Imbalanced data classification and its application in cyber security

Md Moniruzzaman

This thesis is submitted in total fulfilment of the requirement
for the degree of Doctor of Philosophy

School of Engineering, Information Technology and Physical Sciences
Federation University Australia
PO Box 663
University Drive, Mount Helen
Ballarat, Victoria, Australia 3353

Submitted in September 2020

Abstract

Cyber security, also known as information technology security or simply as information security, aims to protect government organizations, companies and individuals by defending their computers, servers, electronic systems, networks, and data from malicious attacks. With the advancement of client-side on the fly web content generation techniques, it becomes easier for attackers to modify the content of a website dynamically and gain access to valuable information. The impact of cybercrime to the global economy is now more than ever, and it is growing day by day. Among various types of cybercrimes, financial attacks are widely spread and the financial sector is among most targeted. Both corporations and individuals are losing a huge amount of money each year. The majority portion of financial attacks is carried out by banking malware and web-based attacks. The end users are not always skilled enough to differentiate between injected content and actual contents of a webpage. Designing a real-time security system for ensuring a safe browsing experience is a challenging task. Some of the existing solutions are designed for client side and all the users have to install it in their system, which is very difficult to implement. In addition, various platforms and tools are used by organizations and individuals, therefore, different solutions are needed to be designed.

The existing server-side solution often focuses on sanitizing and filtering the inputs. It will fail to detect obfuscated and hidden scripts. This is a real-time security system and any significant delay will hamper user experience. Therefore, finding the most optimized and efficient solution is very important. To ensure an easy installation and integration capabilities of any solution with the existing system is also a critical factor to consider. If the solution is efficient but difficult to integrate, then it may not be a feasible solution for practical use.

Unsupervised and supervised data classification techniques have been widely

applied to design algorithms for solving cyber security problems. The performance of these algorithms varies depending on types of cyber security problems and size of datasets. To date, existing algorithms do not achieve high accuracy in detecting malware activities.

Datasets in cyber security and, especially those from financial sectors, are predominantly imbalanced datasets as the number of malware activities is significantly less than the number of normal activities. This means that classifiers for imbalanced datasets can be used to develop supervised data classification algorithms to detect malware activities.

Development of classifiers for imbalanced data sets has been subject of research over the last decade. Most of these classifiers are based on oversampling and undersampling techniques and are not efficient in many situations as such techniques are applied globally. In this thesis, we develop two new algorithms for solving supervised data classification problems in imbalanced datasets and then apply them to solve malware detection problems.

The first algorithm is designed using the piecewise linear classifiers by formulating this problem as an optimization problem and by applying the penalty function method. More specifically, we add more penalty to the objective function for misclassified points from minority classes. The second method is based on the combination of the supervised and unsupervised (clustering) algorithms. Such an approach allows one to identify areas in the input space where minority classes are located and to apply local oversampling or undersampling. This approach leads to the design of more efficient and accurate classifiers.

The proposed algorithms are tested using real-world datasets. Results clearly demonstrate superiority of newly introduced algorithms. Then we apply these algorithms to design classifiers to detect malwares.

DECLARATION

I, Md Moniruzzaman, declare that the PhD thesis entitled "Imbalanced data classification and its application in cyber security" is no more than 35,000 words in length including quotes and exclusive of tables, figures, appendices, bibliography, references and footnotes. This thesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma. Except where otherwise indicated, this thesis is my own work.

I give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of a Research Priority Area Scholarship.

Signature.....

September 2020

Md Moniruzzaman

Dedicated

To

My parents and my family

Table of Contents

List of Tables

List of Figures

List of Publications

1. Md Moniruzzaman, Adil Bagirov, and Iqbal Gondal. Partial undersampling of imbalanced data for cyber threats detection. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '20, New York, NY, USA, 2020. Association for Computing Machinery.

2. Md Moniruzzaman, Adil Bagirov, Iqbal Gondal, and Simon Brown. A server side solution for detecting webinject: A machine learning approach. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 162–167. Springer, 2018.

Chapter 1

Introduction

Cybercrime is a crime that involves a computer and a network. This type of crime may threaten, in particular, individuals, countries security, financial sector, banks and healthcare systems. Cyber security, also known as information technology security, aims to protect organizations and individuals by defending their computers, electronic systems, networks, and data from malicious attacks. Among cybercrimes, financial attacks are widely spread and the majority of these attacks is carried out by banking malware.

In this chapter, we present an introduction to various types of malware and their impacts on individuals and organizations. We discuss the working principles of some common malware and present our research objective on designing an efficient solution against cyber threats. Finally, we will provide a short description of the organization of the thesis.

1.1 What is malware?

The word malware comes from the words ”malicious software”. Malware contains code that causes damage to a target, access personal and valuable information or gain unauthorized access to the system. There are generic malware that can attack any individuals or organization and there are custom designed malware to attack specific target. Malware nowadays are used to gain financial benefits by directly stealing money or by gaining access to banking credentials. Malware are used against government or organization to affect their operation

and sometimes it gains access to sensitive information and demands money in exchange for it.

In early days, the main focus of designing malware was not to do harm. People designed malware as an experiment or pranks and sometimes to find a loophole in a system. The first malware for personal computer (PC) was a virus, known as Brain.A developed by two Pakistani brothers named Basit and Amjad [1]. This harmless malware was designed to show the vulnerability of PC. It simply infects boot sector of a floppy drive and spread it to other floppy drive connected to the infected PC.

Recent malware are more sophisticated and can destroy file-system or operating system of infected devices. Although these type of malware may not cause direct financial damage but it causes a problem for users by damaging data and hardware as well. Valuable resources and efforts are needed to recover from this kind of damage. Instead of individuals, if an organization becomes the victim, the impact of damage becomes significantly large.

1.2 Types of malware

There are various types of malware. They are categorized based on their behaviour, working mechanism, self-execution and self-distribution capabilities. The major types of malware are listed below:

- **Adware:** Adware is the most common and least harmful malware. It automatically delivers an advertisement to the infected computers. Sometimes users may agree on adware while using some free software. In most of the cases, it is a pop-up and does not do any real damage, but it may contain links to a malicious websites and download malicious software which may result into greater damage for infected devices.

- **Spyware:** This type of malware collects personalized information, such as network traffic, keystrokes, surfing behaviour silently from the infected machine. One can get spyware in their system in various ways, such as: drive-by download, piggybacked software, browser add-ons etc.

Sometimes it can be disguised as anti-spyware software and trick people to download it. Often Adware and Spyware are combined to show customized advertisement.

- **Virus:** A virus is a malicious code which attaches itself to a host program and executes automatically when users execute or launches those infected programs. Every virus has two basic capabilities. It can create a copy of itself automatically and it can self-execute. Virus is unable to spread automatically, human assistance is needed to transfer the infected host file from one machine to another. Virus can harm infected device by deleting files, damaging operating system and other programs, flooding network with unnecessary traffic and hamper the performance.

- **Worm:** Worm is a malicious program which has self-reproduction and self-execution capabilities like virus but it can spread without human assistance. It can spread within infected machine and sometimes it can infect the whole network. Worms often consume a lot of resources from infected machine and slows down their performance. Sometimes worms contain a payload, which is a piece of code programmed to perform specific malicious activity including stealing data, deleting files, creating botnets etc.

- **Trojan:** Trojan is a malicious program disguised as a legitimate software. Trojan can spread over email and also it can spread when someone install software from an unreliable source. Trojan can be used to gain access to infected computers and allows attacker to steal data and other information. It can destroy files and install other malwares as well. Trojan containing a worm as payload can cause huge damage to all devices in an infected network.

- **Bot:** Bot, in general, cannot be considered as malware. The bot is a program that automates some process, that usually requires human involvement. The bot can be used for many good purposes (i.e software testing, online help service etc). Cyber criminals use a bot to infect a group of computers and form a ”botnet”. A botnet is connected to Command and Control (C&C) centre. Attackers give instruction to bot

from C&C and can perform a large-scale attack such as DDoS, scraping server data, creating spambot to distribute spam emails etc.

- **Ransomware:** Ransomware is a malware that uses encryption to lock down files or disable access to the system and demands ransom to remove the restriction. In recent time, the attacker demands the payment via cryptocurrency to remain anonymous. Sometimes various gift cards (i.e. iTunes, Amagon, Google play) are used as payment as well. Malicous spam (malspam) is one of the most common ways of spreading ransomware. The email body may contain infected pdf or word document or a link to a malicious website. Once infected by ransomware, users will be locked out of their system and will be provided with a notification about being infected and instructions on payment of demanding ransom.

1.3 How does malware spread?

Cybercriminals use various technique to spread malware onto victim's computer. Each attacker devise their own way to deliver the malware to user's device. There are various ways a device can get infected. Some of the popular methods of spreading malware are:

- **Social Engineering:** Social engineering use psychological manipulation to lure users to give up confidential information or download malicious content embedded with malware. Attackers try to find the interest of the certain user and attract them with similar contents that they might get interested in. Several techniques are used to trick a user, such as: asking for urgent help, declaring you as a winner, responding to a help request that you never asked, posing as other person etc. Sometimes they use fake social media profiles to fool the users as well. Self awareness is the key to be safe from social engineering attack. No security mechanism works if the users make mistakes by themselves. We always have to verify the email address and be cautious about opening email from unknown source.

- **Phishing:** Phishing is one of the oldest tricks to spread malware. Users

often receive emails that seem to be coming from a legitimate source. Even for the technical users sometimes it becomes difficult to distinguish between a real and a phishing email. People with less knowledge and little experience about technology get easily fooled by those emails.These people follow the instruction received in phishing email and easily end up getting infected. Common features of phishing emails are: too good to be true, contains attachment or hyperlinks with misspelled websites, requires urgent action, out of context email from unusual sender. Spear phishing is a phishing technique combined with social engineering to attack specific target. It is very difficult to detect even for the experienced user, as the part of the email contains true information which makes it look like a valid email from a trusted source.

- **Drive-by downloads:** A drive-by-download is an involuntary download of a virus or malware to someones computing device. By this method victim's computer can get infected in two different ways, authorized and fully unauthorized download. In case of authorized download, user knowingly downloads and install an application which claims to be some useful tools, but they contain the malicious code inside and infect the devices without their knowledge. In case of unauthorized download device get infected without any action needed from the user. This type of attacks exploit the vulnerabilities and take advantage of out of date application or operating system. These downloads may be placed on a hijacked website or fake websites pretending to be some legitimate source.

- **Malvertising:** Malvertising is malicious advertisement for transmitting malware and for breaching networks. Unwanted malicious codes are attached to the advertisement for creating the advertising content. JavaScript is often used to create malvertisement. Attackers pay legitimate advertising network to display their ads. Either intentionally or unintentionally users click those advertisement get infected. Malvertisement have a significant difference to addware which is another form of spreading malicious advertisement. Adware is deployed on victim's computer and mavertising uses a publisher's website and their malicious content is deployed there.

- **Using infected hardware:** Self-executing malware like viruses and worms can spread via portable memory devices or networks. One of the most common ways of getting infected is using infected USB memory sticks while transferring files from one device to another. Worms can spread over the network and infects other computers in the same network. One should always scan a portable device before using it and be cautious about plugging it into other devices.

1.4 Financial damage by malware

Although the journey of malware began without the intention of doing any real damage to target devices, soon after realizing the possibilities of gaining financial benefits from malware, attackers started to design malware that causes direct financial damage. Sometimes malware was used to steal money from someone's account, sometimes they gained control over sensitive data and demand ransomware in exchange for getting those data back. In 2013, 6.2% of overall malware were considered to be financial malware [2]. Still, this small percentage causes big financial damage. From the year 2007 to 2010 only the Zeus malware stole more than 100 million US dollars from different bank accounts [3].

Among various types of financial malware, almost 70% are banking malware [2]. The global cost for cybercrime reached close to $600 billion dollar in 2017 [5]. This cost is calculated considering direct financial damage, wasted time, recovering lost data, lost productivity etc. The amount will reach up to $6 trillion dollar by 2021 [6]. Figure 1.1 shows the growth of financial cost for cybercrime in five years between 2013 and 2017. This graph clear demonstrated that the growth in last two years 2016 and 2017 is more than that of for the first three years.

Figure 1.2 shows the annual cost due to cyber attacks on top targeted industries. We can see that financial services are the prime sector as a target for the attackers. Utility and energy services also suffer from a huge damage from the cyber attacks. Sometimes one country may perform secret cyber attacks on other countries important facilities and installations to get secret

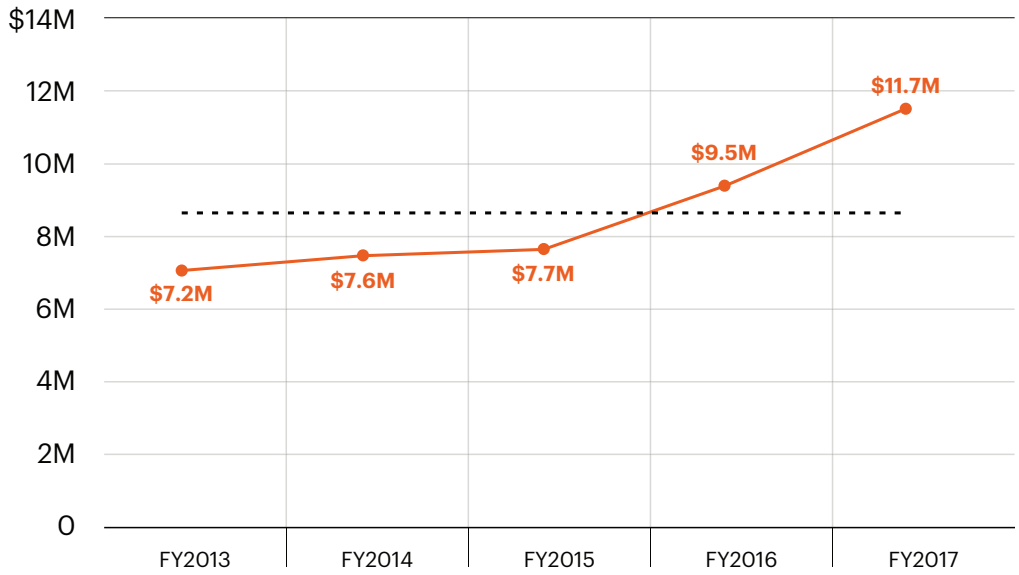

Figure 1.1: Growth in cybercrime cost in last five years
Source: [4]

information. Aerospace and defense systems are among such facilities. This is targeted by terrorist groups as well. Nowadays almost every sector uses technology to ease their functionality. Transport, healthcare, communication, life science are some examples where online services and digitization is introduces. As a result, they are also becoming a target for cybercriminals. Figure 1.2 shows the annual cost due to a cyber attack on top targetted industry.

Among various types of attacks, malware and web-based attacks are the most used one. With the increasing number of internet based services the online activity is increased by a lot in recent times. A large proportion of online service users are unaware of cybercrime activities. They are getting tricked in various ways and becoming a victim of the cybercriminals.

Figure 1.3 shows the annual cost percentage categorized by different attack types among 254 organization. Organizations are divided into two categories. Light orange color represents the larger organization having seats of above median seats and dark color represents smaller organization having below median seats. We can observe that malware and web-based attacks causes highest damage for both small and big organizations. Bigger organizations are vul-

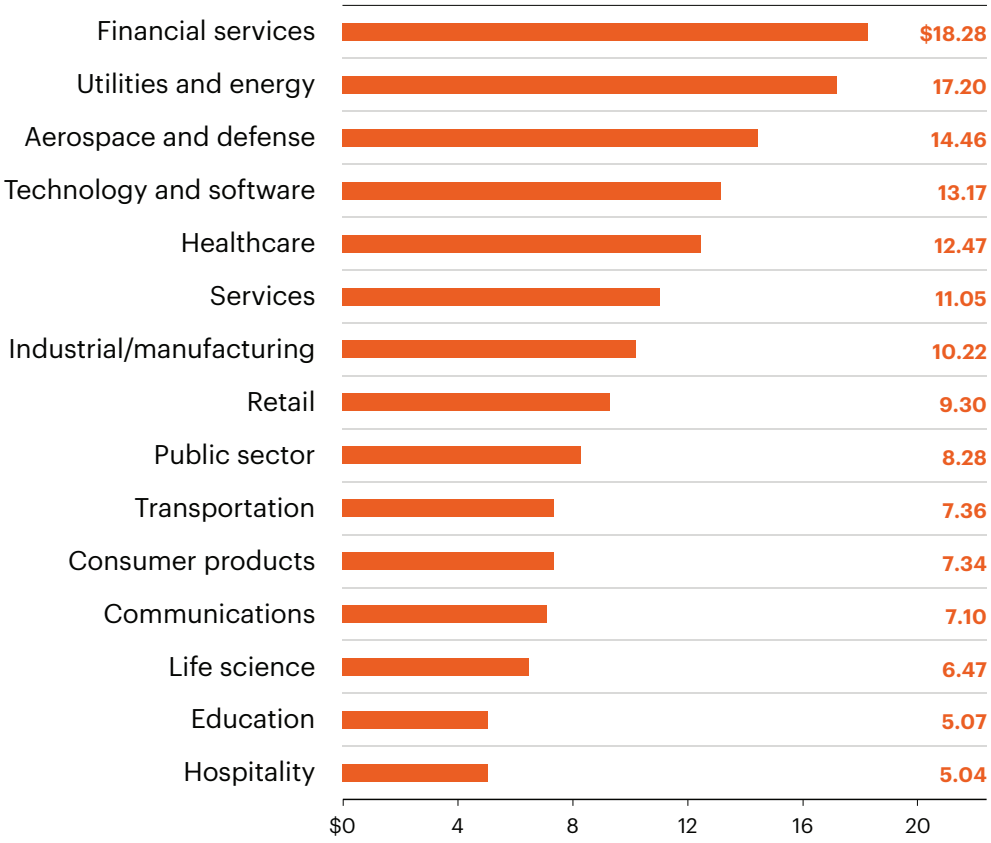

Figure 1.2: Average annual cost of different types of organization
Source: [4]

nerable to denail of services (DDOS) attack, whereas small organization are vulnerable to phishing and social engineering.

