representing different clustering results or data partitions. This technique is inspired by ensemble or consensus clustering [38, 40], which usually provides more accurate data partition than a single clustering. Hence, this idea is novel and may improve the effectiveness of an initial work [35] that uses only one clustering to guide the sampling procedure. For the selection as mentioned earlier, it is designed as an optimization problem that maximizes three different objective functions: (i) the maximum distance from the centroid under the question to other k nearest centroids in the pool, (ii) the maximum distance from the examined centroid to k nearest instances from the minority class, and (iii) the average between functions (i) and (ii), respectively.

3. The proposed model is evaluated against the single clustering alternative of [35], RUS (Random UnderSampling [36]) as the conventional model of undersampling, and other well-known ensemble-level techniques: RUSBoost [36] and IRUS (Inverse Random UnderSampling [37]), the oversampling technique of [28] and its recent extension [41]. A range of basic classifiers and classifier ensemble methods are employed to generalize this experimental investigation. Specific to the research line of IDS, state-of-the-art models were also included as comparison methods. In addition, an analysis of parameters specific to the proposed techniques is included. Finally, to illustrate and discuss the relationship between predictive performance and algorithm variables. Anyone who tries to apply this to a future problem would find it helpful to tailor the model for good performance.

The next part of this paper is arranged as follows. Chapter 2 emphasizes related works and problem definition, with details of the dataset exploited herein. Besides, Chapter 3 presents the proposed undersampling framework, including the underlying optimization process. Then, the performance evaluation of new and compared methods is included and discussed in Chapter 4. In the end, the conclusion with directions for future research is given in Chapter 5.

1.2 Objective

1.2.1 To develop a framework that can solve the imbalance problem using the undersampling approach. That includes the application of multiple clusterings and the furthest first selection.

1.2.2 To develop a prototype of this framework that can be led to the development of the intrusion detection system or be shared with researchers in both data science and cyber security communities.

1.3 Scope

1.3.1 The dataset used in this study is obtained from the published research in 2018 Improving Network Intrusion Detection Classifiers by Non-payload-Based Exploit-Independent Obfuscations: An Adversarial Approach [30]. Focus on detecting new threats to keep up with all types of attacks. Samples are TCP/IP traffic records captured by a log management tool, and they can be categorized into three classes: non-attack (Legitimate), simple attack (Direct attack), and attack with hidden data (Obfuscated), respectively. This dataset is pointed out by the project partners, RTAF and T-NET, as a good representative of actual incidents and the following learning model. Besides, regarding data privacy and the future publication of findings, it is better to use this set than organization-specific data.

1.3.2 The problem is designed as data classification, where some basic classification algorithms will be employed. These include Support-vector machine, Naïve Bayes, Decision trees, and Logistic regression.

1.3.3 To solve the imbalance problem, this research focuses on the undersampling approach, which aims to reduce the number of samples belonging to a majority class. This decision is based on the oversampling counterpart that typically leads to noise generation and the overfitting problem. However, the concept of clustering is brought about to ensure the quality of samples drawn from the entire set. This may help ease information loss, a disadvantage of most undersampling methods.

1.3.4 To obtain an accurate and reliable conclusion, the evaluation framework of n-fold cross-validation is exploited using the standard accuracy metric that can be applied to the overall and class-specific outcomes.

1.3.5 The Python programming language implements all methods of prototypes used in this study.

1.4 Research Plan

This study follows a conventional data-mining development process, starting from problem definition to model development and evaluation. These working stages can be summarized as follows.

Stage 1 – Problem definition: Implementation at the beginning of the project as a collaboration with researchers from partner organizations, e.g., Northumbria University, UK Ministry of Defence, RTAF, and T-NET. Its purpose is to clarify research questions and the scope of the investigations.

Stage 2 – Literature review: the second stage is to find advancements for the previously defined problem. This focuses on the intersection between cybersecurity and data science domains, where this study belongs. It also provides a review of datasets available in the literature, which can be used for model development and comparison with reported benchmarks.

Stage 3 – Data collection and preparation: the target data is collected from the public portal and verified against the published properties. This is to ensure that the dataset is valid for future development. Standard preparation methods like the normalization process are applied to this data to be standardized and ready for the following stages.

Stage 4 – Development and evaluation of initial classification models: basic classification models are generated from the original features. These are assessed using n-fold cross-validation and the accuracy metric.

Stage 5 – Development and evaluation of the clustering-based undersampling method: k-means is used to generate representatives of samples belonging to a majority class. These will then be used with those samples of minority classes to form the target

dataset. Then, it will be evaluated with those classification algorithms identified in the previous stage.

Stage 6 – Development and evaluation of multiple clustering-based with furthest first selection: multiple clusterings of k-means are employed to form a pool of representatives, which will be iteratively selected using the intuition of furthest first. The resulting representatives will then be used with those samples of minority classes to form the target dataset. Then, it will be evaluated with those classification algorithms identified in Stage 4.

Stage 7 – Prototype implementation: All those methods are developed for primary model-based clustering, and the furthest-first selection methods are implemented in the Python programming language.

Stage 8 – prepare final publications and reports.

1.5 Project Plan

Given those stages within the proposed research plan, the following table summarizes the planning and timeline of this study.

Table 1.1 Project time schedule

Stage/ Semester	**1st Semester**	**2nd Semester**	**3rd Semester**	**4th Semester**	**5th Semester**	**6th Semester**	**8th Semester**
Stage 1							
Stage 2							
Stage 3							
Stage 4							
Stage 5							
Stage 6							
Stage 7							
Stage 8							

CHAPTER 2

LITERATURE REVIEW

This chapter presents theories, research, and tools regarding the Intrusion Detection System. Data mining techniques are also introduced, as we have mentioned, to explore a new approach to step up Intrusion Detection Systems due to sophisticated cyber attacks.

2.1 Institution Detection

The cyber attacks have step-up capabilities and are more complex to avoid being detected by network security systems and acquire access controls; the data integrity, as well as access control to the network, would be compromised by the incognito or third parties [42]. Therefore, the intrusion detection systems are designed as an approach to identify and foretell cyber-attack tricks to deal with illegitimate connections through collecting and analyzing network connection data, so intrusion detection systems can be categorized as follows.

Host-based IDS, known as HIDS, are intrusion detection systems capable of detecting and monitoring the internals of computer systems and network connections [43]. Moreover, host-based IDS can also inspect the access rights of network connections running in computer systems.

Network-based IDS, known as NIDS, are intrusion detection systems that are capable of detecting, auditing, and monitoring network connections at a high level like a router to find and identify suspicious network connections; such as an excessive number of internet connection packet requests to connect TCP protocol connections with various port numbers.

The circumstances imply that port scan attacks affect computer systems and other resources [44]. Network connections over NIDS can categorize into groups following connection interpretation, packet-level trace connection data, and IPFIX data with high-dimensional characteristics of network connections in numerical data forms, categorized data, and hybrid data.

Intrusion detection techniques can categorize into three groups: misuse-based, signature-based, anomaly-based, and hybrid, as mentioned below in Table 2.1. Currently, the researchers focus on anomaly-based detection techniques that are flexible to the constantly changing attack. It also covers the recorded infiltrations pattern that has come in the past. And a new approach along the way.

Table 2.1 Network intrusion detection techniques

Techniques	Characteristics
Misuse-based	1. The misuse-based technique detects network connections based on a signature pattern. 2. The signature-based scheme will become obsolete as cyberattacks become more sophisticated.
Anomaly-based	1. The anomaly-based techniques use the baseline of regular network activity to detect non-existent network activity in a network connection. 2. The anomaly-based techniques use the baseline of regular network activity to detect malicious network connections. However, when network activity is not fundamentally the same as the baseline of network activity will result in false positives. 3. The malicious network connections detected by anomaly-based techniques will be false positives. Therefore, it is up to humans to make the decision to reduce false positives.
Hybrid	The hybrid techniques are combinations of misuse and anomaly-based techniques.

As mentioned earlier, intrusion detection systems play an essential role in protecting network safety.

However, cyber-attacks have become more complex, fragmented, and frequent. Furthermore, intrusion detection systems would be out of date to deal with cyber-attacks that affect personal, organization, and national levels. Therefore, stepping up intrusion detection systems is necessary to cope with sophisticated cyber attacks. This section's approaches to increase intrusion detection system performance which can be categorized into two main parts, hardware-based and software-based techniques, are introduced as follows.

1. Software-based approach

To improve detection system performance in a software-based approach, Snort, an open-source, is in charge of acting as an intrusion detection system to detect suspicious network connections that would harm network systems [45].

2. Hardware-based approach

In Hardware-based approaches, there are three main ways to increase intrusion detection system performance: CPU and GPU-based, ASIC, and FPGA approaches.

The Central Processing Unit (CPU) and Graphical Processing Units(GPU) based approach helps improve network connection processing as a massive number of threads can be processed in the high volume network connection.

ASIC, known as Field Programmable Gate ArrayBased, ASIC-based approach acts as microprocessors and can perform in high volume networks in spite of high implement cost as well as low programming capability.

FPGA, known as Application Specific Integrated Circuit, FPGA-based approach provides high performance in high volume network connection with high programmability as well as reconfiguration in spite of lots of time taken to synchronize design and reprogramming [46].

2.2 Data Mining Approach

This topic discusses the implementation of data mining features with instruction detection systems (IPS) in different ways.

In this section, Snort, K-mean clustering, and Classification & Regression Trees (CART) are integrated as means to distinguish data.

The first stage is to use Snort as an intrusion detection system to detect suspicious network connections by packet capture and decode the engine according to signature patterns if any network connection packages are automatically dropped. For the second step, the network connections were analyzed by the first stage, and the K-mean clustering was categorized into groups. In the third stage, after being grouped, network connections are analyzed to find the non-attack and attack. The network connections are dropped if it is an attack-type [47].

In this section, Snort and association rules are integrated to distinguish data, and the first stage is to use Snort as intrusion detection systems to distinguish data and alert when malicious connections are found. Next, the network connections are kept in a database to let NetMIC convert dataa for apriori algorithms to find the correlation between one network connection and another. Finally, after passing apriori algorithms, MinSic is in charge of converting network connection results into Snort rules to identify packet characteristics [48].

A hybrid technique is a combination of unsupervised and supervised techniques. The first stage is to split network connection data into training sets and testing sets, and network connection data are distinguished into groups by K-mean. In the second step, network connection data in each group is identified to find packet types by a Support-vector machine (RBF) [49].

2.3 Related Techniques

In this section, we will talk about the data mining techniques used in this paper, which will focus on how it works.

2.3.1 Clustering

A clustering model is an unsupervised machine learning model without a results prototype. It is a model that groups data into a new data group according to the mutual characteristic and the same working process, such as feature selection or extraction, clustering algorithm design or selection, Cluster validation, and results from interpretation [50].

2.3.2 Classification

Classification is a supervised machine learning model. To use the classification model, data are categorized into training sets and testing sets to let the model learn data from training sets and be tested by testing sets [51]. Classification is also able to measure accuracy by using confusion matrices.

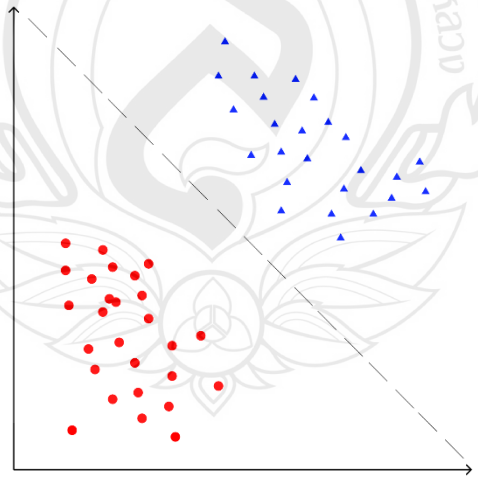


Figure 2.1 Sample data red & blue that use classification to separate the data

2.3.3 Support-vector Machine

The support-vector machine is a supervised machine learning model with associated learning algorithms that analyze data for classification and regression analysis. Support vector machines model nonlinear decision boundaries, with many optional kernel functions. In the face of over-fitting, the support vector machine has strong robustness for high dimensions.

However, support vector machines require a fair amount of skill to select the correct kernel function and are unsuitable for more extensive data [52].