1.5 How banking malware works

Banking malware often steals the confidential information from the user and sends it to the attackers. The information harvesting malware started working in practice during mid-2014 [7]. They start working when the user has an active internet connection and visits the certain website that matches the filter of the malware. They use different techniques for collecting information, such as form grabbing, keylogging, screenshot, video capture etc. Using screengrabber they also collect information even when the user uses the virtual on-screen

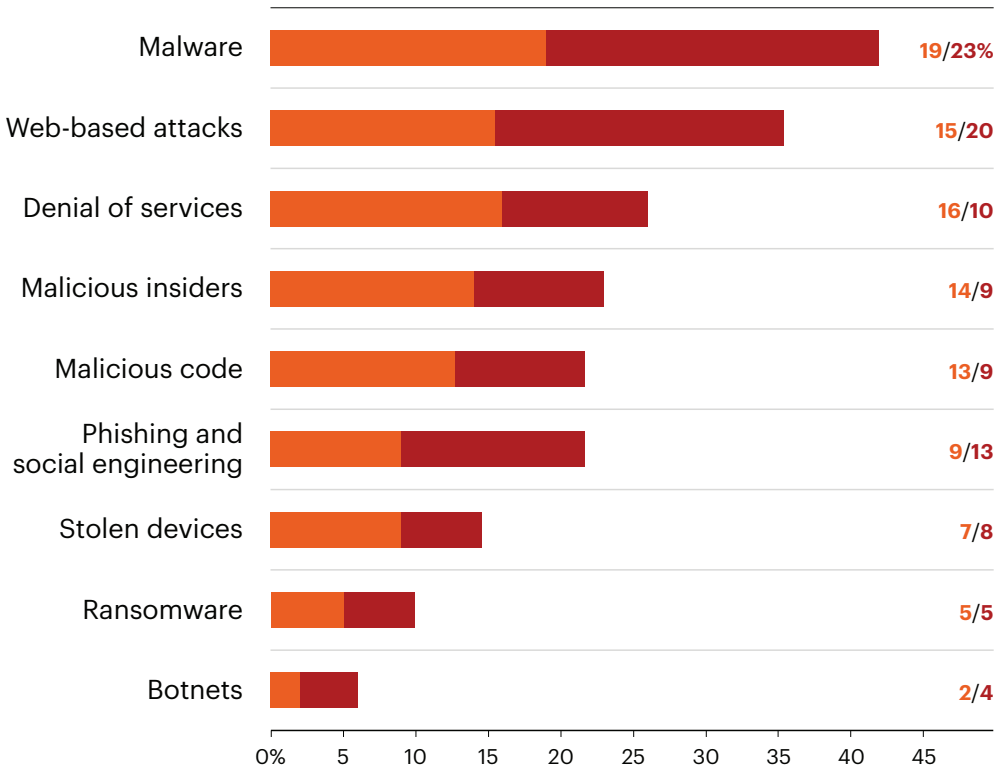

Figure 1.3: Cost for various types of attack
Source: [4]

keyboard. Sometimes user saves their important information in some files into their computer. Some malware looks into the file system and search for the keywords and collects information from the matching files. The collected information is then sent to attackers and this information is used to transfer money from victim's account.

JavaScript has become one of the most effective and common tools use by the attackers due to its flexibility and dynamic characteristics [8]. Banking malware is often custom designed and focused on a specific bank. There are underground markets where one can buy a ready to use malware or order a custom one.

In the infected computer, the malware puts a configuration file which contains the target URL information, the place where the code will be injected and

the injection code, which is usually a JavaScript or HTML [9]. The malware evolves over the time to bypass new defence mechanisms. The recent malware has the capabilities of bypassing Transaction Authentication Number (TAN) and some of the malware from Beatrix family can temper computer's memory to alter the destination bank account [10]. Cross-Site Scripting (XSS) is also a popular method to attack a client-side page. Similar to SQL injection, XSS also exploits the hole in the web page that was introduced due to improper form validation and lack of sanitizing data before executing on the server side [11].

In broad sense there are three categories of XSS attacks: Non-persistent/reflective XSS, persistent/stored XSS and DOM-based XSS [12, 13]. In case of a reflective attack, the malicious code is reflected back from the server as a response (i.e. error message, welcome message, search result etc). In case of a stored XSS attack, the malicious code is saved in the server and when a client visits that infected web page, the malicious code is executed. Client-side script is modified and the payload is executed in case of the DOM-based attack.

JavaScript is not limited to client-side development only. For example, NoSQL and Node.js are now being used to design JavaScript-based web servers. The various kind of server-side JavaScript injections include denial of service, file system access, execution of binary files, NoSQL injection and much more [14].

Many banking malware performs Man-in-the-Browser attacks using JavaScript. It works like a Man-in-the-Middle attack but it works from inside of the actual client's computer. The effectiveness of Man-in-the-middle attack is reduced by raising public awareness about fake look-alike websites and device authentication mechanisms [15]. Man-in-the-Browser lets the malware to change the content what user sees by hijacking user's machine not only the web session. Therefore, device authentication mechanism is unable to detect this. This malware can automatically transfer the money by altering the content before sending the request to server and displays.

1.6 An Example of Banking Malware

In 2012 a variant of Zeus malware named ”Eurograbber” stole 36 million Euros from more than 30 thousand customers from various banks [16]. This malware used JavaScript to inject content and fool the user to provide sensitive information. There is also a mobile variant of Zeus called ”Zeus in the mobile” (ZITMO) to bypass two-factor authentication.

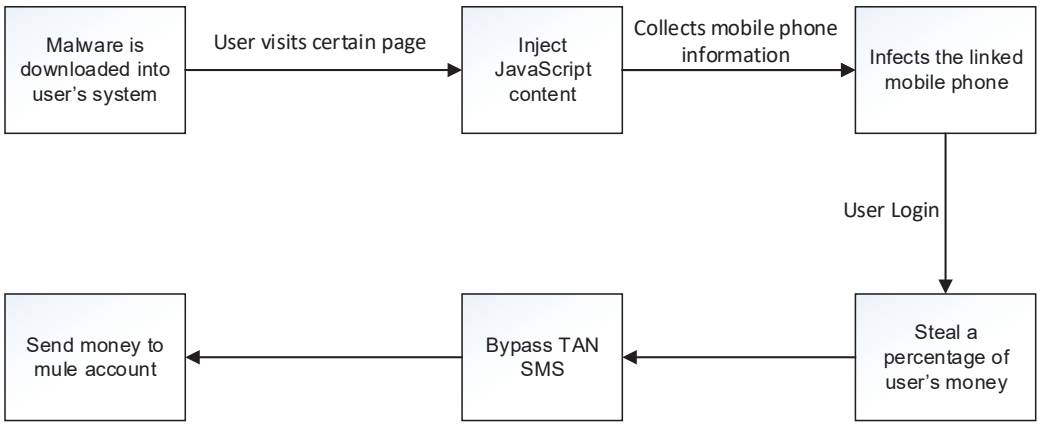

Figure 1.4: Eurograbber working method

Figure 1.4 demonstrates how Eurograbber works. In the first step, a trojan is downloaded into customer's computer via a malicious link. After that, for the first time when a customer visits the desired website, the malware injects JavaScript code showing false alert convince users to provide their mobile number and related information. Then ZITMO is installed into the user's mobile by providing a link to their phone. This ZIMTO has the capability of intercepting SMS containing Transaction Authentication Number and redirects it to attackers. Next time when user login in their account the malware steals a percentage of their money and send it to mule accounts.

1.7 Research Objectives

The end users are not always skilled enough to separate injected content from actual contents of a webpage. Ensuring a safe browsing experience for everyone is a challenging task. Even though web server is always protected with the high level of security measures, but individuals are not always have proper

security system for their devices. They often get infected by various ways, i.e. malicious email, phishing websites, contagious web link and sometimes via physical devices. It is very difficult to install the defensive solution to every single user.

Another challenge is to deal with various implementation methods of malware. Implementation of two malware might differ heavily but their behaviour might be very similar. So we will need different solutions for them. Because of a large volume of malware, it is not a practical approach to design a different solution for each of individual malware. Finally, we need a solution that can deal with unknown and new malware. Moreover, the existing malware might make changes in their source code and use obfuscation technique to hide. Machine learning techniques have been widely applied to solve cyber security problems. Data classification problems in cyber security are predominantly imbalanced data classification problems as the number of malware activities is significantly less than that of normal activities. Our research focuses on developing a machine learning based cyber threat detection technique, which will address the above-mentioned issues. The main research question is:

Q: How can we design an efficient solution for detecting cyber threats?

This question can be answered by answering the following sub-questions:

Q1: How can we make a robust system that can detect new and unexplored threats?

Q2: How to design an efficient classifying method suitable for internet security?

Q3: How to select an optimized Machine Learning model to ensure a good user experience?

Q4: How to integrate our solution with the existing application by minimum effort?

We will try to answer our main research question by proposing a new architecture for detecting cyber threat using machine learning. Our method

will find anomalies created by cyber attacks. Rather than focusing on a client-side solution, our method will work on the server side. This will ensure the solution to reach for every client.

Our first sub-question is about how we can develop a security system that will work on obfuscated code and unknown attacks. Using behaviour-based feature extraction and Machine Learning technique will address these issue. Because WebInject will surely modify the document object model (DOM) and if we select a good feature set we will be able to detect obfuscated and as well as unknown new attacks.

Our feature extractor will not reside on client's computer. It will be sent along with the web page to collect the features and send it for analysis. So it is a challenging task to transfer the code securely to the users without getting noticed. Second sub-questions demand some mechanism to hide our solution code so that attackers can not intercept it.

Feature extraction and classification process will add time overhead. This delay should be minimized as much as possible to give a better user experience. Our third sub research question demands a system that will perform well but without adding too much overhead. Selecting an optimal number of features to obtain the best result is a challenging task. We will focus on improving the accuracy of the proposed approach and reducing our execution time by selecting an optimal number of features and finding the most suitable classification technique that suits best for our feature set.

The fourth sub-question is about deploying the system in practical life. We have to come up with the solution which will assist the developers to integrate this solution while developing a new system. We might have to think about how to integrate the solution with the existing websites with minimum effort.

1.8 Organization of the thesis

The document is organized into the following chapters. Chapter 2 provides a literature review discussing some popular cyber attacks. Different methods to detect those attacks are presented with their shortcoming. Elements of machine learning and various types of machine learning techniques are discussed. Imbalanced data classification and its application in detection of cyber

security are presented later on this chapter. Chapter 3 discuss about popular cyber threats and our motivation for machine learning based threat detection mechanism. We present our preliminary findings and discuss the necessity of designing imbalanced data classfication technique to address cyber attacks. We also present a general architecture of our proposed model. Chapter 4 discusses our first proposed method, a cost sensitive piecewise linear classifier to deal with imbalacned data classification. We propose a modification of an existing classifier named "piecewise linear classifier" by introducing penalty parameter for separating lines. We defined minority classes and based on their imbalanced ratio, a penalty parameter is calculated. That penalty value changes the value of error function which pushes the separating lines towards minority classes to increase the accuracy of detecting minority polints. Chapter 5 discusses our second proposed method about hybrid classifier for imbalanced data. We combined supervised and unsupervised classification technique to overcome some drawbacks of our first method. We used an incremental clustering algorithm to find the groups of similar data points. We further apply restricted undersampling among those groups where the data is imbalanced. This technique increases the accuracy of minority classes without sacrificing too much accuracy from majority classes. Experimental setup and numerical resutls are presented in chapter 6. We used imbalanced dataset from python library and compared the results from both of our methods with four mainstream classifiers. Application of our proposed method in cyber security is discussed in chapter 7. We used four cyber security datasets including network traffic, credit card fraud, mobile malware and spam emails to test our proposed method in cyber security. We also used our own generated dataset by simulating web inject in google chrome browser. Chapter 8 presents a concluding remark and future research direction.

Chapter 2

Literature Review

In this chapter we discuss some common cyber attacks and review the existing methodology to address them. We also provide a literature review on supervised data classification algorithms for imbalanced datasets. We find the shortcomings of the existing methods and justify the necessity of developing new techniques to deal with cyber threats.

2.1 Introduction

With the advancement of internet based services and digital communication, individuals, government and organization are becoming the target for various cyber attacks. These attacks are causing damage by directly stealing money or valuable information and also by wasting resources and man power to prevent and recover from attacks.

2.1.1 Some popular cyber attacks

Some of the most common cyber threats are: denial of service (DoS), man in the middle (MitM), cross-site-scripting(XSS), SQL injection, malware, ransomware. As the prevention techniques for these attacks are getting better, the threats are also evolving to bypass the detection mechanisms.

Denial of Service (DoS). A denial of service is a coordinated attack to prevent actual users from using the service. A distributed denial of service (DDoS) attack is performed by a group of compromised computers. These

compromised computers are used to exhaust the resources and bandwidth of the target computer. The target computers are called ”primary victim” and the compromised computers are called ”secondary victims” [17]. Unlike most of the cyber attacks DDoS attackers does not get benefited directly. This attack can be used to get edge over competitors by disrupting their services. There are few methods to prevent and mitigate the effects of DDoS attacks. Using configured firewall, increasing the connection queue size, decreasing timeout period, applying black hole filtering and others are some popular methods to deal with the DDoS attack.

Man in The Middle. Man in the middle attack is performed by eavesdropping between the communication of two parties. This attack has the capability of listening to the conversation between communicating hosts and able to steal the data. There are various techniques for man in the middle attack, such as: sniffing, packet injection, session hijacking, SSL stripping [18]. Man in the middle attack is very hard to detect and they remain unnoticed until the last moment. Using precautionary methods, we can minimize the chance of becoming victims of the man in the middle attack. Some of these techniques are: being aware of connecting to open network, using strong encryption on access points, using https to ensure public-private key encryption, using VPN.

SQL injection. SQL injection is carried out by injecting SQL query code as user input data. Attacker gains unauthorized access to the database and may steal sensitive information. It can damage or alter the content of the database as well. SQL injection can be done by injecting content through user input, cookies, server variables and second order injection [19]. The target of this attack is to identify injectable parameters, database finger-printing, determine database schema, extract data, add or modify data, bypass authentication and execute remote commands. The detection of SQL injection is difficult but some prevention measures can be taken to stop some of this attack. Input validation is one of the most straightforward solution to address this problem. Input validation can be done by checking input type, pattern matching and encoding inputsc.

Ransomware. By ransomware attacks, the attacker gains access to the data of target device and locks the data applying the encryption. The demand money in exchange for unlocking the data. Ransomware was first introduced in 1980 and evolving over time to bypass new defence methods [20]. Countermeasures of ransomware broadly categorized in three categories. They are: prevention, detection and prediction [20]. Prevention techniques are applied to stop the target from being infected. There are several detection techniques, such as: event-based detection, anomaly based detection and statistical based detection.

XSS. With the increasing availability of internet, web application became the standard to provide services and representing data over World Wide Web. These services became the target for attackers to exploit the vulnerabilities and execute malicious scripts on the victim's web browser. Using cross site scripting the attackers can steal data, browser cookies, passwords, credit card details etc. Several client-side and server-side solution for detecting XSS have been proposed. Our research focus is to design a machine learning based server side solution.

2.2 Detecting cyber threats

Cyber security is the technology and method to protect digital contents, computers, networks, programs and data from attack, unauthorized access or destruction [21]. The components of cyber security systems are network security systems and host security systems. Network security system works by monitoring network traffic and host based system works by monitoring software environment, processes and file activities. Depending on working method, cyber security can be divided in three categories. They are: signature-based techniques, anomaly-based techniques and hybrid techniques [21].

2.2.1 Signature based detection technique

Signature based techniques can detect known threats by comparing against set of known rules and signature database. The main benefit of signature based detection technique is very low false positive rates. It identifies an activity as

threat, if it matches with a already known set of rules and signatures. The signature based techniques fails to detect new and unexplored threats.

One of the earlier work on signature based detection technique is network security monitor (NSM) [22]. They designed a lan monitoring system by developing profile of network resource uses. By comparing current usage pattern with earlier profile they attacks are identified. NetRanger [23], a commercial network monitoring system is built based on this approach. Snort is a open source lightweight signature based network monitoring tool [24] It is a cross platform tool which is easy to deploy without disrupting the operation of the system and suitable for small sized network. Snort sniffs real-time network traffic and logs the misuses based on matching the content with predefined features rules. It also have the capability of generating real-time alert if an attack is detected.

Kirda et al. [25] proposed a custom firewall system for the client side which will decide whether a web page request from the browser is secured or not based on some rules. The user can make their own custom rule depending on their needs. A.K. Dalai et al. [12] proposed a server-side solution which will filter the data before executing based on some criteria. They tested using 230 attack vectors. Some of them is not functioning due to the change of browser policy and functions. The authors of [12] tested using five different browsers. Their solution introduces a little overhead in terms of execution time. The papers [13, 26] used a genetic algorithm to generate suitable XSS from an initial random XSS by applying crossover and mutation and tries to find whether any path in the webpage executes that code.

2.2.2 Anomaly based detection technique

Anomaly based detection techniques work by generating a model with normal behaviour. It identifies an activity as threat if it differs from the normal behaviour. This method is capable of detecting new and unexplored threats, but it has higher false positive rate as this method consider any previously unseen behaviour as a threat.

Malware codes are very often obfuscated and their implementation strategy changes frequently over time. There are several obfuscation technique and packing is one of the most common technique. The availability of various types of packers makes it difficult for traditional signature based packer identifiers to identify malware. A packer classification framework using pattern recognition is designed by Sun *et al.* [27]. Various malware, having similar behaviour, can be written in different programming languages or developed using different platforms. The signature based solution for one might not be applicable to others. But the behaviour of malware remains similar regardless of their development environment. This is the reason why behaviour-based malware analysis brought the attention of researchers in recent time.

Most of the malware adds additional forms or messages on the fly inside the document object model (DOM) a web page. Criscione et al. [28] proposed a method for extracting WebInject signatures from a DOM. They used a java library to automatically visit a list of URLs from several virtual machines to collect the DOM and compare them to find the benign differences. They also collected one DOM from infected virtual machine visiting the same URLs and finds the differences between clean and infected DOMs. A graph based framework combined with anomaly detection is applied to detect insider threat [29]. Their proposed framework contains two main units, "Graphical Processing Unit" (GPU) and "Anomaly Detection Unit" (ADU). Enterprise network data is fed into GPU to form graph representing interrelation between entities and the graph information is passed to ADU. Output of ADU determines the anomaly score for each user, which is used to identify the threat inside an organization.

Continella et al. [30] also used the DOM comparison to find the changes done by malware. Another client-side approach is to integrate the solution into the browser done by Lekies et al. [31]. They modified the source code for Chromium browser to use it for identifying DOMbased XSS issues. Stock et al. [32] worked on top Alexa websites and based on their defined matrices (Number of operations, number of involved functions, number of involved contexts etc) they find the website that contains at least one XSS vulnerabilities. System-centric behaviour-based malware detector is becoming popular because of its

effectiveness against new and unexplored malware. Fattori et al. [33] used the general interactions between benign programs with the operating system to make a behavioural model. The advantage of their system is that they did not need any malware samples to train their model.

Sandboxing mechanism is used to ensure secure browsing experience by limiting permission of JavaScript and other processes. The same-origin policy also restricts instances of one process to use resources of its own origin only [25]. Even with the presence of this technique, XSS can still attack user and steal valuable information. This can be done by tricking the user to click the malicious link and downloads malicious JavaScript code into the trusted site and execute the code. This allows that malicious code to use the resources of the trusted site and collect the valuable information.

2.2.3 Hybrid detection technique

This technique may combine two or more detection techniques. A signature-based and anomaly-based detection method can be combined to create a hybrid method too. Anomaly based method can detect unknown attacks but comes at a cost of high false positive rates. The goal of hybrid method is to increase the detection rate of unknown attacks without generating too many false positive results.

Neural network combined with keyword selection are used for intrusion detection [34]. This hybrid method combines a signature based method (keyword selection) with anomaly based detection (neural network). The authors counted the word frequency of a telnet session to a network and this keyword statics are forwarded to a neural network for classification. A hybrid network security system is proposed by Aydin *et al.* [35] which combined snort (a signature based intrusion detection system) with packet header anomaly detection (PHAD) and network traffic anomaly detection (NETAD). Another hybrid intrusion detection system (HIDS) is proposed by Saleh *et al.* [36]. They combined three different techniques to design their method. A Naïve Bayes feature selection is used to reduce the dimension of the train data. Using optimized support vector machine outlier data is reduced. Finally Prioritized K-Nearest

Neighbour classifier is applied to detect the attack.

2.3 Elements of Machine Learning

Machine learning is a method of designing a computer system that can automatically learn from sample data and past experience without using explicit programming. Before machine learning was introduced, the knowledge-based system was used to make a decision by computers. A knowledge base is comprised of rules defining each and every possible scenario and corresponding actions, usually, in the form of *if-then* rules [37]. A large number of rules are needed to create a knowledge base which is a very challenging task and sometimes frustrating too.

Machine learning can be categorized into three main categories [38]:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

There is another type of learning called "Ensemble Learning" which combines multiple learning techniques to solve the problem and combined results are used to create the final result.

2.3.1 Supervised Learning

In supervised learning, the machine tries to find a model that maps from an input (which is usually a vector) to an output (which contains two or more labels). Given a sample set of training data machine builds a model that can be used to differentiate between those labels. Classification is a type of supervised learning where a group of training data are assigned a label and given a new input without providing the labels, the machine will predict its class.

Some of the examples of supervised learning algorithms are Support Vector Machine (SVM) [39], Decision Tree, Adaboost [40, 41], Naive Bayes, Random Forest [42] etc. Supervised learning is used in many practical fields, such as

Face recognition, Optical Character Recognition (OCR), Spam filtering, Voice recognition, Medical diagnosis etc. In this study, we use two classification methods for classifying cyber threats.

Regression is another type of supervised learning where a probabilistic output value is estimated based on correlation between input and output values of training data. Linear regression is one of the most popular regression method. Linear regression is used where the prediction values are continious (i.e salary prediction, house price prediction etc). This method finds a best fit line to establish a relation between output variable and input variable(s). Logistic regression is applied when the output label is categorical or contains discrete values. Logistic regression is used in fraud detection, loan payback scoring or any other situation where the outcome is a binary value. Multivariate Regression algorithm is used when more than one prediction values are dependant on more than one input values. One the popular multivariate regression method is Multivariate adaptive regression splines (MARS) [43].

2.3.2 Unsupervised Learning

In unsupervised learning, it is, in general, assumed that a prior information about the data is not available and input data is provided without any known labels. The goal is to categorize the data into different groups based on similarities of instances. Sometimes the number of desired groups or cluster is provided and sometimes not. K-means clustering, Mean-shift clustering, Expectation-maximisation (EM) Algorithm, Fuzzy C-means (FCM) clustering, DBSCAN [44], MGKM [45] are some of the examples of unsupervised learning.

Some of the real-life application of unsupervised learning are Document clustering (Digital forensics), Outliers detection, Customer segmentation, Grouping peoples based on their age or gender. Sometimes unsupervised learning is used with supervised learning for practical use. Another unsupervised learning called Isolation forest is used to detect abnormal behaviour [46] and raising alert for the system administrator.

2.3.3 Reinforcement Learning

In some problems, only a single action may not be helpful. To reach the goal a sequence of actions is needed. An example might be playing chess, where a sequence of moves determines the final outcome. Because of limited perception capabilities based on partial observation the decision has to be made. Reinforcement learning has two main components: agent and environment. Provided the current state, the agent performs an action on the environment. Environment then provides the feedback with a pair containing next state and reward values. Based on the reward the agent adjust its next action. The goal of the system is to maximize the rewards.

Model-based reinforcement learning dynamically create an internal model of the transition and calculate transition probability, which is used to make an optimal decision. But it is not suitable for world with large number of states and action space. A model-free reinforcement learning is trained with trial-and-error. The benefit of model-free learning is that, it does not need to store the intermediate transition state-action pair. Reinforcement learning can be used to train computer to play games, training robots, resource management in computer cluster [47], traffic signal control [48], web system configuration [49].

2.3.4 Semi-supervised Learning

Semi-supervised learning is a special type of machine learning technique which is halfway between supervised and unsupervised learning models. It is suitable when training data has small number of labelled data and a large number of unlabelled data. Sometimes labelling data needs a lot of human effort and time, which may not be ideal for many applications. Semi-supervised learning can be applied to solve those problems. Based on prediction behaviour, semi-supervised learning can be divided into two categories: Inductive learning and Transductive learning. When a semi-supervised learning predicts the label of some test data points, it is called inductive learning. The goal of Transductive learning model is to use both labeled and unlabeled instances for training algorithms. Semi-supervised learning is applied to solve many real life

problems, such as video surveillance, spam filtering, speech recognition, person identification [50], activity recognition [51], expression classification [52] etc.

2.4 Machine Learning in Cyber Security

There are various practical life application of machine learning, for example, self-driving car, chess playing robots, self-learning chatbots, natural language processing, image manipulating tools etc. A multiple instance learning is a kind of machine learning technique, where instead of labelling individual training points, a set of points also known as bags are assigned a label. This labelled bags are used to train the model. Multiple instance learning is used to solve several real life problems, such as image classification [53, 54], Melanoma detection [55].

Machine learning algorithms have been also applied in the field of computer or information security. Some widely used machine learning techniques in cyber-security area are discussed below:

Artificial Neural Network. A neural network consists of several layers, input layer, hidden layers and output layer. Each layer has some nodes also called as neuron. The node performs computation based on provided inputs and pass the output to the next layer. Deep learning is a type of neural network with multiple hidden layers. Neural networks are used in cyber security area.

A profit driven artificial neural network is proposed for fraud detection [56]. Instead of data driven approach they used profit driven model, where they assigned various penalty for misclassification of instances depending on their importance and the goal of their Artificial Neural Network is to minimize the overall penalty value. They used a cost matrix where different penalty values are assigned for false negative and false positive classification and no penalties are applied for true negative or true positive. Instead of focusing on percentage of correctly identifying instances their Neural Network adjusts its load to maximize the overall profit gain.

Several types of classifying methods are used to detect Android malware in [57]. They installed 200 goodware and 115 malware application in a controlled

environment and collected data is applied to train a machine learning model which is used to detect malware. The network traffic is applied to detect mobile malware in [58]. They compared various methods including combination of SMOTE and SVM, cost-sensitive SVM and cost-sensitive C4.5S.

A misuse detection system is proposed by Cannady [59], where KNN is used as the multi-category classifier. A network monitor named RealSecure is used to simulate three thousands attacks and seven thousand normal events. Data collected from this simulation is used in testing of their system.

Naïve Bayes. This classifier is a probabilistic classifier which works on the principle of Bayes theorem. Naive Bayes classifier assumes the feature variables of the dataset are independent and not correlated to each other. It is a highly scalable classifier with a linear training time. It is suitable for classification with small training dataset. A simple form of Bayesian network is used to design a framework for anomaly detection [60].

In the paper [61], the authors used Naive Bayes to detect spam email instead of using the rule-based system. Still, nowadays this is one of the main approaches for detecting spam emails. An intrusion detection system with feature reduction was proposed Saurabh Mukherjee and Dr.Neelam Sharma [62]. Using three different feature selection methods, the authors first eliminated redundant or irrelevant features. Then the reduced featured dataset was classified using Naive Bayes classifier.

Decision Tree. A decision tree is a tree-like predictive model. The internal nodes of the tree works as a test based on input feature and the leaves represents classes. Features are preferred to be categorical value for building a decision tree. If not, then the values are converted into discrete values before creating the model. A test sample is classified by testing each feature against the internal nodes of decision tree. Decision tree can be used to solve both regression and classification. ID3 [63] and C4.5 [64] are the most popular decision tree classifiers. These are greedy algorithm and use top-down approach to create a tree iteratively.

A random forest is an ensemble classifier which is a collection of decision trees. A random forest classifier is used to detect spam email. A bag of words is constructed from email body and feature vectors are created using that. Then the system is trained with sample emails containing both spam and normal emails. They used random forest classifier for training the model. Once the model is trained, the system predicts the class of new incoming email. This technique is used to develop a custom firewall system for detecting targeted malicious emails for a particular company [65]. They mainly focus on the custom data set with exclusive features. These features are collected not only from the email body but also from the company and employee's personal information. Combining this information they make a powerful feature descriptor and a good feature always leads to better classification.

Support Vector Machine. Support Vector Machine (SVM) is a supervised machine learning technique that analyze data and recognize patterns, used for classification and regression analysis. An SVM training algorithm works on some training data. It creates a hyperplane for dividing train samples into two separate groups so that the distance from the hyperplane to the closest element in each group is maximized. Testing data are mapped into the same space and its class is predicted based on which side of separating plane it falls.

Given a set of labeled training data:

$$D = \left\{(X_i, y_i) \mid X_i \in R^P, y_i \in \{-1, 1\}\right\}_{i=1}^{n}$$

Here X_i is a p-dimensional vector for representing the y_i labeled data. A hyperplane is derived for a set of points X so that $\mathbf{W}.X - b = 0$ is satisfied. Here W is a normal vector along with the plane. For any testing point for positive class $\mathbf{W}.X - b \geq 0$ and for negative class $\mathbf{W}.X - b \leq 0$.

SVM is primarily designed for binary class separation. But it can be used for multiclass problem as well. Traditional methods for multiclass problems are one-vs-rest and pairwise approaches. Recently simultaneous classification and various loss functions are used for multiclass classification. One-vs-rest which is also known as one-vs-all (OVA) solves K different binary problems that classifies "class k" versus "the rest classes" for $k = 1, \ldots, K$. It assigns

a test sample to the class giving the most positive value for the solution from the k_th problem. Pairwise approach solves $\binom{K}{2}$ different binary problems that classifies "class k" versus "class j" for all $j \neq k$. For prediction at a point, each classifier is queried once and issues a vote. The class with the maximum number of votes is the winner.

TF-IDF method is used by [66] for extracting features and they used Support Vector Machine (SVM) for classification method for spam filtering. A support vector machine combined with unsupervised intrusion detection system is used by Ghanem *et al.* [67] for intrusion and cyber-attack detection. A variant of SVM called least square support vector machine (LS-SVM) is used for intrusion detection system [68]. SVM is used to detect malicious socket address by Chandrapal *et al.* [69].

Bayesian Network. A Bayesian network is a type of probabilistic graph which aim to model the variables and the relationships between them. It is a directed acyclic graph (DAG) where nodes represent a unique random variable and the edges are used to present the relationship between them. Bayesian network is useful to visualize the structure of the model and provides insight about the relationship between the random variables.

A bayesian network along with other classifiers are used to detect spam email [70]. They used Local Binary Pattern (LBP) for feature extraction. LBP is a very popular method for extracting a feature from the image. It is used for various purposes, like expression recognition, face detection, gender classification etc. Similar to LBP, shifted binary pattern is used for extracting a feature from text data [70]. LBP extracts a feature from an image by comparing centre pixel with 8 directional surrounding neighbour pixels. This shifted binary patterns method uses two directional neighbours (left and right) for text data. From a centre character, it checks the P_L number of left neighbours and P_R neighbours in right. They used $P_L + P_R = 8$. For each comparison, if the value is more or equal than the centre they assign 1, otherwise 0. Finally, the feature value is obtained by converting the 8 bits into a decimal value. They calculated values for nine variants of picking (P_L, P_R) value pairs (i.e (0,8), (1,7), (2,6) and so on). Combined all of them they found a feature histogram

that used in training and test.

Association Rules and Fuzzy Association Rules. Association rule is presented as $A \rightarrow B$, where A is a set of items and B is usually a single item. Association rules suffers from a hard boundary problem as it provides a binary decision. It either completely accepts or rejects "then" part of the rules depending on the conditional part being satisfied or not. Fuzzy association allows to assign probabilistic measure to each possible outcomes of B. Brahmi [71] used association rules to find the relation between TCP/IP parameters and attack types in DARPA 1998 dataset.