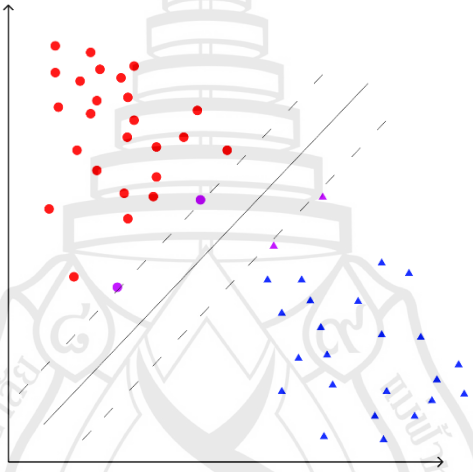


Figure 2.2 Sample data red & blue that use Support-vector machine graph and purple is the support vector

2.3.4 Decision Tree Model

The decision tree model consists of a root node as a starting point, a branch as a test result, and a leaf as an objective class. A working process of decision three models begins at the root node and goes along with branch and leave respectively and orderly [53 - 54].

ID3 is a fundamental way to build a tree model which is not for numeric data or incomplete data and to build the tree model. Data are selected according to max gain values as a starting point to build top to down the design in descending order. Another

way to build the tree model is to select appropriate attributes as a root node according to information gain and Entropy values [54].

C4.5 is an upgraded version of ID3 by using ratio values, and any attributes with max gain values are selected as a root node. C4.5 is also compatible with numeric data and missing values [54].

2.3.5 Naive Bayes Model

The Naive Bayes model is a classification algorithm that assumes that attributes in classification have no correlation to one another and does not work in many cases. The Naive Bayes model is based on conditional probability and the likelihood of occurrence [55].

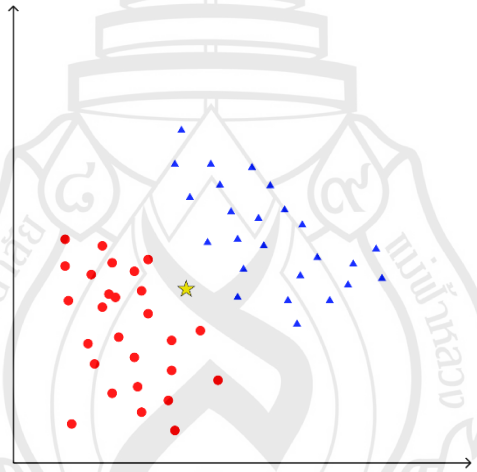


Figure 2.3 Sample data red & blue that use Naive Bayes and the yellow star is naive bayes

2.3.6 Logistic Regression

Logistic regression measures the probability of an occurrence, such as yes or no. Most of the logistic regressions are binary results. Furthermore, it can simulate discrete relationships [56].

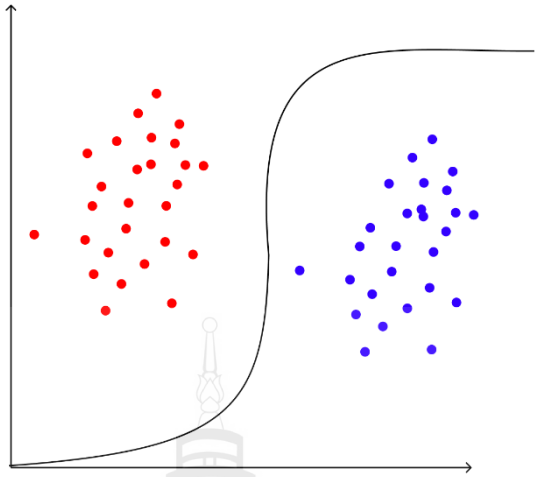


Figure 2.4 Sample data red that use Logistic regression graph

2.3.7 K-means Clustering Algorithm

K-means clustering algorithm requires no labeled response for the given input data; it is also the unsupervised technique.K-means clustering algorithm is used in data mining and widely used clustering algorithms.

K-means clustering algorithm has been used as part of many other algorithms since it is simple, trustable, promising, and mathematically tractable [57].

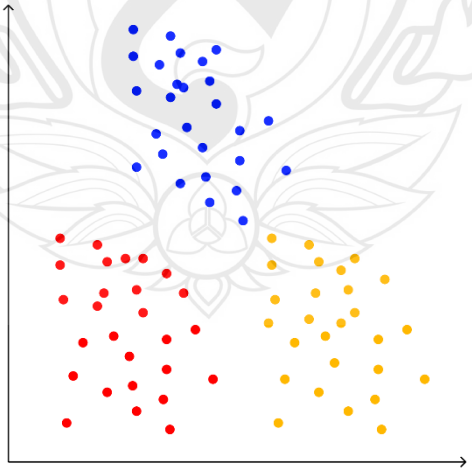


Figure 2.5 Sample data red & blue & yellow that use K-means clustering algorithm graph to separate the sample data

2.3.8 Imbalance Data Classification

Most of the classifications use a well-balanced data set for good results. However, when there is a large difference in data set volume, the classification has an imbalance bias. Therefore, techniques must be used to manage them. There are mainly two types of techniques.

Preprocessing techniques

Is a technique for preparing different data sets with imbalanced quantities to be balanced with all data sets before classification.

The resampling technique is a data sampling technique. To balance different volume sets with other data sets to achieve a variety of datasets when doing classification, There are three sub-methods:

Over-sampling methods are the sampling and synthesizing of the data of the minority group to have the data volume close to that of the majority group. This may result in poor efficiency because it uses repetitive sampling of low volumes of data to create duplicate data.

Under-sampling methods use the opposite technique of over-sampling. It is to randomly reduce the amount of data of the majority group to balance the minority group of random data to reduce the duplication of the data. However, the under-sampling Randomly reducing the data in the majority group may reduce the efficiency or ignore the valuable and important information in that sample [58].

Hybrid methods are a combination of Over-sampling and Under-sampling methods [59]

Cost-sensitive learning is a misclassifying example of a minor group, where Methods based on training data modification are to modify the decision criteria or assign weights to the instance when sampling the training dataset from the cost decision matrix [59].

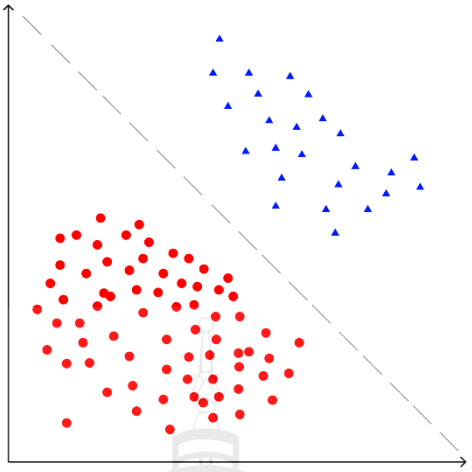


Figure 2.6 Sample data red & blue that use Imbalance data classification graph

2.3.9 Random Subspace

The random subspace method can be used with any machine learning algorithm, although it is well suited to models that are sensitive to large changes to the input features, such as decision trees and k-nearest neighbors [60].

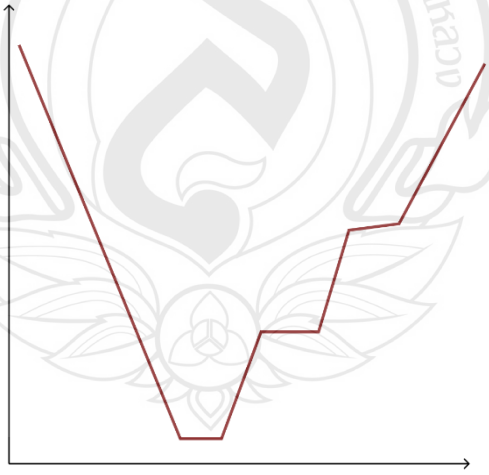


Figure 2.7 Sample data of red lines that use random subspace

CHAPTER 3

METHODOLOGY

3.1 Data Collection and Preparation

Based on the work of [23], the study aims to improve the classification of network intrusion, which belongs to the non-payload approach. In particular, it inspects packet details only at TCP/IP layers but not deep packet content (i.e., packet payload). Note that the dataset acquired within this investigation is based on objects of TCP connections (with multiple possible packets being summarized as one), not single packets. In addition to the objective of recognizing non-payload-based evasions, this study also considers obfuscations (or mutated variations) of network attacks that have drawn a lot of attention [61 - 62].

For obfuscated attacks, there are ways for an adversary to modify the input of the network system, as it has to conform with the protocol specification of the victim's TCP/IP stack. These include modifying exploit code, appending padding at the application layer of exploit code, and artificially influencing network or transport layer protocols. On the one hand, many have attempted to model exploit-dependent obfuscations, which use a massive database of existing exploits. This is a time-consuming method to disguise attack signatures already known to the up-to-date intrusion detection system. However, it is not directly applicable to new exploits. Conversely, it is easier to design non-payload-based obfuscation techniques working at TCP/IP layers, which will mutate instances of known intrusions in an exploit-independent way. This can make attacks similar to legitimate traffic.

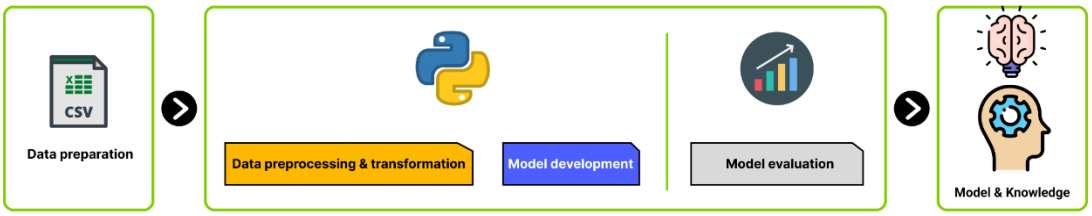


Figure 3.1 The overview of process design

3.1.1 Data Characteristics

According to [23], the ASNM-NPBO dataset (Advanced Security Network Metrics & Non-Payload-Based Obfuscations) is used to evaluate the robustness of machine-learning-based NIDS against obfuscated attacks. In order to obtain this collection of data, A connection representing a legitimate and intrusive attack is established on a vulnerable network service. Note that obfuscation intrusion occurs through obfuscation techniques with some direct attacks so that some parts of the look are similar to the originals. Then, the TCP-level feature extractor called ASNM: Advanced Security Network Metrics [63] is deployed to generate the corresponding feature space. After normalizing value domains, the resulting dataset $X = V \times Y$ consists of a normalized feature space $V \in [0, 1]^{11,445 \times 194}$, where a class of an instance $v_i \in V$ is drawn from the domain of 3 possible connection categories, i.e., $y_i \in Y$, $y_i \in$ {Legitimate, Direct attack, Obfuscated attack}. In Table 3.1, details of this dataset concerning the number of class-specific samples generated using different network services. In addition, Table 3.2 illustrates ASNM feature categories and those 14 ones that have been selected in the report of [23] for classifying instances of legitimate connections and direct attacks (please consult [63] for a complete list of 194 features). The current research focuses on this selection so as not to bias a predictive model with features highly associated with obfuscated cases, thus reducing the feature space to $V \in [0, 1]^{11,445 \times 14}$. Please consult the original data publication [64] for simulation and related settings details.