Clustering. There are different types of clustering: hierarchical clustering, partitional clustering, density-based clustering, fuzzy clustering and model-based clustering. Various algorithms have been developed to solve each of these clustering problems. Nonsmooth optimization problem appears in many applied fields, such as: image denoising, optimal control, data mining, economics, computational physics and chemistry etc. Five clustering algorithm combining local search optimization algorithm and incremental approach are presented by Bagirov *et al.* [72] to address those problems.

Clustering algorithms have been applied significantly less than the supervised data classification algorithms to solve internet security problems. Density-based clustering algorithms were applied in network traffic monitoring. The DBSCAN clustering algorithm is applied by Blower and Williams in [73] for anomaly detection. A novel kNN clustering method is proposed by Xie *et al.* [74] for anomaly detection in wireless sensor network. An optimization based incremental clustering algorithm is proposed for intrusion detection system by Taheri *et al.* [75]. The goal was to reduce false alert with high detection rate. They divided clusters into two subsets (normal and stable) and calculated the distance between centroids of normal clusters. The outliers, which represents the cyberattack is detected among stable clusters using the distance calculated earlier. Authorship analysis of phishing email is done using multiple clustering algorithms [76]. At first features are extracted from email by counting frequencies of word. Inverse document frequency weights are applied to give weights to features based on frequency of the word in the document. Finally four clus-

tering algorithms (k-means, MS-MGKM, INCA and DCClust) are applied to find groups of similar phishing emails.

Ensemble Learning. A machine learning algorithm maybe suitable for very specific type of data. One good classifying technique for one dataset may not perform well for another dataset. Ensemble learning combines multiple training algorithm to get a better prediction. It often combines several weak learners to make a strong one. Adaboost is one of the popular ensemble learning technique. A hybrid approach for intrusion detection is proposed by Zhang *et al.* [77]. They used a rule based system to filter out previously seen attacks. Then outlier analysis is done by random-forest classifier and if the test sample belongs to an attack, then the classification rule is applied to find the type of the attack. They also used oversampling technique to generate more sample for the type off attacks containing less training points.

Evolutionary Algorithm. Evolutionary computation is inspired by biological evolution. An initial solution is selected and by iterating through multiple generation the solution is improved. The genetic algorithm is an example of evolutionary computation. Basic operation in genetic algorithms are: selection, crossover and mutation. In the genetic algorithm usually a randomly generation population is selected at the initial step. By applying a fitness test the feasibility of the population is measured and the population with higher fitness values are used for crossover and mutation operation to obtain the next generation. A crossover is done between two samples and mutation is performed to change one sample. Li [78] used the genetic algorithm for intrusion detection. They used DARPA dataset to test their proposed method.

Hidden Markov Models. A markov process is a mathematical system that represents transition between some states following some probabilistic rules. The probability of the transition depends on the current state only, it does not depend on how the process arrived to the current state. Markov chain assigns probability on observable events but in many cases some events are not observable. Hidden markov model assigns probability of transition with unknown parameters.

Hidden Markov Models (HMM) is used to predict multi-step attack and generate alert by Sendi *et al.* [79]. To extract the interaction between attackers and network HMM is used. Several works are done in detecting fraud using HMM. Tang *et al.* [80] proposed a data driven hidden markov model to detect fraudulent medical claims. Hmm is applied in intrusion detection system [81, 82]. The author of [82] used HMM for a host-based intrusion detection system. The authors used a data preprocessing method to reduce the data and decrease the training time. A HMM based credit card fraud detection system is proposed by Iyer *et al.* [83]. They trained a HMM with normal card transaction and if an incoming transaction is flagged as fraud by their system with high probability, the transaction is consider to be fraudulent. HMM is used for malware detection as well. The work done by Annachhatre *et al.* [84], Austin *et al.* [85], Gharacheh *et al.* [86] and Imran *et al.* [87] are some of the example using HMM for detecting malware.

Inductive Learning. Deductive and inductive learning are two major techniques for inferring information. Deductive learning is a top-down approach where it goes from general rule to specific example. Inductive learning is a bottom up approach where it goes from specific example to general rules and learners discover rules by observing specific examples. Inductive learning starts with a specific observation and formulates the general conclusion theory. C4.5 and AQ are two examples of popular inductive learning algorithm.

Sequential Pattern Mining. A sequence is an ordered list of elements or events. Given a sequential input data, sequential pattern mining finds the relevant pattern between provided examples. Sequence pattern mining is used to detect database intrusion by analyzing the sequence of patterns in a dataset.

2.5 Imbalanced data classification

The performance of machine learning based solution mostly depends on an efficient and accurate classification technique which is suitable for the problem domain. In cyber security area, the number of instances of normal activity is significantly high compared to the number of instances of malicious activity. This creates an imbalanced training data.

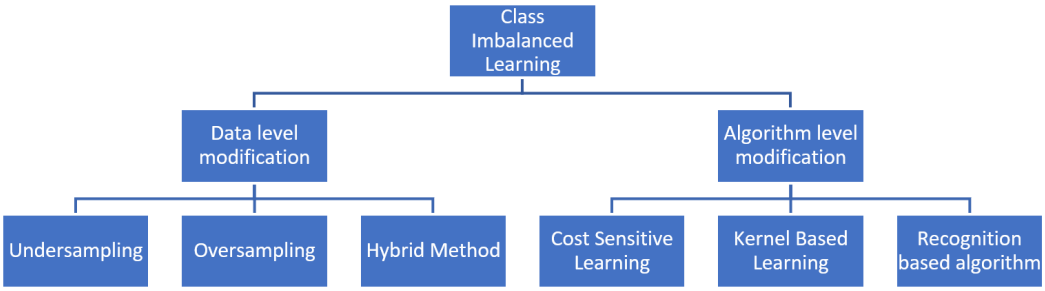

Figure 2.1: Categories of class imbalanced learning

Imbalanced data classification method has drawn attention of a lot of researchers due to its application in many practical scenarios. Various methods are proposed to deal with imbalanced data classification. These methods can be categorized as data level techniques or algorithmic level techniques [88]. Each of them can be further divided into sub categories. We summarize the categories in Figure 2.1. Some researchers considered cost-sensitive algorithms as a combination of both data level and algorithmic level technique [89, 90].

EUSBoost, which is a combination of image processing and machine learning is proposed in [91]. They used a combination of undersampling method with boosting. This method is similar to RUSBoost. The key difference is instead of using random undersmapling, they used evolutionary undersampling (EUS) before boosting and obtained a better result.

The performance of several classifier methods that deals with class imbalanced data are compared in [92]. They also categorized the approaches to deal with imbalanced datasets based on their working principle.

Data level techniques balance the cardinality of different classes in a dataset by either adding more synthetic points into minority classes or removing points from majority classes. Undersampling and oversampling are two main variants of data level techniques. A lot of variants of these methods are proposed by various researchers to deal with imbalanced data classification.

The oversampling method duplicates the samples in the minority classes in order to enhance their cardinality [93]. Several techniques are proposed for oversampling. The simplest oversampling method is random oversampling (ROS), which duplicates randomly selected minority objects.

Figure 2.2 demonstrates general idea of oversampling method. The drawback of this approach is that minority objects are grouped together in small

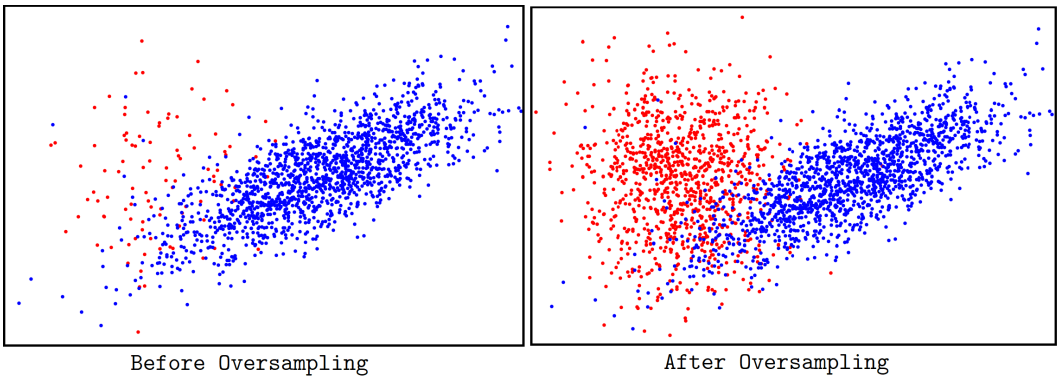

Figure 2.2: Oversampling Method

areas from where the seed for oversampling is selected. This will cause problem for the classifiers with over-fitting problem [94]. The informed oversampling approach like synthetic minority over-sampling techniques (SMOTE) generates synthetic minority class samples to balance the class distribution [95] which eliminates the problem of ROS. It has received a lot of admiration and has the extensive range of practical applications. Many variants of SMOTE have been proposed like adaptive synthetic sampling approach (AdaSyn) [96], Borderline-SMOTE [97], Majority weighted minority oversampling technique and weighted kernel based SMOTE. The drawbacks of oversampling method is that it adds time and memory overhead [98], can cause over-fitting and some features can not be synthetically generated or lose its property in synthetic data.

Undersampling method reduces the number of points from majority classes to make a balance training set. The simplest undersampling method is random undersampling, which randomely removes points from majority classes to balance the data.

Figure 2.3 shows general working methodology of undersampling method. The main problem of undersampling is removing points may remove significant information from dataset which will lead to a poor classification. Instead of removing points, EasyEnsemble and BalanceCascade algorithms [99] create a subset of majority points and a classifier is trained for each subset and all

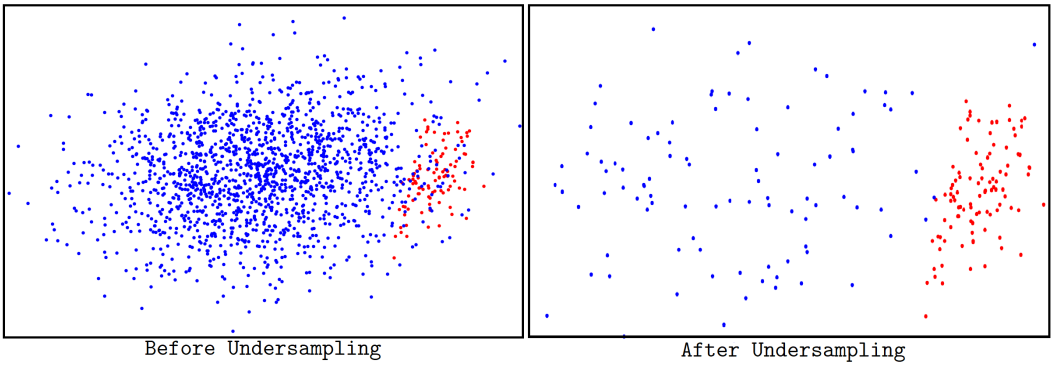

Figure 2.3: Undersampling Method

minority points. All individual classifiers for subsets are combined to get the final decision. BalanceCascade's working method is similar to EasyEnsemble. The main difference is the subset selection. EasyEnsemble randomly selects a subset from majority points in every step, whereas in BalanceCascade the points in majority classes that are correctly classified are not included in the next step. In our experiment we used the EasyEnsemble classifier to detect malicious activities.

Algorithmic level methods deal with class imbalance learning by modifying the classifier design. There are both supervised and unsupervised classification methods to deal with imbalanced data classification. Some methods are designed by combining both supervised and unsupervised method. Nguyen *et al.* [100] prposed such a hybrid method where they create clusters of the data and used the cluster centre to represent all the points that belongs to that cluster. After that a modified feedforward neural network is used to classify the data. A machine learning model called extreme learning machine (ELM) is proposed by Huang *et al.* [101] for general classification. Several variants of this method is proposed by assigning different weights to the training sample for better classification results with imbalanced data. Weighted ELM (WELM) is proposed by Zong *et al.* , [102] which assigns different weights for training sample according to the user need. This can be generalized as cost sensitive learning. Boosting weighted ELM (BWELM) is a variants of WELM, where

adaboost framework is used with WELM for quick execution [103]. EWELM, proposed by Zhang *et al.* , where ensemble approach is used on WELM to improve classification performance by combining several learners [104].

Algorithmic level methods modify a classifier to address the imbalance learning task. A significant subclass of these methods is the class of cost-sensitive methods. Most mainstream classifiers assume that the misclassification costs are the same. Unlike them, the cost-sensitive methods assign more penalty for misclassifying the minority class samples than misclassifying the majority class samples, that is, misclassification of minority class samples is much more costly. WELM [102] and WSVM [105] are two representatives of these methods.

WELM minimizes the weighted cumulative error with respect to each sample. It uses two weighting schemes to assign class-wise weights to the samples. These weighting schemes assign more weight to increase the impact of the minority class while diminishing the relative impact of the majority class. WSVM also minimizes the weighted cumulative error and applies the SVM as the base classifier.

The cost-sensitive methods can also be designed by combining both the algorithmic level and data level approaches [92]. Algorithms based on this approach try to address the imbalance of the data by adjusting weights and modifying probabilities in classic algorithms [89]. In the paper [106], the authors introduce the cost-sensitive learning algorithm by assigning different costs for various predictions and try to create a model with the minimum cost.

A random forest quantile classifier (RFQ) is proposed in [107] to address the class imbalance problem. The proposed q^*-classifier is an extension of a previously designed quantile classifier, called q-classifier. The quantile classifier classifies a sample as a member of a certain class if it exceeds a certain threshold $0 < q < 1$. For the median classifier this value is $q = 0.5$. The q^*-classifier simultaneously maximizes the sum of true positive and negative rates and minimizes the weighted risk.

Different algorithms based on the combination of the algorithmic and data level approaches are proposed in [108, **?**, 109, 110]. These algorithms include the generalized class-specific kernelized extreme learning machine (GSKELM), UnderBagging based kernelized ELM (UBKELM), UnderBagging based reduced Kernelized WELM (UBRK-WELM).

The cost-sensitive methods assign more penalty for misclassifying the minority class samples with respect to the majority class samples, that is, misclassification of minority class samples is much more costly.

2.5.1 Application of imbalanced data classification.

Imbalanced learning have been used for solving several real life problems. The paper [111] discusses the impact of imbalanced dataset on cancer diagnosis and experiments with 18 different data balancing techniques from both undersampling and oversampling strategies. They used four different classifiers to observe the performance of those techniques. The automated detection for fault in wind turbine is proposed in [112]. The authors addressed the problem and proposed a solution using deep learning. They also contributed towards feature extraction method for better fault detection.

Dynamic detection of banking fraud using machine learning is a very challenging task. Fraudulent behaviour is dynamic and heavily diversified from person to person. Fraud-related incidents occurred in a limited number of times and create a highly imbalanced dataset. A faster detection mechanism is also very crucial for a real-time detection.

The authors in [113] proposed a new method called ContrastMiner. It integrates multiple data mining models such as: costsensitive neural network, contrast pattern mining, and decision forest. Their approach is to find unusual behaviour from transactions. From a single online banking a session of series of events are captured with a set of features to describe each event. Contrast are calculated and depending on the finding it either categorized as fraudulent activity to raise an alert or considered as a safe event.

Artificial Neural Network based solution is proposed for fraud detection by Zakaryazad Duman [56]. Instead of data driven approach they used profit driven model, where they assigned various penalty for misclassification of instances depending on their importance and the goal of their Artificial Neural Network is to minimize the overall penalty value. They used a cost matrix where different penalty values are assigned for false negative and false positive classification and no penalties are applied for true negative or true positive. Instead of focusing on percentage of correctly identifying instances their Neural Network adjusts its load to maximize the overall profit gain.

Wang *et al.* [114] used two undersampling and two ensemble methods to predict software defect. Their goal is to predict the defect prone module of a large software. This will eliminate the need of thorough testing of whole software. This will helpful for a small company which does not have resources for such thorough testing.

Machine learning based approaches are used to solve many real life problems. A class imbalance learning model is proposed to predict software defect in [114]. To eliminate the need of thorough testing of a software, their model predicts defect prone module of large software. The authors in [91] used an undersampling method to detect breast cancer.

2.6 Conclusion

In this chapter we discussed several methods to detect various types of cyber attacks and found signature based detection methods are only effective detecting previously explored threats. It has a very low false positive rate as it identifies threats based on signature database. It fails to detect new threats until their signature is discovered and updated in the database. It is very difficult to define proper rules in anomaly based detection engines. Each protocol being analyzed, implemented and tested for accuracy which is not always an easy task. If malicious activity resembles normal activity then this method fails to detect threat. Anomaly based detection requires more hardware compared to signature based detection method.

A machine learning based server side solution will resolve limitations of many signature based and anomaly based detection methods. It will be able to detect new an unexplored threat. Main drawback of machine learning based method is to train a good model for detection. Finding a good training data and suitable classification method is always challenging task. False positive rate for machine learning based detection is also high compared to the signature based method.