1. Total data record is 11445.
2. Total data column is 14.
3. Data are both numeric and nominal.

Table 3.1 Distribution of TCP connection objects in collected dataset

Network Service	Count of TCP Connection			
	Legitimate	**Direct Attacks**	**Obfuscated Attacks**	**Summary**
Apache Tomcat	809	61	163	1033
Dist CC	100	12	23	135
MS SQL	532	31	103	666
PostgreSQL	737	13	45	795
Samba	4641	19	44	4704
Server	3339	26	100	3465
Other Legitimate Traffic	647	n/a	n/a	647
Summary	**10805**	**162**	**487**	**11445**

Table 3.2 ASNM feature categories (with a full list of features available in the work [63]) and those 14 features chosen by the study of [23]. Note that FFT denotes Fast Fourier Transformation

Category (total:selected)	**Feature name & description**
Statistical (77:4)	SigPktLenOut; standard deviation of outbound packet lengths MeanPktLenIn; mean of packet sizes in inbound traffic of a connection ConTcpFinCntIn; number of inbound packets of a connection with FIN flag set ConTcpPshCntIn; number of inbound packets of a connection with PSH flag set
Localization (8:0)	n/a
Distributed (34:0)	n/a
Dynamic (32:0)	n/a

Table 3.2 (continued)

Category (total:selected)	Feature name & description
Behavioral (43:10)	CntOfOldFlows; no. of mutual flows between client/server hosts of analyzed connection 5 mins before it started CntOfNewFlows; no. of mutual flows between client/server hosts of analyzed connection 5 mins after it finished FourGonModulIn[1]; the module of the 2nd coefficient of the FFT in goniometric representation FourGonModulOut[1]; the same as the previous one, but for outbound traffic FourGonAngleN[9]; the angle of the 10th coefficient of the FFT in goniometric representation FourGonModulN[0]; the same as the previous one, but it represents the module of the 1st coefficient of the FFT PolyInd3ordOut[3]; the same as the previous one, but it represents the 4^{th} coefficient of the approximation GaussProds8Out[7]; the same as the previous one, but computed above outbound packets and represents a product of the 8th slice of packets with a Gaussian function that fits to the interval of the packets' slice OutPktLen32s10i[3]; the same as the previous one, but computed above the first 32 secs of a connection. It is totaled outbound packet lengths of the 4th interval OutPktLen4s10i[2]; the same as the previous one, but computed above the first 4 secs of a connection. It is totaled outbound packet lengths of the 3rd interval

	id	label_2	label_poly	label_poly_o	srcIP	dstIP	srcPort	dstPort	srcMAC	dstMAC
0	1	False	3_Other	3_Other	3.125.56.111	3.125.56.202	33772	4444	08:00:27:f5:32:14	08:00:27:85:b2:60
1	2	False	3_Other	3_Other	3.128.56.111	3.128.56.202	60895	4444	08:00:27:f5:32:14	08:00:27:85:b2:60
2	3	False	3_Other	3_Other	3.129.56.111	3.129.56.202	35776	4444	08:00:27:f5:32:14	08:00:27:85:b2:60
3	4	False	3_Other	3_Other	3.131.56.111	3.131.56.202	43105	4444	08:00:27:f5:32:14	08:00:27:85:b2:60
4	5	False	3_Other	3_Other	3.139.56.111	3.139.56.202	56987	4444	08:00:27:f5:32:14	08:00:27:85:b2:60
...	...	...	...	...	...	...	...	...	...	...
11440	11441	False	3_Samba	3_Samba	195.245.169.175	108.42.1.237	53383	139	00:13:3b:9a:12:fc	00:25:b3:77:20:53
11441	11442	False	3_Server	3_Server	252.123.97.150	140.214.110.215	50931	445	b8:e8:56:24:a2:a0	70:54:d2:4b:66:3a
11442	11443	False	3_Server	3_Server	82.103.193.98	142.110.71.140	51090	445	e8:80:2e:e6:fb:4a	00:e0:7d:82:9f:d9
11443	11444	False	3_Samba	3_Samba	53.4.81.100	108.42.1.237	50114	139	00:13:3b:9b:57:8e	00:25:b3:77:20:53
11444	11445	False	3_Server	3_Server	82.103.193.98	144.176.192.18	50891	445	e8:80:2e:e6:fb:4a	ec:9a:74:57:09:a3

11445 rows × 904 columns

Figure 3.2 Show sample data and number of rows and columns

Data categorized into three groups

1 = Direct Attacks (162) 1.4%

2 = Obfuscated Attacks (478) 4.2%

3 = Legitimate = (10805) 94.4%

Figure 3.3 The figure shows the percentage from data set by three types (Direct attacks, Obfuscated attacks and Legitimated) in the form of a pie chart

3.1.2 Data Preparation

Select the ASNM feature categories used in table 3.2 by the script used in Figure 3.4 and the example in Figure 3.5

```
selectDf = df[['SigPktLenDst','MeanPktLenSrc','finCnt<In>','pshCnt<In>','cntOfOldFlows','cntOfNewFlows','fourCoefsGonModulIn[1]'
,'fourCoefsGonModulOut[1]','fourCoefsGonAngleIn[9]','fourCoefsGonModulIn[0]'
,'polynomIndexes3ordOut[3]','gaussProds8Out[7]','OutPktLen32s10i[3]','OutPktLen4s10i[2]']]
```

Figure 3.4 Show command python code to select the columns (feature)

SigPktLenDst	MeanPktLenSrc	finCnt<In>	pshCnt<In>	cntOfOldFlows	cntOfNewFlows	fourCoefsGonModulIn[1]	fourCoefsGonModulOut[1]	fourCoefsGonA
5.656854	68.000000	1	0	4	0	8.000000	8.0	
5.656854	68.000000	1	0	3	2	8.000000	8.0	
5.656854	68.000000	1	0	3	2	8.000000	8.0	
5.656854	68.000000	1	0	4	1	8.000000	8.0	
5.656854	68.000000	1	0	6	0	8.000000	8.0	
...	...	...	...	...	...	...	...	
5.656854	68.000000	1	0	3	0	8.000000	8.0	
7.023769	68.000000	0	0	0	5	12.165525	8.0	
7.023769	68.000000	0	0	2	5	12.165525	8.0	
5.656854	68.000000	1	0	1	3	8.000000	8.0	
9.899495	68.666667	0	0	0	5	14.000000	8.0	

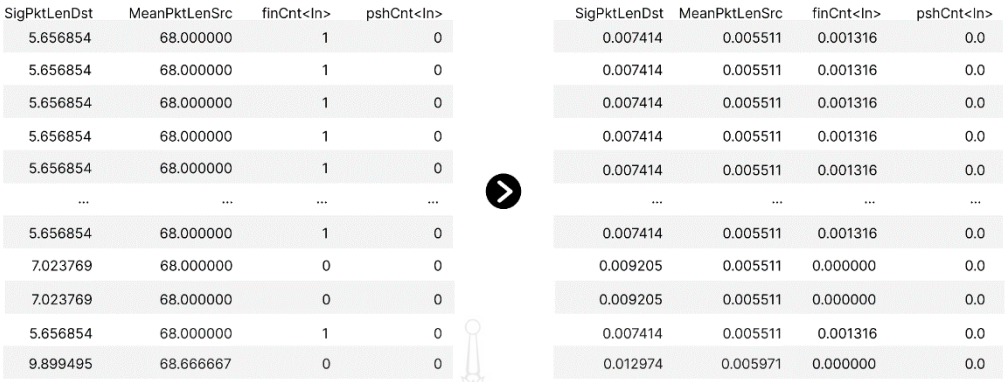
rows × 14 columns

Figure 3.5 Show a sample data of the features that have been selected

Perform normalization in the form of min-max scaler as shown in Figure 3.6 in order to format the data in the same range, resulting in better processing of the model as shown in Figure 3.7

```
x = selectDf.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
normalized_df = pd.DataFrame(x_scaled, columns=selectDf.columns)
```

Figure 3.6 Show the code of normalized min-max scaler



SigPktLenDst	MeanPktLenSrc	finCnt<In>	pshCnt<In>
5.656854	68.000000	1	0
5.656854	68.000000	1	0
5.656854	68.000000	1	0
5.656854	68.000000	1	0
5.656854	68.000000	1	0
...	...	...	...
5.656854	68.000000	1	0
7.023769	68.000000	0	0
7.023769	68.000000	0	0
5.656854	68.000000	1	0
9.899495	68.666667	0	0

SigPktLenDst	MeanPktLenSrc	finCnt<In>	pshCnt<In>
0.007414	0.005511	0.001316	0.0
0.007414	0.005511	0.001316	0.0
0.007414	0.005511	0.001316	0.0
0.007414	0.005511	0.001316	0.0
0.007414	0.005511	0.001316	0.0
...	...	...	...
0.007414	0.005511	0.001316	0.0
0.009205	0.005511	0.000000	0.0
0.009205	0.005511	0.000000	0.0
0.007414	0.005511	0.001316	0.0
0.012974	0.005971	0.000000	0.0

Figure 3.7 An Example data use normalized min-max scaler

3.1.3 Non-Payload-Based Obfuscation

The current work investigates instances of TCP connection, not network packets, which represent application data exchanges between client and server on the TCP/IP stack up to the transport layer. Of course, these are subject to connection-oriented protocol TCP at L4, Internet protocol IP at L3 and Ethernet protocol at L2, respectively. In particular, a TCP connection γ can be presented by start and end timestamps, ports/IP addresses of the client and the server, and sets of packets interchanged between the two ends. Given this assumption, a connection γ can be explained with its network connection features, thus allowing the following extraction function to map γ to the feature space Λ of d dimensions.

$$f(\gamma) \mapsto \Lambda, \Lambda = (\lambda_1, \lambda_2, \dots, \lambda_d)$$

Each function $f_i(\gamma)$ maps the connection to the i^{th} dimension as follows.

$$f_i(\gamma) \mapsto \lambda_i, \qquad i = \{1,2, \dots, d\},$$

According to the research [30], examples of these features include the standard deviation of outbound (client to server) packet sizes, modus of TCP header lengths in all traffic, the number of TCP URG flags occurring in inbound traffic, and the number of TCP FIN flags occurred in inbound traffic. Having specified those, the next part describes the concept of non-payload-based obfuscation, which aims to modify connection characteristics or features for a remote attack. For a connection γ_a

representing a remote attack without any obfuscation, it can be represented by the following equation.

$$f(\gamma_a) \mapsto \Lambda^a, \Lambda^a = (\lambda_1^a, \lambda_2^a, \dots, \lambda_d^a)$$

Then, let's turn to the connection $\gamma_{a'}$ that corresponds to an intrusive communication γ_a to which non-payload-based obfuscations are applied. These modifications change packet sets of the original connection γ_a by insertion, removal, and transformation of the packets. As such, the previous feature space Λ a is transformed to $\Lambda^{a'}$.

$$f(\gamma_{a'}) \mapsto \Lambda^{a'}, \Lambda^{a'} = \left(\lambda_1^{a'}, \lambda_2^{a'}, \dots, \lambda_d^{a'}\right)$$

As a result, a classifier trained without knowledge of obfuscated or modified patterns may not perform as well as it does against intrusive connections with original features. According to the research study [30], a set of techniques has been initiated as part of developing an obfuscation tool in the Unix environment. Examples of functions $f(\gamma_{a'})$ are listed below

1. Spread out packets in time: constant delay of 1 and 8 s., and the normal distribution of delay with 165 5 s. mean with 2.5 s. standard deviation (25% correlation)
2. Packets loss: 25% of packets
3. Unreliable network channel simulation: 25% of packets damaged, 35% of packets damaged, and 35% of packets damaged with 25% correlation
4. Packets duplication: 5% of packets
5. Packet order modifications: reordering 25% and 50% packets; reordered packets are sent within 10 ms. delay and 50% correlation
6. Fragmentation: MTU 1000, MTU 750, MTU 500 and MTU 250
7. Combinations of the techniques mentioned above

3.1.4 Problem Definition of NIDS as Classification

In accordance with previous studies of classification-based NIDS [20 - 22], let a training dataset $X = V \times Y$ be the space of labeled connection instances, where V denotes the feature space of n instances ($V \in R^{n_X d}$), Y represents the corresponding label space of size $n \times 1$, and each entry $x_i \in V$ is classified as $y_i \in V$ with the value of yi being drawn from the domain of class D_X. For a classifier that is trained on the dataset X using the algorithm α, the resulting model CF_x^α estimates the class $y_o \in D_X$ of a new instance $x_o \in R^{1 \times d}$, i.e., $CF_x^\alpha(x_O) = y_o$. Specific to the context of NIDS as a binary classification problem, the predicted class $y_{\gamma a}$ of a connection γ_a whose feature vector is defined as $f(\gamma_a)$, can be defined by the following.

$$y_{\gamma_a} = CF_X^\alpha\big(f(\gamma_a)\big)$$

where $y_{\gamma_a} \in$ {Intrusion, Legitimate}. Now with a connection $y_{\gamma_{a'}}$ whose features are modified through obfuscation functions, the quality of prediction $y_{\gamma_{a'}}$ is the subject worth investigating.

$$y_{\gamma_{a'}} = CF_X^\alpha\big(f(\gamma_{a'})\big)$$

Without any prior knowledge regarding mutated patterns or model adjustment, any classifier CF_X^α can often be sub-optimal. To this extent, the researchers [23] overcomes this difficulty through a feature selection approach, which iteratively adds highly informative features to the desired set. This leads to an improvement towards a robust classification of adversarial intrusions but lacks appropriate handling of the class imbalance problem. Henceforth, a new undersampling method is proposed in the next section to attend to this issue.

3.2 Model Development and Evaluation

This section explains that the proposed undersampling framework can be considered an extension of the research [32], which employs a single clustering to determine representative instances of original samples belonging to the majority class. At the same time, it demonstrates an organic application of the consensus clustering concept [38 - 65] to provide a pool of multiple clustering results from which better cluster-wise representatives can be extracted.

3.2.1 Experimental Overview

This section provides details of different stages developed for this new method, including creating multiple clustering results, selecting representatives from those data partitions, and using data after undersampling to train a classifier, respectively.

It Generated a Pool of Multiple Clusterings. Provided the dataset X, let X_o and X_1 (where $X_0 \cup X_1 = X$) be the set of samples belonging to the majority class and that of the minority class, respectively. An undersampling technique χ is applied to X_0 to extract the set of representative samples $\chi(X_0) = X_0^*$, where $|X_0^*| \ll |X_0|$ and $|B|$ denotes the size of set B. Then, the target data X^* that can be formed as $X^* = X_1 \cup X_0^*$ is used to create a classifier $CF_{X^*}^{\alpha}$ using the classification algorithm α (previously, this is CF_X^{α} without the undersampling process). Based on the study presented by research [21], the function χ is simply represented by a set of centroids Z obtained from a clustering of X_0, where the k-means technique is a common alternative for this analysis. In other words, representative samples are centroids $z_1, z_2, \dots, z_p$from clustering X_o using k-means and the preferred number of clusters as ρ. Note that ρ is set to the desired size of majority class, which normally is the same as that of the other, i.e., $\rho = |X_1|$.

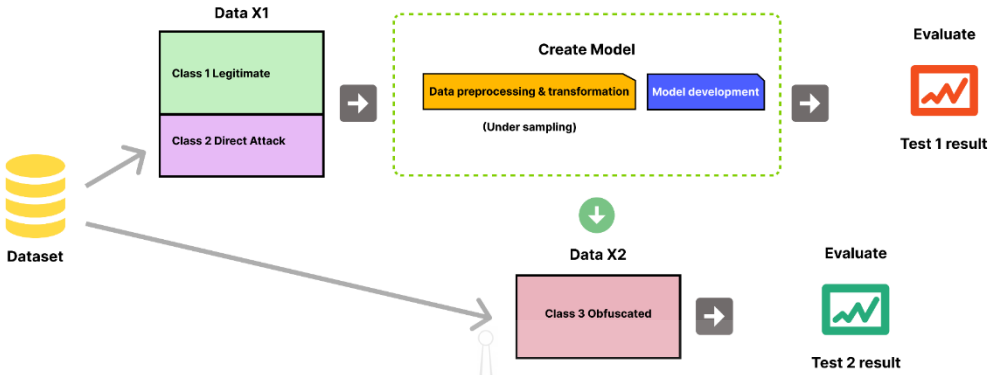


Figure 3.8 The Overview of Experimental

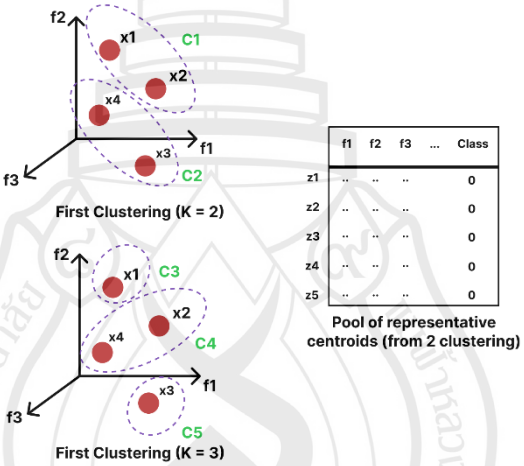


Figure 3.9 Clustering with K-mean random by pool of representative centroids from 2 clustering

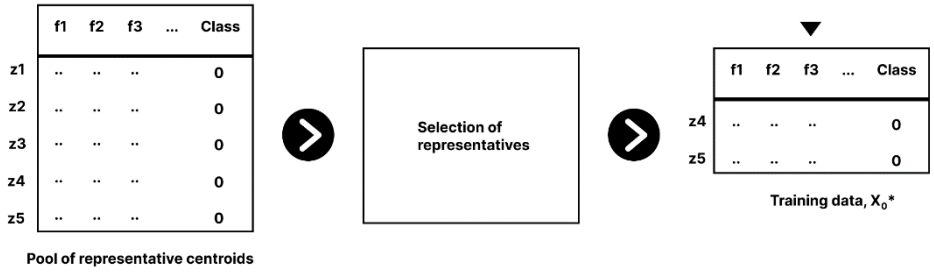


Figure 3.10 Select Cluster wise representation by pool of representative centroids to selection of representative and training data, X_0*

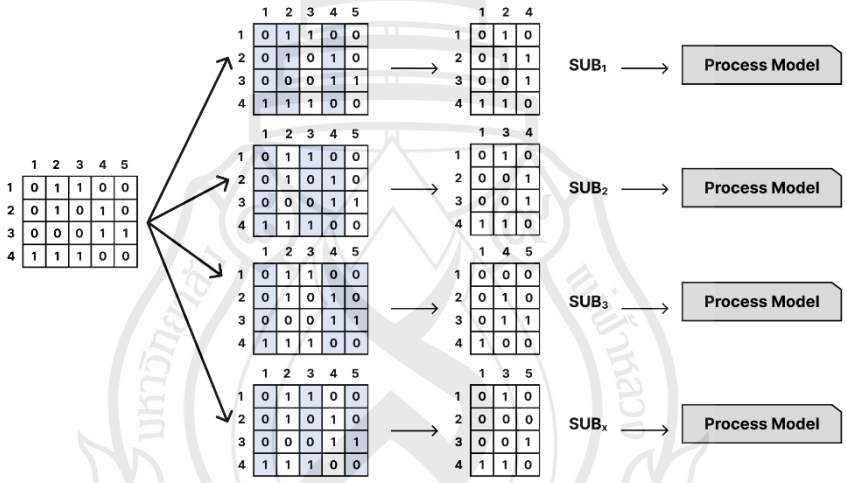


Figure 3.11 Random subspace approach process model method

Specific to the current work, the concept ensemble or consensus clustering [39] is exploited to create a pool of multiple clusterings, from which a collection of ρ clusters are selected to generate X_0^* . Let $V_0 = \{x_1, x_2, \dots, x_{n_0}\}$ be the matrix in the normalized domain $[0,1]^{n_O \times d}$ of n_0 legitimate connection instances with respect to d features. It is noteworthy that the label space Y_0 from a training dataset $X_0 = V_O \times Y_0$ will not be exploited here as a clustering process is an unsupervised model, which develops a data partition (i.e., a set of clusters) without the knowledge of class information. In addition, each sample $x_i \in V_0$is represented as a vector of d feature dimensions or $x_i =$

$(x_{i1,\dots}, x_{id})$, $\forall i \in \{1,2,\dots,n_0\}$. Also let $\Pi = \{\pi_1, \pi_2, \dots, \pi_M\}$ be a cluster ensemble with M base clusterings, i.e., a clustering result or ensemble member. In particular, the g^{th} member delivers a collection of clusters $\pi_g = \{C_1^g, C_2^g, \dots, C_{kg}^g\}$, where $\bigcup_{t=1}^{k_g} C_t^g = X_0$, $\bigcap_{t=1}^{k_g} C_t^g = \emptyset$, and k_g is the number of clusters in partition π_g. To ensure the diversity within Π, the following ensemble generation strategies are used together.

1. Random-k method: these clusterings are generated using k-means with a cluster number that is randomly chosen from the range $\{2, \dots \sqrt{|X_0|}\}$ or $\{2, \dots 50\}$ when $\sqrt{|X_0|} > 50$ (see the report of report [54] for relevant details).

2. Random-subspace method: each base clustering can be generated from a random feature subspace $V_0' \in [0,1]^{n_0 \times d'}$ of the feature space V_0. Each of the data subspaces is created with respect to the following interval d'.

$$d' = d'_{min} + \lfloor \epsilon (d'_{max} - d'_{min}) \rfloor ,$$

provided that $\epsilon \in [0,1]$ is a uniform random variable, while d'_{min} and d'_{max} denote the lower and upper bounds of the generated subspace V_0'. Following the initial work of [66], d'_{min} and d'_{max} are set to $0.75d$ and $0.85d$, respectively. With this being decided, one of d features is picked up at a time to form the desired subspace of d' non-duplicated features are obtained. For that, the index of each randomly selected feature is determined by the following.

$$h = \lfloor 1 + \eta D \rfloor ,$$

where h denotes the h th feature in the pool of d attributes and $\eta \in [0, 1)$ is another uniform random variable.

Selecting Cluster-Wise Representatives. Having obtained the ensemble Π, a pool of centroids Z_Π is created such that each centroid $z_a \in Z_\Pi$ represents a particular cluster C_a in the set $\{C_1^1, C_{2'}^1, \dots, C_{k_1}^1\} \cup \{C_1^2, C_2^2, \dots, C_{k_2}^2\} \cup \cdots \{C_1^M, C_2^M, \dots, C_{k_M}^M\}$. Note that every centroid in this collection is represented by the original space of d features, i.e., $z_a \in [0,1]^{1 \times d}$, $\forall z_a \in Z_\Pi$. This processing stage is to select ρ of these centroids to form the target set of representative samples $Z = \{z_1, z_2, \dots, z_\rho\}$. It can be summarized by the following steps.

1. Step 1: To start with, the first member z_1 of Z is selected from the Z_Π, provided that z_1 maximizes the average distance to other centroids in Z_Π.

$$z_1 = \forall z_a \in Z_\Pi argmax \frac{\sum_{\forall z_b \in Z_\Pi, z_b \neq z_a} d(z_a, z_b)}{|Z_\Pi| - 1}$$

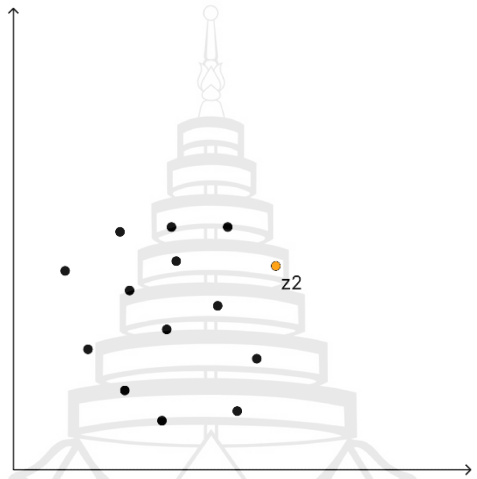


Figure 3.12 Show sample Furthest-first of z2

where $d(x_i, x_j)$ denotes the distance between two samples x_i and x_j . Note that the Euclidean metric is employed in this research to estimate the distance measurement. In addition, $|A|$ represents the size of set A. At the end of this step, $|Z|$ is 1, while that of Z_Π is reduced by 1 as well, i.e., $Z_\Pi = Z_\Pi - z_1$.

2. Step 2: Next, a new member is iteratively chosen from Z_Π and moved to Z. To be exact, in each iteration, a greedy optimization is exploited to determine the best centroid $z_c \in Z_\Pi$ using one of three different objective functions defined below. As a result, the two sets are updated by $Z = Z \cup z_c$ and $Z_\Pi = Z_\Pi - z_c$, respectively. This is repeated until all ρ members are obtained, i.e., $|Z| = \rho$. The following equation describes the first objective function, called *Furthest-First-Majority* or shortly as *FF-Majority*(Z, Z_Π). This is to find a centroid that represent a unique feature space that is minimally overlapping with those of others in Z_Π. It is leveraged with a similar assessment to existing members of Z, which is the first term in this function.

$$z_c = \forall z_a \in Z_\Pi argmax\left(\frac{\sum_{\forall z_e \in Z_\Pi} d(z_a, z_e)}{|Z|} + \frac{\sum_{\forall z_b \in Z_\Pi, z_b \neq z_a} d(z_a, z_b)}{|Z_\Pi| - 1}\right)$$

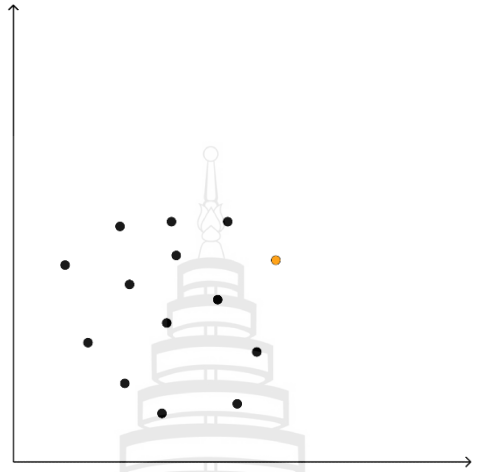


Figure 3.13 Show sample Furthest-first-Majority

The second objective function, called *Furthest-First-Minority* or shortly as *FF-Minority*(Z, Z_Π, X_1). It finds a centroid in Z_Π that is largely different from samples of the minority class X_1. This can be formally specified as follows.