The accuracy of a machine learning based detection method heavily depends on the training data; and the dataset in cyber security domain is imbalanced. So finding a suitable classifying method that can deal with imbalanced

data plays a crucial role in machine learning based detection. Undersampling method removes a large number of training points which leads poor classification rate for majority class. This will have bad impact in many real life problems. Oversampling adds synthetic training points to the dataset, which creates over-fitting problem and in some cases it is not feasible to generate synthetic data.

To obtain a better classification results on imbalanced data we proposed two different classifying methods and also applied into cyber threat detection. Further research can be done focusing on keeping the solution secure and easy integration to the existing system.

To make the solution available for user is also a challenging task. Client side solution needed to be deployed in all of the client's machine. Not all the clients are able to follow the instruction which makes it hard to safeguard everyone by a client side solution. A server side solution will be easy to deploy for all the clients.

Chapter 3

Machine Learning based Cyber Threat Detection: Motivation

In this chapter we discuss few popular cyber threats, their working methodologies and the general machine learning based framework to detect them. We also review some of the existing works with their strengths and drawbacks, which motivated us to develop a machine learning based server side solution for detecting cyber threats.

3.1 Introduction

With the advancement of technology and ease of access to the internet, a lot of individuals and organizations are becoming the target for various cyber attacks such as malware, ransomware, spyware, SQL-injection, web-inject etc. The target of these attacks is to steal money or valuable information from the victims. Spyware collects personalized information silently from the infected machine. It can be used to collect network traffic, keystrokes and other information from the target. Often Adware and Spyware are combined to show customized advertisement. Ransomware restricts the user from using certain functionality or locks down files of the target user. It demands money to remove the restriction. Various types of web-injects are used to target individuals and organizations. SQL injection and cross-site scripting (XSS) are two most popular methods of web-inject.

A majority portion of online attacks is now done by WebInject. On the fly

client-side web content generation techniques make it easier for attackers to modify the content of a website dynamically and gain access to valuable information. The end users are not always skilled enough to differentiate between injected content and actual contents of a webpage.

In broad category there are three types of XSS attacks: Non-persistent or reflective, persistent or stored and DOM-based [12, 13]. A document object model (DOM) represents a webpage as nodes and objects. By interacting with DOM a program can change the document structure, style and content. It is an object oriented structure of the webpage and can be modified using scripting language. To detect the attack, we need to identify these changes inside the DOM caused by the malware.

3.2 A sample of WebInjection

A sample WebInject is simulated by designing a Google Chrome extension, which checks for a list of target URLs and inject some contents into the webpage if target is found. This extension does not perform any real malicious activity, but there are some malicious browser extension that can cause harm to the infected machine.

```
$('document').ready(function(){
   var new_item = document.createElement("LI");
   var textnode = document.createTextNode("For Security Reason, Enter Password Again");
   new_item.style.fontSize = "21px";
   new_item.appendChild(textnode);

   var list = document.getElementById("cssParsedBox");
   list.insertBefore(new_item, list.childNodes[2]);

   var table = $('.LoginTextBold').parents('table').first();
   var row = table.find('input[type=password]').parents('tr').first();
   row.after('<tr><td class="LoginTextBold">Confirm Password:</td><td align="left"
colspan="1"><input type="password" value="" onpaste="return false;" oncopy="return false;"
name="fldPassword" style="width:262px;" class=""></td></tr>');
});
```

Figure 3.1: Sample of Injection code

A chrome extension named ”HTTP Request Headers”, that had more than 500,000 downloads contained malicious code in it. It secretively visited ad-

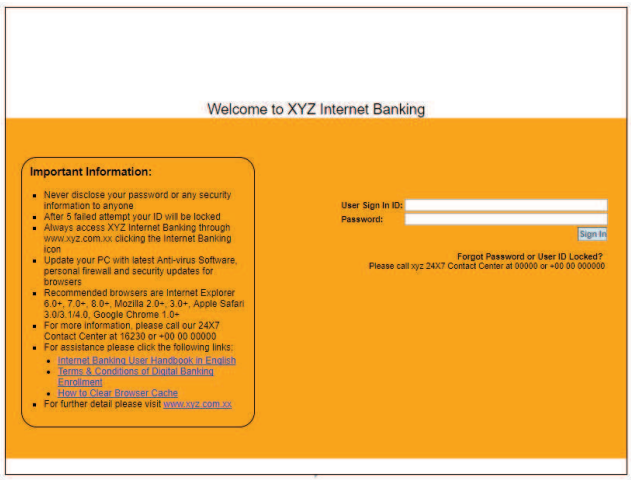

Figure 3.2: Sample login page without WebInject

vertising URLs generating a huge network traffic for the victims [115]. Banking malware can also be designed using Chrome extension. One example is ”Interface Online”, which was used by Brazilian attackers to collect banking credentials from corporate users [116].

Figure 3.1 shows a sample injection code. This code is custom designed for our sample login page. The injection code is a part of source code of a Google Chrome browser extension. The extension is configured in such a way that it only reacts when our desired webpage is visited. The page appears as it is when the extension is not enabled. Figure 3.2 shows the screenshot of the page without any injection.

When the extension is enabled it adds few component to the page and to users it appears to be coming from the actual server. Figure 3.3 shows the screenshot of the page after the injection. As we have mentioned earlier this extension does not perform any malicious activity except adding additional content to the page.

Malware codes are often obfuscated so that it becomes hard for the anti-malware solutions. Although the obfuscated appears to be very difficult to analyze, the outcome of the code remains same. Sample obfuscated code is shown in Figure 3.4 and Figure 3.5, that performs the same task as the code shown in Figure 3.1.

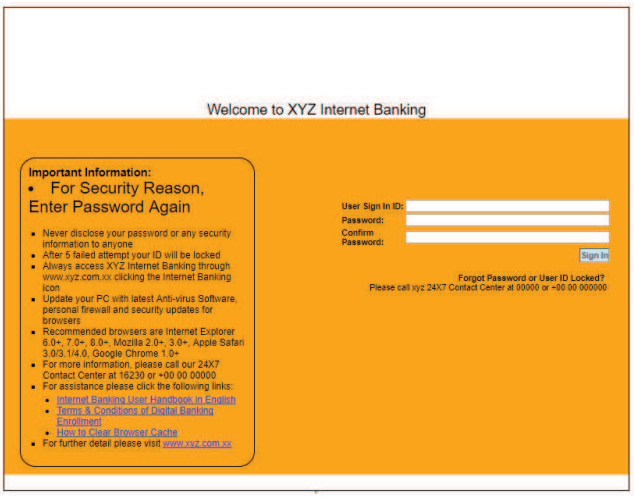

Figure 3.3: Sample login page with WebInject

```
eval(function(p,a,c,k,e,d){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?String.fr
omCharCode(c+29):c.toString(36))};if(!''.replace(/^/,String)){while(c--
){d[e(c)]=k[c]||e(c)}k=[function(e){return d[e]}];e=function(){return'\\w+'};c=1};while(c--
){if(k[c]){p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('$(\'3\').u(t(){0
5=3.s("v");0 c=3.w("z y x, r 9 n");5.j.q="o";5.p(c);0 8=3.A("M");8.L(5,8.K[2]);0
7=$(\'.b\').d(\'7\').g();0 f=7.O(\'a[e=l]\').d(\'6\').g();f.I(\'<6><4 k="b">C 9:</4><4 F="H"
G="1"><a e="l" E="" D="h i;" J="h i;" P="N" j="B:m;"
k=""></4></6>\')});',52,52,'var|||document|td|new_item|tr|table|list|Password|input|Lo
ginTextBold|textnode|parents|type|row|first|return|false|style|class|password|262px|A
gain|21px|appendChild|fontSize|Enter|createElement|function|ready|LI|createTextNode
|Reason|Security|For|getElementById|width|Confirm|onpaste|value|align|colspan|left|a
fter|oncopy|childNodes|insertBefore|cssParsedBox|fldPassword|find|name'.split('|'),0,{}))
```

Figure 3.4: Sample obfuscated code 1

3.3 Existing methods

As we can see from the example of malware explained above, the malware codes are very often obfuscated and their implementation strategy changes frequently over time but the behavior of malware remains similar regardless of their development environment. That is why behavior based malware analysis brought the attention of recent researchers. In DOM based XSS, malware injects additional contents on the fly inside the DOM of a web page. Criscione *et al.* [28] proposed a method for extracting WebInject signatures by inspecting DOM.

```
var _0xc7ab =
["\x4C\x49","\x63\x72\x65\x61\x74\x65\x45\x6C\x65\x6D\x65\x6E\x74","\x46\x6F\x72\x2
0\x53\x65\x63\x75\x72\x69\x74\x79\x20\x52\x65\x61\x73\x6F\x6E\x2C\x20\x45\x6E\x74\
x65\x72\x20\x50\x61\x73\x73\x77\x6F\x72\x64\x20\x41\x67\x61\x69\x6E","\x63\x72\x65
\x61\x74\x65\x54\x65\x78\x74\x4E\x6F\x64\x65","\x66\x6F\x6E\x74\x53\x69\x7A\x65","\
x73\x74\x79\x6C\x65","\x32\x31\x70\x78","\x61\x70\x70\x65\x6E\x64\x43\x68\x69\x6C\x
64","\x63\x73\x73\x50\x61\x72\x73\x65\x64\x42\x6F\x78","\x67\x65\x74\x45\x6C\x65\x6
D\x65\x6E\x74\x42\x79\x49\x64","\x63\x68\x69\x6C\x64\x4E\x6F\x64\x65\x73","\x69\x6E
\x73\x65\x72\x74\x42\x65\x66\x6F\x72\x65","\x66\x69\x72\x73\x74","\x74\x61\x62\x6C\
x65","\x70\x61\x72\x65\x6E\x74\x73","\x2E\x4C\x6F\x67\x69\x6E\x54\x65\x78\x74\x42\x
6F\x6C\x64","\x74\x72","\x69\x6E\x70\x75\x74\x5B\x74\x79\x70\x65\x3D\x70\x61\x73\x7
3\x77\x6F\x72\x64\x5D","\x66\x69\x6E\x64","\x3C\x74\x72\x3E\x3C\x74\x64\x20\x63\x6C
\x61\x73\x73\x3D\x22\x4C\x6F\x67\x69\x6E\x54\x65\x78\x74\x42\x6F\x6C\x64\x22\x3E\x
43\x6F\x6E\x66\x69\x72\x6D\x20\x50\x61\x73\x73\x77\x6F\x72\x64\x3A\x3C\x2F\x74\x6
4\x3E\x3C\x74\x64\x20\x61\x6C\x69\x67\x6E\x3D\x22\x6C\x65\x66\x74\x22\x20\x63\x6F
\x6C\x73\x70\x61\x6E\x3D\x22\x31\x22\x3E\x3C\x69\x6E\x70\x75\x74\x20\x74\x79\x70\x
65\x3D\x22\x70\x61\x73\x73\x77\x6F\x72\x64\x22\x20\x76\x61\x6C\x75\x65\x3D\x22\x2
2\x20\x6F\x6E\x70\x61\x73\x74\x65\x3D\x22\x72\x65\x74\x75\x72\x6E\x20\x66\x61\x6C\
x73\x65\x3B\x22\x20\x6F\x6E\x63\x6F\x70\x79\x3D\x22\x72\x65\x74\x75\x72\x6E\x20\x
66\x61\x6C\x73\x65\x3B\x22\x20\x6E\x61\x6D\x65\x3D\x22\x66\x6C\x64\x50\x61\x73\x7
3\x77\x6F\x72\x64\x22\x20\x73\x74\x79\x6C\x65\x3D\x22\x77\x69\x64\x74\x68\x3A\x32
\x36\x32\x70\x78\x3B\x22\x20\x63\x6C\x61\x73\x73\x3D\x22\x22\x3E\x3C\x2F\x74\x64\
x3E\x3C\x2F\x74\x72\x3E","\x61\x66\x74\x65\x72","\x72\x65\x61\x64\x79","\x64\x6F\x63
\x75\x6D\x65\x6E\x74"];$(_0xc7ab[22])[_0xc7ab[21]](function(){var
_0xd50dx1=document[_0xc7ab[1]](_0xc7ab[0]);var
_0xd50dx2=document[_0xc7ab[3]](_0xc7ab[2]);_0xd50dx1[_0xc7ab[5]][_0xc7ab[4]]=
_0xc7ab[6];_0xd50dx1[_0xc7ab[7]](_0xd50dx2);var
_0xd50dx3=document[_0xc7ab[9]](_0xc7ab[8]);_0xd50dx3[_0xc7ab[11]](_0xd50dx1,_0xd50
dx3[_0xc7ab[10]][2]);var
_0xd50dx4=$(_0xc7ab[15])[_0xc7ab[14]](_0xc7ab[13])[_0xc7ab[12]]();var
_0xd50dx5=_0xd50dx4[_0xc7ab[18]](_0xc7ab[17])[_0xc7ab[14]](_0xc7ab[16])[_0xc7ab[12]]
();_0xd50dx5[_0xc7ab[20]](_0xc7ab[19])})
```

Figure 3.5: Sample obfuscated code 2

Continella *et al.* [30] also used the DOM comparison to find the changes done by malware. Another client-side solution is to integrate the solution to the browser [31]. Sebastian et al. modified the source code for Chromium browser to use it for identifying DOMbased XSS issues. Stock *et al.* [32] worked on top Alexa websites and based on their defined matrices, they find the website that contains at least one XSS vulnerabilities. Fattori *et al.* [33] used the general interactions between benign programs with the operating system to make a behavioral model.

Sandboxing mechanism is used to ensure secure browsing experience by limiting permission of JavaScript and other processes. Same-origin policy also restricts instances of one process to use resources of its own origin only [25]. Even with the presence of these techniques, XSS can still attack user and steal valuable information. This can be done by tricking user to click malicious link and downloads malicious JavaScript code into trusted site and execute it inside it. Which allows that malicious code to use the resources of the trusted site and collect the valuable information.

A custom firewall system was proposed by Kirda *et al.* [25] for the client side which will decide whether a web page request from browser is secured or not based on some rules. User can make their own custom rule depending on their needs. A.K. Dalai *et al.* [12] proposed a server side solution based on some predefined criteria which will filter the data before executing. They tested using 230 attack vectors. Some of them are not functioning due to the change of browser policy and functions. They tested the algorithm using five different browsers. Their solution introduces a little overhead in terms of execution time. The papers [13, 26] used the genetic algorithm to generate suitable XSS from an initial random XSS by applying crossover and mutation and tried to find whether any path in the webpage executes that code.

3.4 Challenges

Ensuring a safe browsing experience for everyone is a challenging task. Not every computers or networks is always very secured and often get infected in various ways, i.e malicious emails, phishing websites, contagious web links and sometimes via physical devices. Another challenge is to deal with various implementation methods of malware and also to identify new and unknown malware. A machine learning based server side solution will resolve this problem.

The performance of a machine learning based solution is highly dependent on a good training model. A model is trained with an existing labelled dataset. Cyber threats detection data sets are imbalanced as the number of malicious activities in a certain domain is significantly low compared to the number of normal activities. Consider the dataset information presented in the table 3.1,

Table 3.1: Dataset information

Datset Name	Clean samples	Infected samples	Imbalance Ratio
Drebin	123453	5560	22.20
Credit card	284315	492	577.88
NSL-KDD (subset)	77054	737	104.55
Abalone 19	4142	32	129.44

which shows the number of clean and infected samples in some datasets. NSL-KDD dataset listed several types of attacks. We used a subset of this dataset by selecting all samples from one target attack and all normal samples. To train a good model with imbalanced dataset we need to design classifiers which can perform well with imbalanced dataset.

Various techniques are used to deal with imbalanced data classification. Two most popular data level methods are undersampling and oversampling. Undersampling technique removes a large portion of data from majority classes which causes loss of information and results in poor classification for majority class. Oversampling creates synthetic data for minority classes which cause over-fitting problem and in some cases it is not suitable to generate synthetic data. We need to design classifying technique for imbalanced data, that will overcome some of the problems of existing technique and provide better classification.

3.5 Proposed approach

Signature-based detection methods fail to keep up with the constantly evolving new threats. Machine learning based detection has drawn more attention of researchers due to its capability of detecting new and modified attacks based on previous attack's behaviour. Some of the existing solutions are designed for client side and all the users have to install it in their system, which is a challenging task. In addition, various platforms and tools are used by individuals, so different solutions needed to be designed.

Existing server side solution often focuses on sanitizing and filtering the inputs. It will fail to detect obfuscated and hidden scripts. We propose a server-

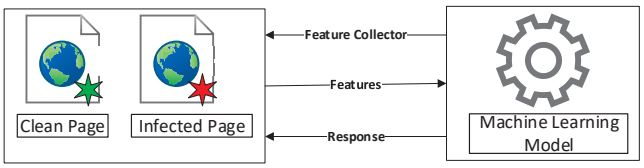

Figure 3.6: Basic architecture of our proposed system

side solution using a machine learning approach to detect threats caused by various cyber attacks. Unlike other techniques, our method collects features of a Document Object Model (DOM) and classifies it with the help of a pre-trained model. We experiment with both real and simulated banking environment. After analyzing the DOM in the client's browser of a website to find the features and send it back to the server. A pre-trained machine learning model will help to classify the page based on the collected features.

Figure 3.6 shows basic architecture of our proposed system. A feature collector using JavaScript is designed and it will be sent along with the actual webpage to collect features. We designed feature extractor as a browser extension for google chrome which analyzes the DOM to collect features. We have collected following information from the DOM of a webpage: total number of forms in the page, total number of anchors, number of saved cookies, character encoding method, server domain name, count of embedded elements, last modified date, total number of images, total number of links and total number of scripts.

Two google chrome extensions are designed for our experiment. One extension is used to inject some contents into the DOM of a webpage simulating the injection of a malware attack. Another extension is used to collect the above-mentioned features. The feature collector uses library functions JavaScript to collect the information. If the user's browser supports JavaScript execution, the feature collector does not need any further permission from the client to collect the features for our experiment.

After receiving the feature response, server classifies the current page. If server detects unusual content in the web page then it will send a negative

response to the client, otherwise it will give a positive response indicating that the page is safe to browse.

Our proposed approach is designed for server side and it will be easily reachable to all the users. Irrespective of which platform is used, our approach will still be able to perform well as it doesn't need to be deployed in client's device. Our proposed method will be able to deal with different implementation of malware also, as it does not analyze the source code, rather than it looks for behaviour or the features that have been introduced by an injection. The activities of an WebInject remains almost similar no matter how they are implemented. Selecting the features carefully and designing a good behavioral model will ensure higher detection rate.

3.6 Preliminary results

We implemented the proposed approach by simulating a local client server setup. A simulated banking environment is created resembling a real life banking website. We also used three different banks' live webpage to collect features.

A customized browser extension is used to simulate web injection. We collected features from 793 web-pages and among them 693 pages were clean and rest 100 pages contain injection. Like the real world situation, we tested our system with imbalanced dataset where number of infected instances are lower than number of clean instances. We also evaluated our system with a reduced dataset containing 307 clean and 100 infected instances. Classifiers based on different approaches are used to identify the one suitable for our dataset. Classification accuracy of classifiers are computed using 10-fold cross validation technique. We apply several mainstream classifiers implemented in WEKA. The classifiers are:

- **LibSVM:** We used Support Vector Machines (SVM) with two kernel variation (radial and polynomial).
- **Random Forest classifier:** Random forest classifier is a combination of several decision tree classifiers

- **Bayes Network:** Two Bayesian network based classifiers (BayesNet and NaiveBayes) are used in our experiment.
- **Bagging:** Bagging works by combining multiple predictors. Bagging can be used for both regression and classification. In case of classification, majority voting is used to determine final outcome.
- **IBK.** This is an instance based K-nearest neighbour classifier.
- **K-star:** This is also an instance based classifier which use entropy-based distance function for similarity measurement.

Results of our experiment are summarized in Tables 3.3 to table 3.5.

Figure 3.7: Accuracy of various classifiers

Figure 3.7 shows the accuracy of the classifiers for both full and reduced dataset. We observe that the Random Forest provided highest accuracy for the full dataset and Bayes Net provided highest accuracy for the reduced dataset. There is scope to work on the feature selection and that will impact on the classification process.

Although overall accuracy of all the classification method obtains a satisfactory result, but the accuracy of minority class is low when the imbalanced ratio is high. Table 3.2 shows the classwise and overall accuracy for both full and reduced datasets. In full dataset where imbalance ratio is high, the accuracy of minority class is low. The imbalance ratio of reduced datasets is low and the performance on minority class is improved for all the classifiers. In

Table 3.2: Classwise and overall accuracy

Classifier	Full Dataset			Reduced Dataset		
	Majority	Minority	Overall	Majority	Minority	Overall
BayesNet	97.84	44.00	91.04	96.09	80.00	92.13
Random Forest	98.70	83.00	96.72	97.72	96.00	97.29
Bagging	98.70	78.00	96.09	95.76	89.00	94.10
Naive Bayes	100.00	0.00	87.34	100.00	0.00	75.43
SVM Radial	99.86	23.00	90.16	99.35	26.00	81.32
SVM Polynomial	94.23	76.00	91.93	96.09	89.00	94.35
Kstar	96.97	79.00	94.70	97.72	94.00	96.80
IBK	97.84	82.00	95.84	98.04	91.00	96.31

cyber security domain, the imbalance ratio will be very high as the number of samples for threat will be very low compared to the number of normal samples. The preliminary results indicate that the mainstream classifier will not be suitable for detection of threats. We need to design a classification method for the imbalanced dataset to apply in cyber threat detection.

Table 3.3: Precision, Recall and F-Measure for clean and infected pages

Classifier	Clean Page			Infected Page		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
BayesNet	0.924	0.978	0.950	0.746	0.440	0.553
Random Forest	0.976	0.987	0.981	0.902	0.830	0.865
Bagging	0.969	0.987	0.978	0.897	0.780	0.834
Naive Bayes	0.874	1.000	0.933	0.000	0.000	0.000
SVM Radial	0.900	0.999	0.947	0.958	0.230	0.371
SVM Polynomial	0.965	0.942	0.953	0.655	0.760	0.704
Kstar	0.970	0.970	0.970	0.790	0.790	0.790
IBK	0.845	0.820	0.832	0.974	0.978	0.976

Table 3.3 presents the precision, recall and F-measure for the clean and infected webpages.

We calculated cost of our classification using cost matrix shown in Table 3.4. As the dataset is imbalanced and one class have more signifance over

Table 3.4: Cost matrix

	prediction $y = 1$	prediction $y = 0$
label $h(x) = 1$	$C_{1,1} = 0$	$C_{0,1} = 5$
label $h(x) = 0$	$C_{1,0} = 1$	$C_{0,0} = 0$

*0 = Clean Page and 1 = Infected Page

Table 3.5: Weighted measure

Classifier	Cost (Full dataset)	Cost (Reduced dataset)
BayesNet	295	112
Random Forest	94	27
Bagging	119	68
Naive Bayes	500	500
SVM Radial	386	372
SVM Polynomial	160	67
Kstar	126	37
IBK	105	51

another one so we assigned different costs for various prediction. We assumed right prediction for both clean and infected data does not add any cost and false prediction of an infected page is assigned five times more cost than false prediction of a clean page. Table 3.5 shows the cost of our prediction for both datasets. As we can see even though some of the classifiers show better accuracy than others, but their cost is higher. Therefore, depending on the need we have to find the most suitable classifier for imbalanced data.

3.7 Conclusion

As we discussed in this chapter that, malware codes can be very obfuscated and hidden. Various implementation methods are also used to design these attacks, which makes it difficult for a signature based detection method to identify new and unexplored malware, so machine learning based server-side solution is very suitable for this problem domain and overcomes many limitations of signature based solutions. We also emphasized on designing a server side solution rather than a client side solution, which is a one stop solution for all the users of a certain organization.

As the number of malicious activity is significantly low compared to the normal activities which creates imbalanced data for our training model. We observed that mainstream classification methods failed to provide good accuracy for the minority classes, which is critical in cyber threat detection. To obtain a better detection model we propose two different supervised data classification techniques that can deal with imbalanced data. Next two chapters

present those classification methods.

Chapter 4

Piecewise linear classifier for imbalanced data

In this chapter we introduce a cost-sensitive piecewise linear classifier to solve the supervised data classification problems in imbalanced datasets. We define the imbalance ratio for a given dataset and then depending on this ratio assign a cost to each class in the dataset. Finally, using costs for each class we formulate the problem of finding piecewise linear boundaries between classes as a constrained optimization problem. We also discuss an algorithm for solving the optimization problems.

4.1 Introduction

The success of the supervised learning heavily depends on the class distribution of the training set. A training set with balanced classes leads into a good performance of a classification algorithm in the testing step. However, if in a training set classes are not balanced then large classes may dominate the learning model resulting in poor classification accuracy for small classes. A dataset is categorized as imbalanced when the number of instances in some classes is significantly larger than that of in some other classes. In such datasets mainstream classifiers fail to accurately classify observations from minority classes. This is due to the fact that for these classifiers in the training step large classes dominate the learning model which results in poor classification accuracy for minority classes. Therefore, special approaches are needed to

design classifiers to solve the supervised classification problems in imbalanced datasets.

The development of classifiers for learning from datasets with imbalanced class distributions have attracted increasing attention over the last decade. In most real-world datasets class distributions are not uniform and in many such datasets some classes have significantly less data points than others. In some applications such classes were considered as outliers and they removed from the training set to design a better classification model. However, there are many other applications where such classes cannot be ignored. Supervised and unsupervised classification problems for fraud detection and detection of malicious emails are among such applications. Fraud-related incidents and malicious emails occur in a limited number of times. For these reasons such incidents and emails constitute minority classes and their detection becomes a classification problem in imbalanced datasets. Mainstream classifiers are heavily biased by the majority classes and they are not performing well in classifying minority classes in such datasets.

In this chapter, a new piecewise linear classifier, PWLCI, is introduced to solve supervised classification problems in imbalanced datasets. A new classifier is an extension of a piecewise linear classifier for general datasets. This classifier constructs a boundary between classes incrementally starting from one linear function (hyperplane) and adds more linear functions (hyperplanes) at each iteration of the incremental algorithm. The problem of finding of boundaries between classes is modeled as an optimization problem where the objective function is the misclassification error function and constraints are formulated using classification errors for minority classes. The optimization problem is solved applying the penalty function method. The proposed classifier is tested using real-world imbalanced datasets and compared with several mainstream classifiers as well as with classifiers developed specifically for imbalanced data sets.

4.2 Piecewise linear classifier

In this section we describe the piecewise linear classifier for the general supervised data classification problems. First we introduce some notions which will

be used in the rest of the thesis.

In what follows $\mathbb{R}^n$ is the n-dimensional space of vectors $u = (u_1, \ldots, u_n)$, $\langle u, v\rangle = \sum\limits_{i=1}^{n} u_i v_i$ is the inner product of vectors $u, v \in \mathbb{R}^n$ and $\|u\| = (\sum\limits_{i=1}^{n} u_i^2)^{1/2}$ is the associated Euclidean norm in $\mathbb{R}^n$. For given $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ a hyperplane $H(a, b)$ is defined as follows:

$$H(a, b) = \Big\{x \in \mathbb{R}^n : \quad \langle a, x\rangle + b = 0\Big\}.$$

The hyperplane is the line in the 2-dimensional case and it is the plane in the 3-dimensional case.

Let A and B be given disjoint sets containing $m \geq 1$ and $n \geq 1$ number of d-dimensional vectors, respectively:

$$A = \{a^1, \ldots, a^m\},\ a^i \in \mathbb{R}^d,\ i = 1, \ldots, m,$$
$$B = \{b^1, \ldots, b^n\},\ b^j \in \mathbb{R}^d,\ j = 1, \ldots, n.$$

Assume that these sets can be represented as a union of the finite number of sets such that

$$A = \bigcup_{i=1}^{p} A_i, \quad B = \bigcup_{j=1}^{q} B_j \tag{4.2.1}$$

and

$$\operatorname{conv} A_i \cap \operatorname{conv} B_j = \emptyset,\ i = 1, \ldots, p,\ j = 1, \ldots, q. \tag{4.2.2}$$

Here "conv" stands for a convex hull of a set. Let $I = \{1, \ldots, p\}$ and $J = \{1, \ldots, q\}$.

Note that any finite points set has the representation (4.2.1) and any two finite point disjoint sets satisfy the condition (4.2.2). Indeed, if each sets A_i and B_j are singletons containing only one point from corresponding sets then we will have these representations. However, we expect that $p \ll m$ and $q \ll n$.

One possible representation of the sets A and B is illustrated in Figure 4.1. Here both sets A and B are represented as the union of three subsets and convex hulls of all pairs of subsets from opposite classes are disjoint.

Since $\operatorname{conv} A_i \cap \operatorname{conv} B_j = \emptyset$ for sets A_i and B_j, $i \in I,\ j \in J$, there exists a hyperplane (x^{ij}, y_{ij}) with $x^{ij} \in \mathbb{R}^d,\ y_{ij} \in \mathbb{R}$ separating these two sets. This

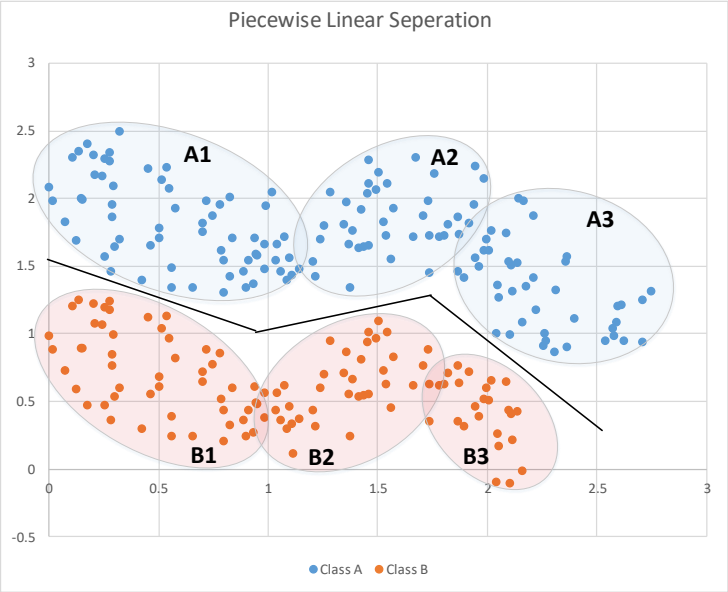

Figure 4.1: Representation of sets A and B and their piecewise linear separation.

means that

$$\langle x^{ij}, a\rangle + y_{ij} < 0 \quad \forall a \in A_i, \tag{4.2.3}$$

and

$$\langle x^{ij}, b\rangle + y_{ij} > 0 \quad \forall b \in B_j. \tag{4.2.4}$$

Using some transformations and keeping the same notations one can rewrite (4.2.3) and (4.2.4), respectively, as

$$\langle x^{ij}, a\rangle + y_{ij} \leq -1 \quad \forall a \in A_i,$$

and

$$\langle x^{ij}, b\rangle + y_{ij} \geq 1 \quad \forall b \in B_j.$$

This is proved in the next proposition.

Definition 1. Let A and B be sets in $\mathbb{R}^n$ both containing finite number of points. These sets are piecewise linearly separable if there exist hyperplanes $\{(x^{ij}, y_{ij})\},\ j \in J_i,\ i \in I$ such that:

$$\min_{j \in J_i} \left\{ \langle x^{ij}, a\rangle - y_{ij} \right\} < 0 \quad \text{for all } i \in I and \quad a \in A$$

and for each $b \in B$ there exists at least one $i \in I$ such that

$$\min_{j \in J_i} \Big\{ \langle x^{ij}, b \rangle - y_{ij} \Big\} > 0.$$

Proposition 1. *The sets A and B are piecewise linearly separable if and only if there exists a set of hyperplanes $\{x^j, y_j\}$ with $x^j \in \mathbb{R}^n,\ y_j \in \mathbb{R}^1,\ j \in J$ and a partition $J^r = \{J_1, \ldots, J_r\}$ of the set J such that*

1)

$$\min_{j \in J_i} \big\{ \langle x^j, a \rangle - y_j \big\} \leq -1 \quad \text{for all} \quad i \in I \quad \text{and} \quad a \in A;$$

2) for any $b \in B$ there exists at least one $i \in I$ such that

$$\min_{j \in J_i} \big\{ \langle x^j, b \rangle - y_j \big\} \geq 1.$$

Proof: Sufficiency is straightforward.