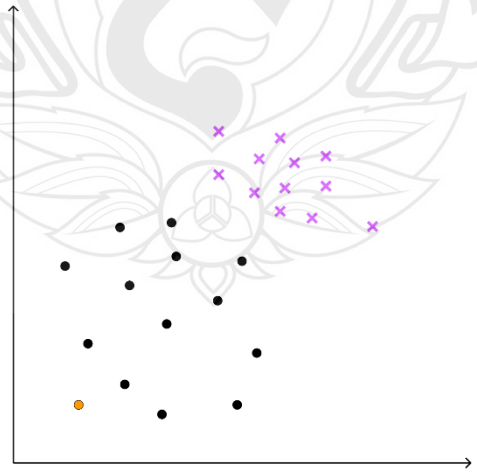


Figure 3.14 Furthest-First-Minority

$$z_c = \forall z_a \in Z_{\Pi} argmax\left(\frac{\sum_{\forall z_e \in Z_{\Pi}} d(z_a, z_e)}{|Z|} + \frac{\sum_{\forall x_i \in X_1} d(z_a, x_i)}{|X_1|}\right)$$

And the third alternative of objective function, called *Furthest-First-Overall* or shortly as *FF-Overall*(Z, Z_{Π}, X_1), combines the two previous functions to gain the overall justification based on samples belonging to both majority and minority classes.

$$z_c = \forall z_a \in Z_{\Pi} argmax\left(\frac{\sum_{\forall z_e \in Z_{\Pi}} d(z_a, z_e)}{|Z|} + \frac{1}{2}\left(\frac{\sum_{\forall z_b \in Z_{\Pi}, z_b \neq z_a} d(z_a, z_b)}{|Z_{\Pi}| - 1} + \frac{\sum_{\forall x_i \in X_1} d(z_a, x_i)}{|X_1|}\right)\right)$$

Application to Training a Classifier. The set of sample representatives Z acquired from the last phase corresponds to the features space $V_0^* \in [0,1]^{\rho \times d}$ of the majority class $Y_0^* = \{1\}^{\rho \times 1}$, i.e., $X_0^* = V_0^* \times Y_0^*$. The resulting training set is to aggregate this with the set of samples assigned to the minority class, or $X^* = X_1 \cup X_0^*$. Consequently, a classifier $CF_{X^*}^{\alpha}$ can be trained with the balanced data X^* using the choice of classification algorithm α. As such, the prediction $y_{\gamma_{a'}}$ of a connection instance whose features are altered by different obfuscation techniques is determined by the following definition.

$$y_{\gamma_{a'}} = CF_{X^*}^{\alpha}(f(\gamma_{a'}))$$

It is noteworthy that other methods to handle the class imbalance problem can also be used in this way to generate the data X^*. These will be included in the empirical study reported next.

CHAPTER 4

EXPERIMENTAL RESULT

After defining the problems during the examination and the methods proposed in the last two parts, this paper continues with an empirical study in which the new under-sampling framework is assessed against published results in the original work and other methods to deal with the problem of level imbalance class. Results and discussion are especially included in this section.

4.1 Basic Experiment and Experimental Design

The dataset under investigation was obtained from the original study [23], which evaluated NIDS's robustness against obfuscated attacks using machine learning. To collect this data, connections that represent legitimate and intrusive attacks are established on vulnerable network services. Note that the obfuscated intrusions occur through obfuscation techniques to some direct attacks. So that some of the appearances partly resemble the original. Then, a TCP-level feature extractor called ASNM: Advanced Security Network Metrics [63] is deployed to generate the corresponding feature space. After normalizing the domain result dataset to $X = V \times Y$ consists of a normalized feature space $V \in [0, 1]^{11,445 \times 194}$, where a class of an instance $v_i \in V$ is drawn from the domain of 3 possible connection categories, i.e., $y_i \in Y$, $y_i \in$ {Legitimate, Direct attack, Obfuscated attack}. In Table 3.1, details of this dataset are presented for some class-specific samples generated using various network services. In addition, Table 3.2 illustrates ASNM feature categories and those 14 ones that have been selected in the report of [23] for classifying instances of legitimate connections and direct attacks (please consult [63] for a complete list of 194 features). The current

research focuses on this selection to avoid bias in the predictive models that feature high relevance to obfuscated cases, thus reducing the feature space to $V \in [0, 1]^{11,445 \times 14}$.

For a thorough evaluation, a rich collection of compared methods are included in this empirical study, in addition to the proposed framework with three different objective functions: FF-Majority, FF-Minority, and FF-Overall, respectively. The 'Baseline' of these new models is initially introduced by the original study [23], i.e., a classifier is developed from the original dataset X with the presence of the imbalanced class problem. Another primary competitor is the single clustering technique proposed [29], referred to as 'SingleClus' hereafter. Also, other undersampling algorithms are investigated, including RUS or Random UnderSampling [36] and its extensions to an ensemble approach. These include RUSBoost [36] and IRUS or Inverse Random UnderSampling [37], with their hyper-parameters being set according to the comparative study [67]. Note that the target size of the majority class after an undersampling process is the number of samples belonging to the minority. Besides those undersampling and ensemble models, this assessment also explores the oversampling counterpart that increases the cardinality of X_1 such that $|X_1| = |X_0|$. Specific to this work,

The benchmark SMOTE technique [28] and its recent variant named 'Outlier-SMOTE' are employed, using the parameter setting recommended in the original report [68]. This extension is picked up because it uses a similar instinct to the goal FF-Majority, where samples that are primarily different from others will be considered more critical. Other settings are summarized as follows.

1. For the proposed framework, the target size of the multiple-clustering pool or M is set to 150, and each specific set of these models is repeated for 30 trials to achieve a reliable conclusion from non-deterministic processes based on averages across multiple runs.

2. For the classification algorithm α, four techniques are included here. These include decision tree (C4.5) with the maximum depth of 15, Naive Bayes (NB) with the Gaussian kernel function, support vector machine (SVM) with the Radial kernel function, and Logistic Regression (LR), respectively. These are employed with the dataset generated by those filter and wrapper techniques to create a classifier.

3. Since the current research focuses on the robustness of machine-learning based NIDS to obfuscated intrusions, a classifier must be trained with instances representing legitimate connections and direct attacks only. Without the knowledge of those obfuscated instances, this study to see how well different methods recognize unseen threats. As such, the stratified 10-fold cross-validation is firstly applied to classifying legitimate connections and direct attacks, i.e., the examined data is the combination of instances belonging to these two classes only. The corresponding results illustrate the quality of classifiers to capture usual patterns. On top of that, the classifier trained in each fold will be used to predict obfuscated intrusions as either legitimate or attacks. The last experiment leads to the comparison of robustness as a classifier encounters truly new intrusive connections. At last, metrics used to assess predictive performance are TRP (True Positive Rate), FPR (False Positive Rate) and F1, respectively.

The researchers' goal was to know to predict the obfuscated connection and how effective the experimental outcome would be.

4.2 Result and Discussion

4.2.1 Experimental I

The experimental result by using Homoliak stage 1 method in Figure 4.1 for the experiment will show the comparison of F1 & TPR. The dataset that was used are Legitimate & Direct attack data of 10-fold cross validation to data training by using four classifier Decision tree (C4.5), Naive Bayes (NB), Support Vector Machine (SVM), and Logistic Regression (LR) after obtaining the training data Legitimate & Direct attack result with three different objective function: FF-Majority, FF-Minority, and FF-Overall that give 30 of the data sample that makes experiment table have 100 samples include Baseline 10 sample, SingleClus 10 sample, RUS 10 sample, RUSBoost 10 sample, IRUS 10 sample, SMOTE 10 sample, Outliner-SMOTE 10 sample.

For the overview of experimental I results of F1 & TPR and comparison of F1& TPR. The F1 percent is 90.40%, 89.48%, 91.52% of FF-Majority, FF-Minority, and FF-Overall in Figure 4.1 and the TPR percent are 88.27%,87.19%,89.35% of FF-Majority, FF-Minority, and FF-Overall respectively in Figure 4.1

4.2.1.1 F1 and True Positive Rate (TPR) Result

The experimental results of the comparison result of F1 and TPR following by Figure 4.1 by implementing the FF-Majority, FF-Minority, and FF-Overall methods to improve the F1 metric achieved and using four classifier Decision tree (C4.5), Naive Bayes (NB), Support Vector Machine (SVM), and Logistic Regression (LR) for cope the undersampling data. As for the result ,these are satisfactory. The average data of F1 & TPR are increasing their percentage number which means the data have high chance of prediction, which can use in further development, Baseline 89% & 85.60%, SingleClus 87.31% & 84.88%, FF-Majority 90.40% & 88.27%, FF-Minority 89.48% & 87.19%, FF-Overall 91.58% & 89.35%, RUS 84.17% & 81.48%, RUSBoost 86.60% & 83.80%, IRUS 86.39% & 83.64%, SMOTE 89.48% & 87.19%, Outlier-SMOTE 89.97% & 87.50%, respectively. By the result FF-Overall has the highest percentage number of F1 & TPR, average measures across 4 classifiers.

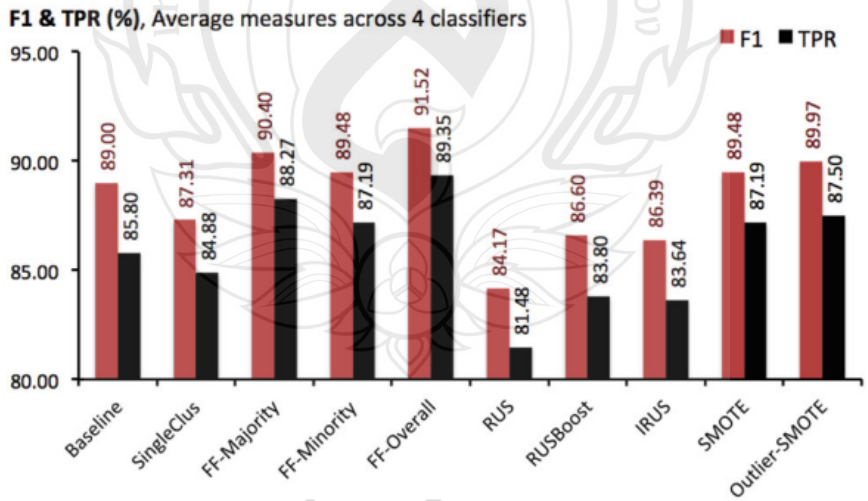


Figure 4.1 Comparison of F1 and TPR scores obtained by examined methods, as averages across classification models and 30 trials of 10-fold cross validation

4.2.1.2 False Positive Rate (FPR) Result

The FPR result followed by Figure 4.2 by implementing the FF-Majority, FF-Minority, and FF-Overall methods to reduce the FPR achieved by using four classifier Decision tree (C4.5), Naive Bayes (NB), Support Vector Machine (SVM), and Logistic Regression (LR) for cope the undersampling data. As for the result, these are satisfactory. The average data of FPR are decreasing their percentage number which means the data have a less chance of failure prediction, which can use in further development. Baseline 0.09%, SingleClus 0.13%, FF-Majority 0.10%, FF-Minority 0.11%, FF-Overall 0.08%, RUS 0.17%, RUSBoost 0.14%, IRUS 0.14%, SMOTE 0.11% ,Outlier-SMOTE 0.10%, respectively. By the result, FF-Overall has the least percentage number of FPR, average measures across four classifiers.