Necessity. Since A and B are piecewise linearly separable there exists a set of hyperplanes $\{\bar{x}^j, \bar{y}_j\}$ with $\bar{x}^j \in \mathbb{R}^n,\ \bar{y}_j \in \mathbb{R}^1,\ j \in J$, a partition J^r of the set J and numbers $\delta_1 > 0,\ \delta_2 > 0$ such that

$$\max_{a \in A} \max_{i \in I} \min_{j \in J_i} \big\{ \langle \bar{x}^j, a \rangle - \bar{y}_j \big\} = -\delta_1$$

and

$$\min_{b \in B} \max_{i \in I} \min_{j \in J_i} \big\{ \langle \bar{x}^j, b \rangle - \bar{y}_j \big\} = \delta_2.$$

We put $\delta = \min\{\delta_1, \delta_2\} > 0$. Then we have

$$\max_{i \in I} \min_{j \in J_i} \big\{ \langle \bar{x}^j, a \rangle - \bar{y}_j \big\} \leq -\delta,\ \forall a \in A, \tag{4.2.5}$$

$$\max_{i \in I} \min_{j \in J_i} \big\{ \langle \bar{x}^j, b \rangle - \bar{y}_j \big\} \geq \delta,\ \forall b \in B. \tag{4.2.6}$$

We consider the new set of hyperplanes $\{x^j, y_j\}$ with $x^j \in \mathbb{R}^n,\ y_j \in \mathbb{R}^1,\ j \in J$, defined as follows:

$$x^j = \bar{x}^j / \delta,\ j \in J,$$

$$y^j = \bar{y}^j / \delta,\ j \in J.$$

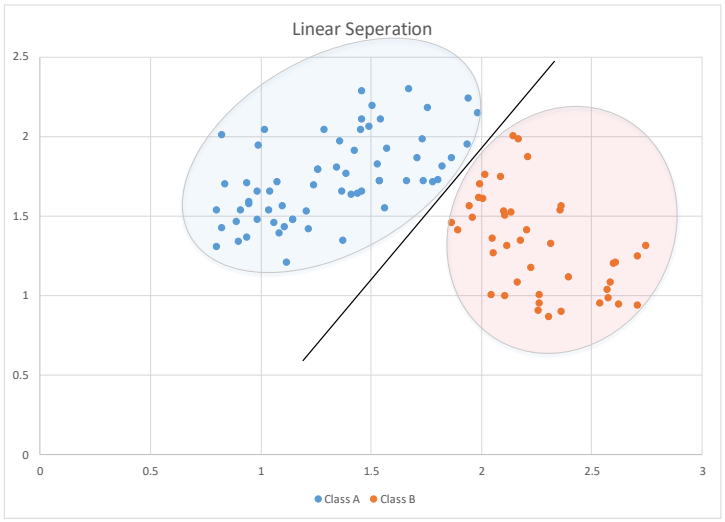

Figure 4.2: Linear separation of two sets.

Then it follows from (4.2.5) and (4.2.6) that

$$\max_{i\in I}\min_{j\in J_i}\left\{\langle x^j, a\rangle - y_j\right\} \le -1, \quad \forall a \in A,$$

$$\max_{i\in I}\min_{j\in J_i}\left\{\langle x^j, b\rangle - y_j\right\} \ge 1, \quad \forall b \in B,$$

which completes the proof. □

We obtain that the function

$$\varphi(x^{ij}, y_{ij}, c) = \langle x^{ij}, c\rangle + y_{ij}$$

separates the sets A_i and B_j. The linear separation of two finite point sets is depicted in Figure 4.2.

Take any set $B_j, j \in J$. Then the following function separates the set A from the set B_j:

$$\psi(x^j, y_j, c) = \min_{i\in I}\ \varphi(x^{ij}, y_{ij}, c).$$

Here $x^j = (x^{1j}, \ldots, x^{pj}) \in \mathbb{R}^{dp}$, $y_j = (y_{1j}, \ldots, y_{pj}) \in \mathbb{R}^p$. It is clear that

$$\psi(x^j, y_j, a) \le -1 \ \ \forall a \in A$$

and

$$\psi(x^j, y_j, b) \ge 1 \ \ \forall b \in B_j.$$

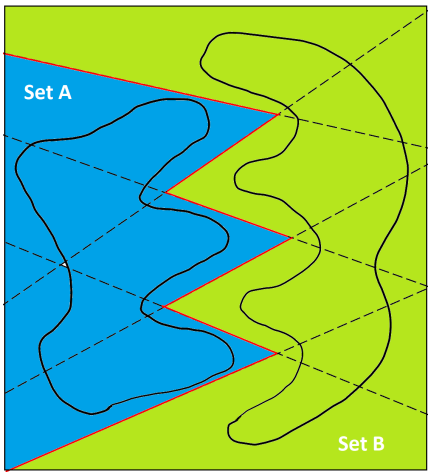

Figure 4.3: Max-min separation of two sets.

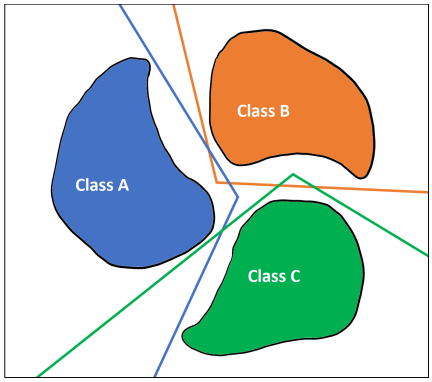

Figure 4.4: Max-min separation of two sets.

Then the piecewise linear function

$$f(x,y,c) = \max_{j\in J} \ \psi(x^j, y_j, c) = \max_{j\in J} \ \min_{i\in I} \varphi(x^{ij}, y_{ij}, c) \tag{4.2.7}$$

separates the sets A and B. Here $x = (x^1, \ldots, x^p) \in \mathbb{R}^{dpq}$, $y = (y_1, \ldots, y_q) \in \mathbb{R}^{pq}$. One example of piecewise linear separation of two finite point sets is illustrated in Figure 4.1.

Figure 4.3 presents a more complex max-min separation for two classes. Separation of three classes are illustrated in the Figure 4.4. It is proved in [117] that the sets A and B are piecewise linearly separable if and only if they are disjoint: $A \cap B = \emptyset$.

4.2.1 Error function

The error function is defined as

$$F_{pq}(x,y) = \frac{1}{|A|}\sum_{a\in A}\max\Big\{0, f(x,y,a)+1\Big\} + \frac{1}{|B|}\sum_{b\in B}\max\Big\{0, -f(x,y,b)+1\Big\}. \tag{4.2.8}$$

Equivalently, it can be rewritten as follows

$$F_{pq}(x,y) = F^1_{pq}(x,y) + F^2_{pq}(x,y), \tag{4.2.9}$$

where

$$F^1_{pq}(x,y) = \frac{1}{|A|}\sum_{a\in A}\max\left\{0, \max_{j\in J}\min_{i\in I}\left[\langle x^{ij}, a\rangle - y_{ij}\right]+1\right\}, \tag{4.2.10}$$

$$F^2_{pq}(x,y) = \frac{1}{|B|}\sum_{b\in B}\max\left\{0, -\max_{j\in J}\min_{i\in I}\left[\langle x^{ij}, b\rangle - y_{ij}\right]+1\right\}. \tag{4.2.11}$$

It is proved in [117] that the sets A and B are piecewise linearly separable if and only if there exists a set of hyperplanes $\{x^{ij}, y_{ij}\}, j \in J,\ i \in I$ such that $F_{pq}(x,y) = 0$. Moreover, $x = 0 \in \mathbb{R}^{dpq}$ cannot be an optimal solution. The error function (4.2.8) is nonconvex and if the sets A and B are piecewise linearly separable with the given number of hyperplanes, then the global minimum of this function is $F_{pq}(x^*, y_*) = 0$ and the global minimizer is not always unique.

One can see that the error function $F_{pq}(x,y)$ is represented as the sum of two nonconvex piecewise linear functions. Therefore, this function is also nonconvex piecewise linear. The error function has many local minimizers and the number of its local minimizers increases drastically as the number of hyperplanes increases. The function $F_{pq}(x,y)$ is nonsmooth and it is locally Lipschitz function. The problem of minimization of this function is a global optimization problem. However, global optimization methods cannot be applied to solve this problem since it has many decision variables.

The problem of the piecewise linear separability is reduced to the following minimization problem:

$$\text{minimize}\ \ F_{pq}(x,y)\ \ \text{subject to}\ \ (x,y) \in \mathbb{R}^{dpq} \times \mathbb{R}^{pq}. \tag{4.2.12}$$

Algorithms for solving Problem (4.2.12) are developed in [117, 118, 119]. Algorithms introduced in [118, 119] construct piecewise linear boundary between classes (sets) incrementally starting from one linear function that is starting with the linear separation. At each iteration, the subset of data points which do not contribute to boundaries between classes are identified and removed from further consideration. Such an approach allows one to significantly reduce the training time and solve the problem (4.2.12) efficiently. Furthermore, the incremental approach allows to calculate as many hyperplanes as necessary for the separation of classes. However, the maximum number of hyperplanes should be restricted to avoid the overfitting problem.

When there are two classes in the dataset it is sufficient to calculate only one piecewise linear function separating these two classes. To classify the new observation we compute the value of the piecewise linear function for this observation. If its value is negative then this observation is classified to the set a and it is classified to the set B otherwise.

The problem (4.2.12) can be applied to solve the supervised data classification problems in datasets containing more than two classes. In this case applying "the one versus others" strategy we consider $K > 2$ different separation problems. For each of them we solve the problem (4.2.12) and find the piecewise linear function f, defined in (4.2.7), separating the k-th class from all other classes. We will denote this function by f^k where the superscript k shows that this function separates the k-th class from the rest of the dataset where $k = 1, \ldots, K$. This means that for each class we find one piecewise linear function separating this class from other classes. To classify the new observation we calculate values of each of these piecewise linear functions for this observation and then we calculate the minimum of these values. The observation is classified to the class whose piecewise linear function provides this minimum.

4.3 Piecewise linear classifier for imbalanced datasets

In this section we modify the piecewise linear classifier described in the previous subsection to solve the supervised data classification problems in imbalanced

datasets. To design such a classifier we reformulate the problem (4.2.12) by adding constraints defined using misclassification errors for points from minority classes. Then the constrained optimization problem is reduced to the unconstrained problem by applying the penalty function method. The discrete gradient method from nonsmooth optimization is applied to solve the unconstrained optimization problem and to find parameters of the piecewise linear classifier.

Let D be a data set containing K classes: $C_1, \ldots, C_K$:

$$D = \bigcup_{k=1}^{K} C_k.$$

Without loss of generality assume that the first K_0, $1 \leq K_0 < K$ of these classes $C_1, \ldots, C_{K_0}$ are majority and the rest $K - K_0$ classes $C_{K_0+1}, \ldots, C_K$ are minority classes. We consider the case when $K_0 < K$, that is the data set A contains minority class(es).

General purposed classifiers fail to solve the supervised data classification problems in such data sets. These classifiers assign most of new observations into one of the majority classes and in this way achieve high overall classification accuracy, however they fail to classify new observations belonging to minority classes.

As mentioned above we compute the piecewise linear functions f^k, $k = 1, \ldots, K$ for each class by solving the problem (4.2.12). However, in imbalanced datasets we propose to compute these functions by solving different optimization problems for majority and minority classes. For majority classes we solve the unconstrained minimization problem (4.2.12) to find the function f^k. For minority classes $k = K_0 + 1, \ldots, K$ this problem is replaced by the constrained optimization problem by adding constraints defined using misclassification errors for points from minority classes. More precisely, this problem is

$$\begin{cases} \text{minimize} & F_{pq}(x, y) \\ \text{subject to} & f^k(x, y) \leq 0, \\ & (x, y) \in \mathbb{R}^{(n+1)pq}. \end{cases} \tag{4.3.1}$$

The constraint $f_k(x, y) \leq 0$ in problem (4.3.1) plays crucial role to increase importance of minority classes. Unconstrained minimization of the error function does not lead to finding of balanced boundaries between large and small classes. This leads to the perfect classification of instances from large classes and to failure in classification of instances from minority classes. The use of constraints for minority classes helps to find balanced piecewise linear boundaries between large and minority classes and to improve classification accuracy for minority classes.

Using the penalty function this problem is replaced by the following problem:

$$\text{minimize} \quad \bar{F}_{pq}(x, y) \quad \text{subject to} \quad (x, y) \in \mathbb{R}^{(n+1)pq}. \tag{4.3.2}$$

where

$$\bar{F}_{pq}(x, y) = F_{pq}(x, y) + \tau \sum_{k=K_0+1}^{K} \max\Big\{0, f^k(x, y)\Big\}.$$

Here $\tau > 0$ is the penalty parameter.

The objective functions F_{pq} and $\bar{F}_{pq}$ are nonconvex nonsmooth. More precisely, both functions are nonconvex piecewise linear functions. Usually, the Clarke subdifferential is used to design algorithms for minimizing such functions (see, for example, [120] for the definition of the Clarke subdifferential). However, the calculation of subgradients of these functions is not always an easy task as the calculus exists only in the form of inclusions. Therefore, we apply the discrete gradient method [121] to solve problems (4.2.12) and (4.3.1). This method uses the values of the objective function to approximate its subgradients.

4.3.1 Discrete gradient method

In this subsection we briefly describe the discrete gradient method (DGM) which is applied to solve problems (4.2.12) and (4.3.1). More details can be found in [121] (see, also [122, 120, 123]). The DGM can be considered as a version of bundle methods. In contrast with the bundle methods, which require the computation of a single subgradient of the objective function at each trial point, the DGM approximates subgradients by discrete gradients using function values only. Similarly to bundle methods, discrete gradients computed at

a given point are gathered into a bundle to compute search directions.

Let

$$G = \Big\{ e \in \mathbb{R}^n : e = (e_1, \ldots, e_n), |e_j| = 1,\ j = 1, \ldots, n \Big\}$$

be a set of all vertices of the unit hypercube in $\mathbb{R}^n$. For $e \in G$ and $\alpha \in (0, 1]$ define the sequence of n vectors

$$e^j = e^j(\alpha) = (\alpha e_1, \alpha^2 e_2, \ldots, \alpha^j e_j, 0, \ldots, 0),\ j = 1, \ldots, n.$$

Take any $d \in \mathbb{R}^n$ such that $\|d\| = 1$ and compute $i \equiv i(d) = \text{argmax}\{|d_j|,\ j = 1, \ldots, n\}$. Note that the index i depends on the direction d. For $x \in \mathbb{R}^n$ and $\lambda > 0$, consider the points

$$x_0 = x + \lambda d, \qquad x_j = x_0 + \lambda e^j(\alpha), \qquad j = 1, \ldots, n.$$

Definition 2. The discrete gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ at $x \in \mathbb{R}^n$ is the vector $\Gamma^i(x, d, e, \lambda, \alpha) = (\Gamma^i_1, \ldots, \Gamma^i_n) \in \mathbb{R}^n$ with the following coordinates:

$$\Gamma^i_j = [\lambda \alpha^j e_j)]^{-1} \Big[f(x_j) - f(x_{j-1}) \Big], \qquad j = 1, \ldots, n, \quad j \neq i,$$

$$\Gamma^i_i = (\lambda d_i)^{-1} \Big[f(x + \lambda d) - f(x) - \lambda \sum_{j=1, j \neq i}^{n} \Gamma^i_j d_j \Big].$$

It is proved in [122], under semismoothness and quasidifferentiability assumptions on the function f, that the set

$$D_0(x, \lambda) = \text{cl}\,\text{conv} \Big\{ v \in \mathbb{R}^n : v = \Gamma^i(x, d, e, \lambda, \alpha), \|d\| = 1,\ e \in G,\ \alpha \in (0, 1] \Big\}$$

is an approximation to the subdifferential $\partial^\circ f(x)$ for sufficiently small $\lambda > 0$ in the sense of the Hausdorff distance. The computation of the whole set $D_0(x, \lambda)$ is not easy, and therefore, in the DGM only a few discrete gradients are used to calculate the search direction.

Next we describe the direction finding procedure. Let us denote by l the index of the subiteration, by k the index of the outer iteration and by s the index of the inner iteration in the direction finding procedure. In what follows we use only the iteration counter l whenever possible without confusion. At every iteration k_s, we first compute the discrete gradient $v_1 = \Gamma^i(x, d_1, e, \lambda, \alpha)$

with respect to an initial direction $\|d_1\| = 1$, set $\overline{D}_1(x) = \{v_1\}$ and $l = 1$. Then we compute the vector

$$w_l = \operatorname*{argmin}_{w \in \overline{D}_l(x)} \|w\|^2.$$

If $\|w_l\| < \delta$ for a given tolerance $\delta > 0$ then the point x is accepted as an approximate stationary point and we go to the next outer iteration of the DGM. Otherwise, we compute another search direction $d_{l+1} = -\|w_l\|^{-1} w_l$. If d_{l+1} is the descent direction satisfying the condition

$$f(x + \lambda d_{l+1}) - f(x) \leq -c_1 \lambda \|w_l\|,$$

with the given numbers $c_1 \in (0, 1)$ and $\lambda > 0$, then we set $d_{k_s} = d_{l+1}$, $v_{k_s} = w_l$ and stop the direction finding procedure. Otherwise, we compute another discrete gradient $v_{l+1} = \Gamma^i(x, d_{l+1}, e, z, \lambda, \alpha)$ in the direction d_{l+1}, update the bundle of discrete gradients $\bar{D}_{l+1}(x) = \text{conv}\{\bar{D}_l(x) \cup \{v_{l+1}\}\}$ and continue the direction finding procedure with $l = l + 1$. It is proved in [121] that the direction finding procedure finitely converges.