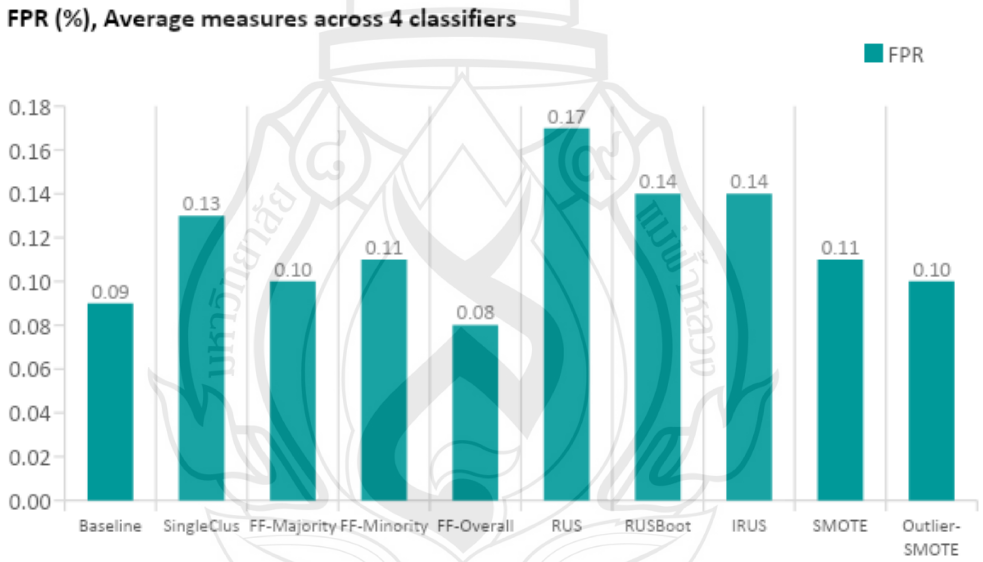


Figure 4.2 Comparison of FPR scores obtained by examined methods, as averages across classification models and 30 trials of 10-fold cross validation

Table 4.1 Investigated Method Classifier Decision tree (C4.5) and 30 trials of 10-fold cross validation

Investigated Method	F1	TPR
Baseline	0.9419	0.9506
SingleClus	0.9174	0.9259
FF-Majority	0.9444	0.9444
FF-Minority	0.9419	0.9506
FF-Overall	0.9506	0.9506
RUS	0.8910	0.8827
RUSBoost	0.9108	0.9136
IRUS	0.9141	0.9198
SMOTE	0.9451	0.9568
Outlier-SMOTE	0.9480	0.9568

Table 4.2 Investigated Method Classifier Naive Bayes (NB) and 30 trials of 10-fold cross validation

Investigated Method	F1	TPR
Baseline	0.9753	0.9753
SingleClus	0.9235	0.9321
FF-Majority	0.9632	0.9691
FF-Minority	0.9538	0.9568
FF-Overall	0.9785	0.9815
RUS	0.8963	0.9074
RUSBoost	0.9226	0.9198
IRUS	0.9231	0.9259
SMOTE	0.9500	0.9383
Outlier-SMOTE	0.9500	0.9383

Table 4.3 Investigated Method Classifier Support vector machine (SVM) and 30 trials of 10-fold cross validation

Investigated Method	F1	TPR
Baseline	0.8859	0.8148
SingleClus	0.8867	0.8457
FF-Majority	0.8939	0.8580
FF-Minority	0.8860	0.8395
FF-Overall	0.9132	0.8765
RUS	0.8431	0.7963
RUSBoost	0.8562	0.8086
IRUS	0.8590	0.8086
SMOTE	0.8974	0.8642
Outlier-SMOTE	0.9061	0.8642

Table 4.4 Investigated Method Classifier Logistic Regression (LR) and 30 trials of 10-fold cross validation

Investigated Method	F1	TPR
Baseline	0.7568	0.6914
SingleClus	0.7645	0.6914
FF-Majority	0.8146	0.7593
FF-Minority	0.7973	0.7407
FF-Overall	0.8185	0.7654
RUS	0.7365	0.6728
RUSBoost	0.7744	0.7099
IRUS	0.7593	0.6914
SMOTE	0.7867	0.7284
Outlier-SMOTE	0.7947	0.7407

4.2.2 Experimental II

The experimental result by using Homoliak stage 2 method for the experiment will show the comparison of F1 & TPR. The dataset that was used is Legitimate & Direct attack data of 10-fold cross validation to data training by using four classifier Decision tree (C4.5), Naive Bayes (NB), Support Vector Machine (SVM), and Logistic Regression (LR) after obtaining the training data Legitimate & Direct attack result, then take it to test the Obfuscated data to obtain the result with three different objective function: FF-Majority, FF-Minority, and FF-Overall that give 30 of a data sample, that makes experiment table have 100 samples include Baseline 10 sample, SingleClus 10 sample, RUS 10 sample, RUSBoost 10 sample, IRUS 10 sample, SMOTE 10 sample, Outliner-SMOTE 10 sample.

For the overview of experimental II results of F1 & TPR and comparison of TPR and FPR. The TPR percent is increasing, especially the FF-Majority, FF-Minority, FF-Overall in Figure 4.1, and the FPR percent are decreasing around 0.10, 0.11, 0.08, of FF-Majority, FF-Minority, and FF-Overall respectively.

True Positive Rate (TPR) Result

The TPR result followed by Figure 4.3 by implementing FF-Majority, FF-Minority, and FF-Overall methods to increase the TPR achieved by using four classifier Decision tree (C4.5), Naive Bayes (NB), Support Vector Machine (SVM), and Logistic Regression (LR) for cope the undersampling. As for the result, these are satisfactory. The average data of TPR are increasing their percentage number, which means the data have a high chance of prediction, which can use in further development. Baseline 49.111%, SingleClus 51.046%, FF-Majority 52.354%, FF-Minority 52.197%, FF-Overall 53.243%, RUS 49.372%, RUSBoost 49.948%, IRUS 50.052%, SMOTE 48.954%, Outlier-SMOTE 49.477%, respectively. As a result, FF-Overall has the highest percentage of TPR, average measures across four classifiers.

Table 4.5 TPR scores as averages across from 30 trials of 10-fold cross validation, categorized by a combination of classifier and examined method. Note that corresponding values of standard deviation are given in (brackets)

Examined Method	NB	C4.5	SVM	LR
Baseline	**81.172 (4.182)**	**36.402 (2.891)**	**15.690 (3.446)**	**63.180 (3.852)**
SingleClus	**82.636 (5.376)**	**38.494 (3.784)**	**17.782 (4.103)**	**65.272 (3.448)**
FF-Majority	**83.891 (3.842)**	**39.540 (3.019)**	**19.038 (3.211)**	**66.946 (2.562)**
FF-Minority	**84.100 (2.871)**	**39.540 (2.995)**	**18.619 (3.103)**	**66.527 (2.383)**
FF-Overall	**85.146 (3.004)**	**40.377 (2.981)**	**19.665 (3.164)**	**67.782 (2.410)**
RUS	**80.753 (5.122)**	**35.983 (4.721)**	**16.946 (5.673)**	**63.808 (6.714)**
RUSBoot	**81.172 (4.219)**	**36.402 (4.016)**	**17.573 (4.862)**	**64.644 (4.673)**
IRUS	**81.381 (5.134)**	**36.611 (3.673)**	**17.782 (4.523)**	**64.435 (4.381)**
SMOTE	79.079 (4.873)	35.565 (4.822)	16.318 (3.749)	64.854 (4.006)
Outlier-SMOTE	79.498 (4.027)	36.192 (3.760)	16.946 (3.662)	65.272 (3.885)

From the following table FF-Overall that gives the largest value at 85.146% in classifier Naive Bayes (NB). FF-Overall that is the combination of FF-Majority value at 83.891% and FF-Minority value at 84.100%, both of which are inferior to FF-Overall.

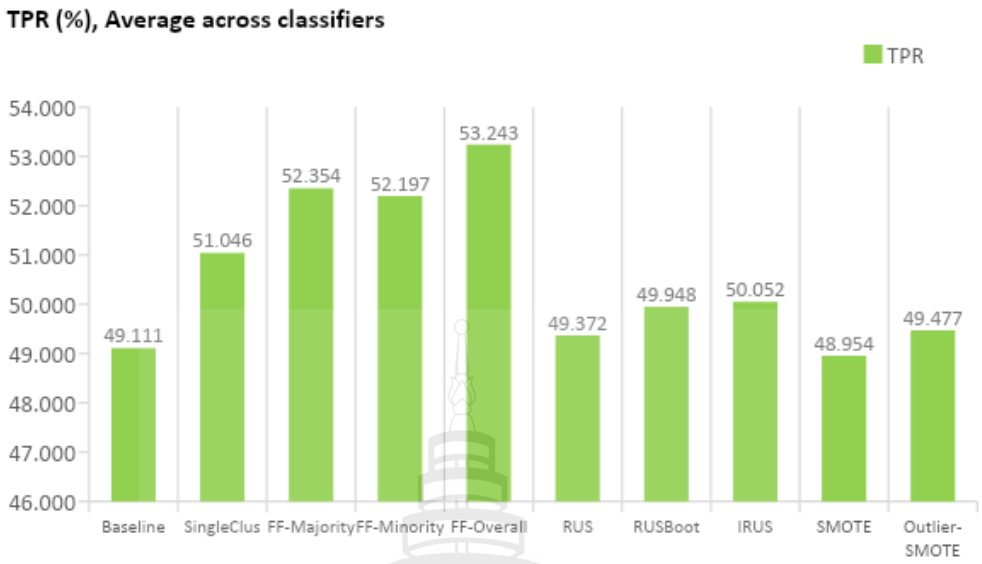


Figure 4.3 Comparison of TPR scores obtained by different methods for classifying obfuscated instances. These are averages across classification models and 30 trials of 10-fold cross validation

4.3 Discussion

For those comparing experimental I & II, the result of experimental two is satisfied, the result of TPR is increasing, which means the percentage of predictable has more percentage to be correct, and the result of FPR is decreasing, which means the percentage of predictable has less chance of failure prediction.

The technique Furthest-First used in the experiment has higher numbers predictable in three techniques: FF-Majority, FF-Minority, and FF-Overall. The purpose is to improve the handling of imbalance problems.

Table 4.6 Statistical assessment of TPR scores reported in Table 4.3 for the classification of unseen obfuscated instances. Note that, both better and worse metrics are reported for all combinations of compared methods and classification algorithms

Examined Method	NB	C4.5	SVM	LR
	better/worse	better/worse	better/worse	better/worse
Baseline	**24/45**	**38/43**	**8/70**	**13/82**
SingleClus	**50/31**	**46/37**	**41/31**	**47/38**
FF-Majority	**73/23**	**70/30**	**55/18**	**74/28**
FF-Minority	**87/20**	**69/28**	**49/23**	**67/32**
FF-Overall	**101/12**	**93/19**	**80/14**	**90/13**
RUS	**20/54**	**13/64**	**21/52**	**20/69**
RUSBoot	**26/41**	**36/42**	**36/36**	**34/52**
IRUS	**30/37**	**38/40**	**39/29**	**31/58**
SMOTE	8/96	5/69	16/58	36/49
Outlier-SMOTE	16/76	19/55	32/46	49/40

$$[\,\mu(i,t) \;=\; \frac{1}{n}\sum_{n=1}^{n} \quad TPR_n\,(i,t)\,]$$

where TPRη(i, t) denotes the TPR score obtained from the η-th fold within the t-th run of method i. The comparison of means obtained from a single trial of cross validation may be misleading, as the difference between means may not be statistically significant at times. As such, it is more reliable to make a decision based on the 95% confidence interval for the

$$[\,\mu(i,t\,) \;-1.96]\,\frac{Std(i,t)}{\sqrt{n}}\,,\mu(i,t) \;+1.96\frac{Std(i,t)}{\sqrt{n}}\,]$$

where Std(i, t) denotes the standard deviation of TPR measures across n-folds cross validation of the t-th trial, for a technique i. The statistical significance of the difference between any two methods i,i ∈ T C is found if there is no intersection

between their confidence intervals of μ(i, t) and μ(i , t). In particular, a model i is significantly better than the other model i when

$$[\,\mu(i,t\,) - 1.96]\,\frac{Std(i,t)}{\sqrt{n}} < [\,\mu(i,t\,) - 1.96]\,\frac{Std(i',t)}{\sqrt{n}}$$

Following that, the frequency that one method i ∈ T C is 748 significantly better than others across all experimented trials, 749 i.e., B(i), is calculated by the next equation.

$$B(i) = \sum_{\forall t = 1...30} \sum_{\forall i' \in TC, i' \neq i} better(i, i', t)$$

Where

$$better(i, i', t) = \begin{cases} 1 \text{ if } \left(\mu(i,t) - 1.96\frac{Std(i,t)}{\sqrt{n}}\right) > \left(\mu(i',t) + 1.96\frac{Std(i',t)}{\sqrt{n}}\right) \\ 0 \text{ otherwise} \end{cases}$$

Likewise, the frequency that one technique i ∈ T C is significantly worse than others, i.e., W(i), is estimated as follows.

$$W(i) = \sum_{\forall t=1...30} \sum_{\forall i' \in TC, i' \neq i} worse(i, i', t),$$

Across four different classifiers, the result of TPR average across classifiers Figure 4.4(A) is improved in particular to FF-Overall will have the most outstanding value in the range 250 of M, which is approximately 54.200% of the maximum measured value. In other words, adding more clustering results to the pool will not yield any further significant improvement. Similarly, with two models, FF-Majority and FF-Minority, where FF-Majority is not much different from each FF-Minority value. Also, the same comparison of TPR NB classifier Figure 4.4(B) is improved in particular to FF-Overall will have the most excellent value in the range 250 of M, which is

approximately 86.673% of the maximum measured value. Like both models, FF-Majority and FF-Minority, where FF-Majority is not much different from each FF-Minority value.

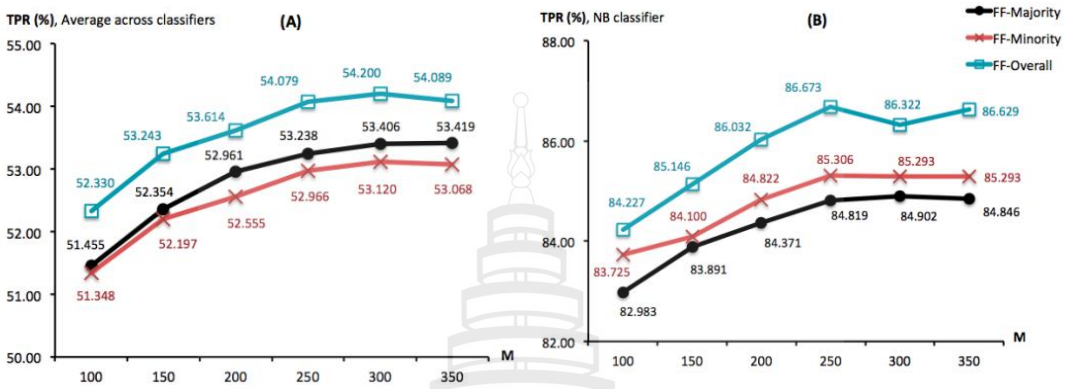


Figure 4.4 TPR scores obtained by FF-Majority, FF-Minority and FF-Overall, with different ensemble sizes of M ∈{100, 150, 200, 250, 300, 350}. These are summarized from 30 trials of 10-fold cross validation: (A) across four classifiers and (B) specific to NB

Another necessary sample to be studied and discussed is the size ρ of reduced majority class X_0. By default, it is set the same as the value of the minority X_1 counterpart. However, with the process of 10-fold cross-validation of FF-Majority, FF-Minority, and FF-Overall, it is automatically configured to be around 146 samples that will be referred to as Q hereafter. Therefore, the subsequent investigation is to discover the relation between the overall result and classifier-specific TPR scores and different values of Q: Q, 2Q, and 3Q that correspond to target numbers of representative samples of the majority class are 146, 292, and 438, respectively. Based on the experimental result of classifying obfuscated connections data by using M = 250, the result summarized 10-fold cross-validation of FF-Majority, FF-Minority, and FF-Overall from the four classifiers are shown in Figure 4.5(A), in the case of 2Q, it has the most accurate more than Q and 3Q classification model. For instance, the TPR score of FF-Overall tends to increase with the size number of the reduced majority class information. In that case, this suggests a loss of majority class information encountered

while selecting a small number of representative samples. However, having too many of these values according to 3Q will result in poor effective results because there will be an imbalance of re-visit. Observed from Figure 4.5(B) specific result of NB. In particular, FF-Overall, a higher TPR is 87.246% with 2Q, compared to 86.673% with the initial Q setting.

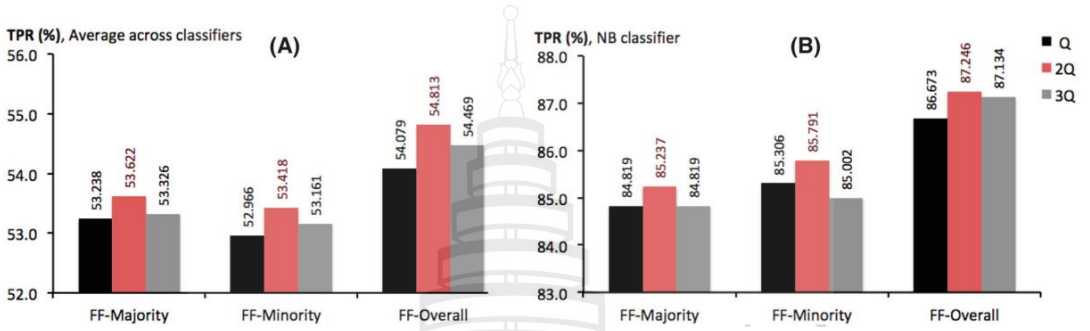


Figure 4.5 TPR scores obtained by FF-Majority, FF-Minority and FF-Overall, with different sizes of the majority class, Q, 2Q and 3Q. These are summarized from 30 trials of 10-fold cross validation: (A) across four classifiers and (B) specific to NB

The experiment used for improved handling of imbalance problems will be the experimental II by Homoliak stage 2 method.

The following points must be clarified regarding the implications of the proposed methods for severe class imbalance, in which the size of the minority class gets extremely small. For this experiment, all training data collections, i.e., the size of the majority and minority classes of 10.805 and 162, are used to define those hiding obfuscated classes. In observing that, the experiment repeated from 30 trials with has a smaller class of $|X_1| \in \{162, 122, 81, 41\}$ more than the prior experiment, while M is 250 and the target size of the majority class is 2Q, i.e., twice as large as the other class. According to Figure 4.6 (A), which shows TPR scores obtained by the NB classifier, the tendency of predictive performance constantly declines as $|X_1|$ drops from 162 to 41. This is regularly observed with contrasting methods examined here, with FF-Majority, FF-Minority, and FF-Overall providing the best measures and outperforming Baseline and SingleClus. It was also observed with the C4.5 technique, which results

in Figure 4.6 (B). Therefore, it would be possible that inferring the proposed framework can be an effective alternative to developing a classification model for a high imbalance issue discovered in various real-world domains, e.g., fraud detection and cancer diagnosis.

In addition, FF-Minority implementations tend to be less efficient as $|X_1|$ decreases, which is sensible given that the underlying objective function makes use of distances from a centroid of interest to members of X_1. As a result, the performance of FF-Overall is affected by this estimate, where the TPR score is lower than the FF-Majority score as $|X_1|$ getting to the smallest size of 41. For such a case, FF-Majority is preferred over the others, but it should be noted that the accuracy of the resulting model will still be vastly lower than that achieved with a bigger $|X_1|$. To this end, collecting additional samples of the minority classes may be better.

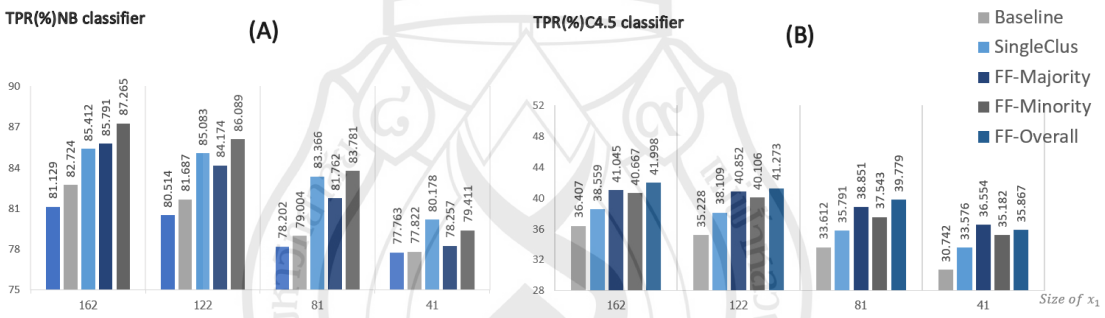


Figure 4.6 TPR scores obtained by FF-Majority and FF-Overall,with different sizes of the minority class $X_1 \in \{162, 122, 81, 41\}$. These are summarized from 30 trials for two specific classifiers: (**A)** specific to NB and (**B)** specific to C4.5

CHAPTER 5

DISCUSSION AND CONCLUSION

5.1 Discussion and Conclusion

This research has presented a new approach to handling imbalance problems for machine-learning-based NIDS with opponent attacks, where those legitimate data, direct attacks, and obfuscated intrusions are more accurately classified. To complement an initial attempt to seek a feature subset optimal for this classification task, the current work focuses on solving the difficulty, as mentioned earlier, by extending the concept of clustering-based undersampling. Instead of referring to a single data partition, select a target set of samples that are representative of a group of multiple grouping clustering results, which has been created by using the concept of the ensemble clustering algorithm. This ensures a variety of data partitions, so the resulting centroid, which some can choose the original set to represent the majority class. Three variations of the proposed framework are recommended depending on various objective functions used for this iterative and greedy process: FF-Majority, FF-Minority, and FF-Overall.In particular, the first looks for a centroid that is mainly different from others within the pool, while the second prefers the one that is dissimilar to samples of the minority class. As suggested by its name, the last integrates these two to achieve an overall evaluation of candidates. Based on the evaluation of different classification algorithms and the published dataset in which some direct attacks have been altered using several obfuscation techniques, those new methods are more advantageous than the baseline (i.e., using a subset of selected properties without having to deal with class imbalance problem) and the single-clustering based undersampling. This is also observed compared to another benchmark undersampling/oversampling method and a recent extension. The proposed framework is generic so that it can be coupled with any standard classification model.

In the same way, this applies to the exploitation of other objective functions to distinguish the centroid good as a representative of the majority class. In addition, parameter analysis and implications are included in this research as a guideline for their applications.

5.2 Suggestion and Future work

Despite this success, there are several works to be further conducted. Firstly, it is to evaluate these new techniques on more datasets in such a way as to draw a reliable conclusion for practical uses. In addition to selecting representative samples, other objective functions and optimization methods may be largely explored. Besides, other aspects of the underlying feature space may well be addressed, for instance, the treatment of missing values [68] and discretization of a feature domain [69].

Finally, introducing fuzzy sets and vocabularies can support the explainability of the prediction process [70]. Hence, the final remark it is something that interests a lot to provide the emerging trend of explainable AI for modern applications.

REFERENCES

REFERENCES

[1] Agarwal, N., & Hussain, S. Z. (2018). A closer look at intrusion detection system for web applications. *Secur Commun Netw*, Article ID 9601357. https://doi.org/10.1155/2018/9601357

[2] Anthi, E., Williams, L., Rhode, M., Burnap, P., & Wedgbury, A. (2021). Adversarial attacks on machine learning cybersecurity defences in industrial control systems. *J Inf Secur Appl., 58*(102717), 1–9. https://doi.org/10.1016/j.jisa.2020.102717

[3] Farhan, B. I., & Jasim, A. D. (2022). A survey of intrusion detection using deep learning in internet of things. *Iraqi Journal For Computer Science and Mathematics, 3*(1), 83–93.

[4] Kumar, D. P., Amgoth, T., & Annavarapu, C. S. R. (2019). Machine learning algorithms for wireless sensor networks: A survey. *Information Fusion, 49*, 1–25.

[5] Jia, Y., Qi, Y., Shang, H., Jiang, R., & Li, A. (2018). A practical approach to constructing a knowledge graph for cybersecurity. *Engineering, 4*(1), 53–60.

[6] Kravchik, M., & Shabtai, A. (2018). Detecting cyber attacks in industrial control systems using convolutional neural networks. In *ACM International Workshop on Cyber-Physical Systems Security and Privacy* (pp. 72–83). ACM. https://doi.org/10.1145/3264888.3264896

[7] Alcaraz, C. (2018). Cloud-assisted dynamic resilience for cyberphysical control systems. *IEEE Wirel Commun, 25*(1), 76–82.

[8] Gao, L., Shen, W., & Li, X. (2019). New trends in intelligent manufacturing. *Engineering, 5*(4), 11–20.

[9] Tarter, A. (2017). Importance of cyber security. In P. S. Bayerl, R. Karlović, B. Akhgar & G. Markarian (Eds.), *Community policing-A European perspective: Strategies, best practices and guidelines* (pp. 213–230). Cham, Switzerland: Springer.

[10] Wang, D., Wang, X., Zhang, Y., & Jin, L. (2019). Detection of power grid disturbances and cyber-attacks based on machine learning. *Journal of Information Security and Applications, 46*, 42–52. https://doi.org/10.1016/j.jisa.2019.02.008

[11] Ashibani,Y., & Mahmoud, Q. H. (2017). Cyber physical systems security: Analysis, challenges and solutions. *Computer Security, 8*, 81–97.

[12] Kumar, D. P., Amgoth,T., Annavarapu, C. S. R. (2021). CANintelliIDS: Detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU. *IEEE Transactions on Network Science and Engineering, 8*(2), 1456–1466.

[13] Li, J., Qu, Y., Chao, F., Shum, H., Ho, E., & Yang, L. (2018). Machine learning algorithms for network intrusion detection. In *AI in Cybersecurity. Intelligent systems reference library* (vol. 151, pp. 151-179). Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-319-98842-9_6

[14] Barreno, M., Nelson, B., Joseph, A., & Tygar, J. (2010). The security of machine learning. *Mach Learn, 81*(2), 121–148.

[15] Sethi, T. S., & Kantardzic, M. (2018). When good machine learning leads to bad security. *Ubiquity, 2018*(May), 1–14. https://doi.org/10.1145/3158346

[16] Najafabadi, M., Villanustre, F., Khoshgoftaar, T., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data, 2*(1), 1-21. https://doi.org/10.1186/s40537-014-0007-7

[17] Prasad, R., & Rohokale, V. (2020). Artificial intelligence and machine learning in cyber security. In *Cyber security: The lifeline of information and communication technology* (pp. 231–247). Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-030-31703-4

[18] Karatas, G., Demir, O., & Sahingoz, O. (2020). Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset. *IEEE Access, 8,* 32150–32162.

[19] Yan, B., & Han, G. (2018). Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. *IEEE Access, 6*, 41238–41248.

[20] Yao, H., Fu, D., Zhang, P., Li, M., & Liu, Y. (2018). MSML: A novel multilevel semi-supervised machine learning framework for intrusion detection system. *IEEE Internet of Things Journal, 6*(2), 1949–1959.

[21] Shen, Y., Zheng, K., Wu, C., Zhang, M., Niu, X., & Yang, Y. (2018). An ensemble method based on selection using bat algorithm for intrusion detection. *The Computer Journal, 61*(4), 526–538.

[22] Gao, X., Shan, C., Hu, C., Niu, Z., & Liu, Z. (2019). An adaptive ensemble machine learning model for intrusion detection. *IEEE Access, 7*, 82512–82521.

[23] Homoliak, I., Teknös, M., Ochoa, M., Breitenbacher, D., Hosseini, S., & Hanáček, P. (2018). Improving network intrusion detection classifiers by nonpayload-based exploit-independent obfuscations: An adversarial approach. *EAI Endorsed Transactions on Security and Safety, 5*(17), 1-15.

[24] Dka, C., Papa, J., Lisboa, C., Munoz, R., Dvhc, A. (2019). Internet of things: A survey on machine learning-based intrusion detection approaches. *Comput Networks, 151*, 147–157.

[25] Teixeira, M. A., Salman, T., Zolanvari, M., Jain, R., Meskin, N., & Samaka, M. (2018). SCADA system testbed for cybersecurity research using machine learning approach. *Future Internet, 10*(8), 76.

[26] Aljanabi, M., Ismail, M. A., & Ali, A. H. (2021). Intrusion detection systems, issues, challenges, and needs. *Int J Comput Intell Syst, 14*(1), 560–571.

[27] Fossaceca, J. M., Mazzuchi, T. A., & Sarkani, S. (2015). MARK-ELM: Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection. *Expert Syst Appl, 42*, 4062–4080.

[28] Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (20020). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

[29] Lin, C.-T., Hsieh, T.-Y., Liu, Y.-T., Lin, Y.-Y., Fang, C.-N., Wang, Y.-K., . . . Chuang, C.-H. (2018). Minority oversampling in kernel adaptive subspaces for class imbalanced datasets. *IEEE Transactions on Knowledge and Data Engineering, 30*, 950–962.

[30] Blaszczynski, J., & Stefanowski, J. (2015). Neighborhood sampling in bagging for imbalanced data. *Neurocomputing, 150*, 529–542.

[31] Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence, 5*, 231–232.

[32] Lin, W. C., Tsai, C. F., Hu, Y. H., & Jhang, J. S. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences, 409-410*, 17–26.

[33] Sun, Z., Song, Q., Zhu, X., Sun, H., Xu, B., & Zhou, Y. (2015). A novel ensemble method for classifying imbalanced data. *Pattern Recogn, 48*, 1623–1637

[34] Homoliak, I., Ovsonka, D., Gregr, M., & Hanáček, P. (2014). NBA of obfuscated network vulnerabilities exploitation hidden into HTTPS traffic. In *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)* (pp. 310–317). London: IEEE. https://doi.org/doi: 10.1109/ICITST.2014.7038827

[35] Corona, I., Giacinto, G., & Roli, F. (2013). Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Inf Sci, 239*, 201–225.

[36] Seiffert, C., Khoshgoftaar, T., Hulse, J. V., & Napolitano, A. (2010). Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on System, Man and Cybernertics, Part A, 40*, 185–197.

[37] Tahir, M., Kittler, J., & Yan, F. (2012). Inverse random under sampling for class imbalance problem and its application to multi-label classification. *Pattern Recognition, 45*, 3738–3750.

[38] Iam-On, N. (2020). Clustering data with the presence of attribute noise: A study of noise completely at random and ensemble of multiple k-means clusterings. *International Journal of Machine Learning and Cybernetics, 11*, 491–509.

[39] Panwong, P., Boongoen, T., & Iam-On, N. (2020). Improving consensus clustering with noise-induced ensemble generation. *Expert Systems with Applications, 146*, 113–138.

[40] Iam-On, N., & Boongoen, T. (2017). Improved student dropout prediction in thai university using ensemble of mixed-type data clusterings. *Int J Mach Learn Cybern, 8*(2), 497–510.

[41] Turlapati, V. P. K., & Prusty, M. R. (2020). Outlier-SMOTE: A refined oversampling technique for improved detection of COVID-19. *Intelligence-Based Medicine, 3-4*, 100023.

[42] Heady, R., Luger, G., Maccabe, A., & Servilla, M. (1990). *The architecture of a network level intrusion detection system*. Computer Science Department, University of New Mexico. https://doi.org/10.2172/425295

[43] Ghilardi, S., Gianola, A., Montali, M., & Rivkin, A. (2011). Petri net-based object-centric processes with read-only data. *Information Systems, 107*. https://doi.org/10.1016/j.is.2022.102011

[44] Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2011). Surveying port scans and their detection methodologies. *The Computer Journal, 54*(10), 1565–1581.

[45] Bul'ajoul, W., James, A., & Pannu, M. (2015). Improving network intrusion detection system performance through quality of service configuration and parallel technology. *Journal of Computer and System Sciences, 81*(6), 981–999.

[46] Abdulhammed, R., Faezipour, M., & Elleithy, K. M. (2016). Network intrusion detection using hardware techniques: A review. In *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)* (pp. 1-7). https://doi.org/10.1109/LISAT.2016.7494100

[47] Patel, J., & Panchal, K. (2015). Effective intrusion detection system using data mining technique. *Journal of Emerging Technologies and Innovative Research, 2*(6), 1869-1878.

[48] Khamphakdee, N., Benjamas, N., & Saiyod, S. (2015). Improving intrusion detection system based on snort rules for network probe attacks detection with association rules technique of data mining. *Journal of ICT Research and Applications, 8*(3), 234-250.

[49] Ravale, U., Marathe, N., & Padiya, P. (2015). Feature selection based hybrid anomaly intrusion detection system using K means and RBF kernel function. *Procedia Computer Science, 45*, 428-435. https://doi.org/10.1016/j.procs.2015.03.174

[50] Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks, 16*(3), 645-678.

[51] Dey, A. (2016). Machine learning algorithms: A review. *International Journal of Computer Science and Information Technologies, 7*(3), 1174-1179.

[52] Qiu, S. (2018). *The kaust repository: Support vector machine and application in seizure prediction* (Master's thesis). King Abdullah University of Science and Technology. https://doi.org/10.25781/KAUST-V882P

[53] Sharma, H., & Kumar, S. (2016). A survey on decision tree algorithms of classification in data mining. *International Journal of Science and Research, 5*(4), 2094-2097.

[54] Rokach, L., & Maimon, O. (2005). *Decision trees, data mining and knowledge discovery handbook*. Springer.

[55] Karthika, S., & Sairam, N. (2015). A naïve bayesian classifier for educational qualification. *Indian Journal of Science and Technology, 8*(16), 1-5.

[56] Edgar, T. W., & Manz, D. O. (2017). *Research methods for cyber security*. Syngress.

[57] Liu, T. (2020). *Improved K-means clustering algorithms*. Master's thesis. Massey University.

[58] Jiang, K., Lu, J., & Xia, K. (2016). A novel algorithm for imbalance data classification based on genetic algorithm improved SMOTE. *Arab J Sci Eng, 41*, 3255–3266.

[59] Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems With Applications, 73*, 220-239.

[60] Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(8), 832-844.

[61] Rubin, S., Jha, S., & Miller, B. (2004). Automatic generation and analysis of NIDS attacks. In *20th Annual Computer Security Applications Conference* (pp. 28–38). Tucson, AZ: IEEE.

[62] Homoliak, I., Ovšonka, D., Koranda, K., & Hanáček, P. (2014). Characteristics of buffer overflow attacks tunneled in HTTP traffic. In *2014 International Carnahan Conference on Security Technology (ICCST)* (pp. 188–193). Rome, Italy: IEEE.

[63] Homoliak, I., Barabas, M., Chmelar, P., Drozd, M., & Hanáček, P. (2013, July 22-25). *ASNM: Advanced security network metrics for attack vector description* [Paper presentation]. The 2013 International Conference on Security and Management (SAM'13) (pp. 350–358). Las Vegas, Nevada, United States.

[64] Homoliak, I., Malinka, K., & Hanáček, P. (2020). ASNM datasets: A collection of network attacks for testing of adversarial classifiers and intrusion detectors. *IEEE Access, 8*, 112427– 112453.

[65] Iam-On, N., & Boongoen, T. (2015). Diversity-driven generation of link-based cluster ensemble and application to data classification. *Expert Systems with Applications, 42*, 8259–8273.

[66] Iam-On, N., Boongoen, T., Garrett, S., & Price, C. (2011). A link-based approach to the cluster ensemble problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 33*, 2396–2409.

[67] Nanni, L., Fantozzi, C., & Lazzarini, N. (2015). Coupling different methods for overcoming the class imbalance problem. *Neurocomputing, 158*, 48–61.

[68] Pattanodom, M., Iam-On, N., & Boongoen, T. (2016). Hybrid imputation framework for data clustering using ensemble method. In *Proceedings of the 20th Pacific Asia Conference on Information Systems (PACIS 2016)* (pp. 86–91). Chiayi, Taiwan: IEEE.

[69] Sriwanna, K., Boongoen, T., & Iam-On, N. (2017). Graph clustering-based discretization of splitting and merging methods (graphs and graphm). *Human-centric Computing and Information Sciences*, *7*, 1–39.

[70] Fu, X., Boongoen, T., & Shen, Q. (2010). Evidence directed generation of plausible crime scenarios with identity resolution. *495 Applied Artificial Intelligence, 24*, 253–276.