When the descent direction d_{k_s} has been found, we compute the next (inner) iteration point $x_{k_{s+1}} = x_{k_s} + t_{k_s} d_{k_s}$, where the step size t_{k_s} is defined as

$$t_{k_s} = \text{argmax}\left\{t \geq 0 : f(x_{k_s} + t d_{k_s}) - f(x_{k_s}) \leq -c_2 t \|v_{k_s}\|\right\},$$

with given $c_2 \in (0, c_1]$.

DGM is globally convergent for locally Lipschitz continuous functions under assumption that the set of discrete gradients uniformly approximate the subdifferential [122, 121].

4.3.2 The classification algorithm

The proposed classification algorithm constructs boundary between classes incrementally and it starts with one linear function that is with the linear separation. This algorithm is a modification of the piecewise linear classifier introduced [118]. Here we consider the case when the number of linear functions under each minimum function is the same. Such a choice of the number of linear functions is justified according to the definition piecewise linear separa-

bility given above.

Algorithm 1 Piecewise linear classifier for imbalanced data.

Input: Data set D with two classes A and B. The maximum number m_M of minimum functions under maximum, the maximum number m_L of linear functions under each minimum, the tolerances $\varepsilon_1,\ \varepsilon_2 > 0$.
Output: A piecewise linear boundary between classes.

Step 1: Select $x^{11} \in \mathbb{R}^n,\ y_{11} \in \mathbb{R}$. Calculate the value of the objective function f^1, defined in (4.2.7) when $k = 1$, at the point $(\bar{x}^{11}, \bar{y}_{11})$. Set $I = \{1\},\ J = \{1\}$ and $k := 1$.

Step 2: Solve the problem (4.3.2) starting from the point $(x^{ij}, y_{ij}),\ i \in I,\ j \in J$. Assume that $\bar{x}^{ij}, \bar{y}_{ij},\ i \in I,\ j \in J$ is the solution.

Step 3: Calculate the value of the objective f^{k+1} defined in (4.2.7) at the point $(\bar{x}^{ij}, \bar{y}_{ij}),\ i \in I,\ j \in J$

Step 4: (*The first stopping criterion*). If $f^{k+1} \leq \varepsilon_1$ then **STOP**. The point $(\bar{x}^{ij}, \bar{y}_{ij}),\ i \in I,\ j \in J$ is the solution.

Step 5: (*The second stopping criterion*). If $f^k - f^{k+1} \leq \varepsilon_2$ and $|I| > 1$ then **STOP**. The point $(\bar{x}^{ij}, \bar{y}_{ij}),\ i \in I,\ j \in J$ is the solution.

Step 6: If $|J| < m_L$, then set $J = \{J\} \cup \{j+1\}$ and

$$x^{tp} = \bar{x}^{tp},\ y_{tp} = \bar{y}_{tp},\quad t = 1, \ldots, i,\ p = 1, \ldots, j,\ x^{i,j+1} = \bar{x}^{ij},\quad y_{i,j+1} = \bar{y}_{ij}.$$

Go to Step 2. Otherwise go to Step 7.

Step 7: If $|I| < m_M$, then set $I = \{I\} \cup \{i+1\}$ and

$$x^{tp} = \bar{x}^{tp},\ y_{tp} = \bar{y}_{tp},\quad t = 1, \ldots, i,\ p = 1, \ldots, j,$$

$$x^{i+1,j} = \bar{x}^{ij},\quad y_{i+1,j} = \bar{y}_{ij},\ j = 1, \ldots, m_L.$$

Go to Step 2. Otherwise go to Step 8.

Step 8: (*The third stopping criterion*). If $|I| = m_M$ and $|J| = m_L$, then **STOP**. The point $(\bar{x}^{ij}, \bar{y}_{ij}),\ i \in I,\ j \in J$ is accepted as the solution.

The necessary explanation on Algorithm 1 follows. This algorithm computes the piecewise linear boundary between classes A and B with m_M and m_L number of linear functions. This piecewise linear boundary is represented as a maximum of minimum of linear functions and the number minimum functions under maximum function is m_M and the number of linear under each minimum function is m_L. The algorithm selects any starting point and in Step 2 solves

the linear separability problem. This problem is convex optimization problem and the algorithm finds its global minimum.

If the sets A and B are not linearly separable then we first compute one minimum function to separate these sets. This function is computed in Steps 2 and 6. In these steps we add one linear function under minimum function until the number of linear functions reaches the number m_L. The starting point to find this minimum function is defined using the solution from the previous iteration of the algorithm.

If the sets A and B are not separable with one minimum function then in Step 6 we add one minimum function under the maximum. This step is repeated until the number of minimum functions under maximum reaches the number m_M. The starting point for finding piecewise linear functions is updated using the solution from the previous iteration of the algorithm. Such definition of the starting points allows one to address nonconvexity of the problem (4.3.2).

Algorithm 1 has three stopping criteria. The first stopping criterion is given in Step 4. This condition means that the sets A and B are separable with current number of linear functions. The second stopping criterion is described in Step 5. This criterion means that adding more linear functions does not significantly improve the separation of the sets A and B and adding more linear functions will lead overfitting problem. The third criterion is given in Step 8. This condition means that the algorithm uses the maximum number of linear functions. Such a condition is important as it allows to avoid the overfitting problem.

4.3.3 Implementation of the algorithm

Algorithm 1 contains several parameters and tolerances to be defined before its execution. Main parameter is the number m_L of linear functions under each minimum and the number m_M of minimum functions under each maximum. We take $m_L = 2$ in all cases. The maximum number of minimum functions is $m_M = 4$, however the algorithm may stop before reaching this number. This happens when one of stopping criteria in Steps 4 and 5 satisfies.

We select the tolerances as $\varepsilon_1, \varepsilon_2 = 0.001$. The problem (4.3.2) is solved by applying the discrete gradient method. Details of implementation of this

method can be found in [121].

In case of binary classification problem only one piecewise linear function separating two classes is computed. For multi-class classification problem one vs all others approach is applied. It means that in this case we calculate one piecewise linear function separating each class from all others that is in the multi-class case the number of piecewise linear functions is the number of classes.

Different classification rules are applied for binary and multi-class classification problems. In the binary classification problems one piecewise linear function is used to separate classes and any new observation is classified according to the sign of the value of the piecewise linear function. In the multi-class case the number m of piecewise linear functions equals to the number of classes that is for any new observations we get m values and this observations is assigned to the class with smallest value.

The proposed algorithm has been implemented in Fortran 95 using *gfortran* compiler, and numerical experiments have been executed on a 2.7 GHz Intel Core i5 computer with RAM 8GB working under Windows 10.

4.4 Conclusion

In this chapter, a new classifier is introduced to solve the supervised data classification problems in imbalanced datasets. It is designed using the piecewise linear classifier. The problem of calculating of piecewise linear boundaries between classes is formulated as a constrained optimization problem where the objective function is the classification error function and constraints are formulated using misclassification errors of points from minority classes. The discrete gradient method is applied to solve the constrained optimization problem and to find piecewise linear functions separating classes. The proposed classifier belongs to the class of cost-sensitive classifiers for imbalanced datasets.

Chapter 5

Hybrid classifiers for imbalanced data

In this chapter, the problem of the supervised data classification in imbalanced datasets is discussed and an algorithm based on the combination of the clustering, undersampling and supervised data classification techniques is developed to solve it. The algorithm consists of three phases. In the first phase, we apply a clustering algorithm to find groups of similar points. In the second phase we apply an undersampling technique only to those clusters where data is imbalanced among classes. Such an undersampling is called the partial undersampling. Finally, in the third phase, we apply a classification algorithm to solve classification problems in clusters containing points both from majority and minority classes.

5.1 Introduction

In many applications minority classes in imbalanced datasets are considered as outliers and they are removed from the datasets before application of the supervised data classification algorithms. Different approaches have been developed to design algorithms to detect outliers. Clustering techniques have been widely applied to design such algorithms.

As mentioned above most classifiers for imbalanced datasets in one or another way apply oversampling or undersampling techniques. The use of these techniques allows one to improve balance majority and minority classes which

leads to better classification accuracy for minority classes. However, in all these classifiers oversampling or undersampling techniques are applied to the whole datasets. The use of the oversampling techniques may lead to the decrease of importance of majority classes. On the other side the application of undersampling techniques may lead to the loss of significant information on majority classes. This means that in both cases any improvement on minority classes is achieved on expense of majority classes, that is improvement in classification accuracy over minority classes leads to worsening of the classification accuracy over majority classes.

Therefore, it is important to develop an approach which applies oversampling or undersampling techniques only to some parts of majority and/or minority classes. Such an approach allows to prevent the loss of important information on majority classes. This aim can be achieved, in particular, by applying clustering algorithms. Applying clustering algorithms one can get some cluster distribution and then oversampling or undersampling techniques can be applied only to those clusters which contain elements from both majority and minority classes.

We develop a new classifier for imbalanced datasets which is based on this approach. In this chapter, the problem of the supervised data classification in imbalanced datasets is discussed and an algorithm based on the combination of the clustering, undersampling and supervised data classification techniques is developed to solve it. The algorithm consists of three phases. In the first phase, called the clustering phase, we apply a clustering algorithm to find groups of similar points. Several stopping criteria are designed to terminate the clustering algorithm. In the second phase, called partial undersampling, we apply an undersampling technique only to those clusters where data is imbalanced among classes. Such an undersampling is called the partial undersampling. Finally, in the third phase, called the supervised data classification phase, we apply a classification algorithm to solve classification problems in mixed clusters containing points both from majority and minority classes. A classification rule is designed to classify new observations.

The rest of the chapter is organized as follows. Section 5.4 discusses an incremental clustering algorithm which is used in our hybrid method. A brief description of some mainstream classifiers are given in Section 5.5. The proposed algorithm is introduced in Section 5.3. Section 5.6 provides some concluding

remarks.

5.2 Majority, minority and very minority classes

We start with the definition of majority and minority classes. Note that in the literature there is no a unique approach to define such classes. This definition depends on the size of a dataset and in some cases on an application area.

Let $A \subset \mathbb{R}^n$ be a dataset containing p classes: $A_1, \ldots, A_p$. In addition, let n_i be a number of points in the class A_i:

$$n_i = |A_i|,\ i = 1, \ldots, p.$$

Then the total number N of points in the dataset A is:

$$N = \sum_{i=1}^{p} n_i$$

and the average number $\bar{N}$ of points per class is:

$$\bar{N} = N/p.$$

Define the following threshold:

$$N_T = \alpha \bar{N}$$

where $\alpha \in (0, 1)$. In this thesis we take $\alpha = 0.5$.

A class $A_j,\ j \in \{1, \ldots, p\}$ is called a *majority class* if $n_j \geq N_T$, otherwise it is called a *minority class*. We further divide the set of minority classes into two subsets: minority classes and *very minority classes*. A class A_j is called a very minority class if

$$n_j \leq \max\{\bar{n}, \beta \bar{N}\}, \tag{5.2.1}$$

where $\bar{n} > 0$ is sufficiently small integer and $\beta \in (0, 0.5\alpha)$. In large datasets β usually is sufficiently small. In this thesis we take $\bar{n} = 5$ and $\beta = 0.001$.

Consider the set of indices $I = \{1, \ldots, p\}$. We can divide this set into three

subsets $I_{maj},\ I_{min}$ and I_{vm} such that

$$I = I_{maj} \cup I_{min} \cup I_{vm},$$

$$I_{maj} \cap I_{min} = I_{maj} \cap I_{vm} = I_{min} \cap I_{vm} = \emptyset$$

and the set I_{maj} contains indices of all majority classes, the set I_{min} contains indices of minority classes and the set I_{vm} contains indices of all very minority classes.

5.3 The proposed classification algorithm

In this section we introduce a new algorithm for solving the supervised data classification problems. The description of this algorithm will be presented by explaining its each phase in detail. The proposed algorithm consists of the following distinct phases:

Phase 1. In this phase all very minority classes are identified and removed from the data set.

Phase 2. In this phase we apply the incremental clustering algorithm to calculate clusters in the data set after removing very minority classes.

Phase 3. In this phase a undersampling technique is applied to clusters containing data points from both majority and minority classes.

Phase 4. Using outcomes of Phase 3 in this phase we apply a supervised data classification algorithm to separate majority classes as well as majority and minority classes in balanced clusters.

After these four steps we apply the classification rule in order to classify new observations. Next we will explain each phase in detail.

Phase 1. Identifying the very minority classes. Very minority classes are not used both in clustering and supervised classification phases of the algorithm. Such classes are identified by applying the formula (5.2.1) where $\bar{n} = 5$ and $\beta = 0.001$. They do not contain sufficient number of data points to

get any reasonable separation of these classes from other classes or to apply oversampling techniques to them. Therefore, these classes are removed from the dataset before training of the clustering and classification algorithms.

However, the very minority classes will be used in the prediction step. We propose two different approaches to use these points in prediction. We first define some threshold number M_T and if the number of points in the very minority class is less than this number then we define some neighborhood of individual points from these classes. The number M_T is defined as $M_T = 2\bar{n}$. If the number of points is more than this threshold then we calculate their representative. In particular, this representative can be the centroid of the very minor class.

In the first case, neighborhoods of training points from the very minority classes are defined as follows. First we find the distance between points from these very minor classes and those from other classes. Then we take minimum of distances and the radius of the neighborhood is set to be half of the minimum distance. Assume that a point a belongs to one of the very minor classes, say A_j where $j \in I_{vm}$. Define the set $\bar{I} = I_{maj} \cup I_{min} \cup \{I_{vm} \setminus \{j\}\}$ and consider

$$d_{min} = \min_{b \in A_j,\ j \in \bar{I}} \|a - b\|. \tag{5.3.1}$$

Here $\|\cdot\|$ is the Euclidean norm. Then we compute the neighborhood of the point a by selecting any $\gamma \in (0, 0.5]$ and defining the radius $r_a = \gamma d_{min}$. We set $\gamma = 0.5$.

Let us consider the second case. Assume the class $A_j,\ j \in I_{vm}$ satisfies the second condition, that $n_j > M_T$. First, we compute the centroid of A_j:

$$c_j = \frac{1}{n_j} \sum_{a \in A_j} a$$

and then the radius of A_j:

$$r_j = \max_{a \in A_j} \|c_j - a\|.$$

Then we approximate the very minority class A_j with the ball centered at c_j with the radius r_j. It is possible that approximation balls for two very

minor classes have overlaps. We will address this problem when we formulate classification rules.

Phase 2. Clustering. In this phase we remove all very minority classes from the dataset and apply the incremental clustering algorithm to find clusters using the rest of the dataset. Clusters found at each iteration of the incremental algorithm belong to one of these types of clusters:

Type 1: clusters containing only points from majority classes;

Type 2: clusters where majority of points are from majority classes with very few points from minority classes;

Type 3: clusters where majority classes are dominating however, the ratio of minority classes to majority is higher than that of for the original dataset;

Type 4: clusters where minority and majority classes are balanced;

Type 5: clusters where minority classes are dominating;

Type 6: clusters containing only minority classes.

It is important to formulate stopping criteria for the clustering algorithm. The first criterion is formulated by selecting the maximum number k_{max} of clusters which is allowed to calculate. This number is usually large, it is defined using the number of classes as follows $k_{max} = ln_c$ where we set $l = 3$ and n_c is the number of classes in the dataset.

The second stopping criterion is that the algorithm stops when there is no any cluster in the cluster distribution that is of Type 3.

Next, we explain types of clusters which are obtained at each iteration of the incremental clustering algorithm:

1. If any cluster in the cluster distribution belongs to Type 1 we do not store any information about this cluster as the aim of the proposed algorithm is to identify regions of the input space which are used to approximate minority classes and the rest of the input space is used to approximate majority classes.

2. For any cluster $\bar{C}$ belonging to Type 2 we apply the technique similar to that applied to very minority classes with two exceptions. We do not apply the second rule and the neighborhood of these are defined so that they are subset of the cluster. Take any point $a \in \bar{C}$ which from minority class. In this case the distance $\bar{d}_{min}$ is computed as follows:

$$\bar{d}_{min} = \min_{b \in A_j \cap \bar{C},\ j \in \bar{I}} \|a - b\|. \tag{5.3.2}$$

   Let $\bar{r}$ be a radius of the cluster $\bar{C}$ and $\bar{x}$ is its center. Define $\hat{d} = \|\bar{x} - a\|$. Then the radius of the neighborhood of a is defined as

$$r_a = \min\left\{\gamma \bar{d}_{min}, \bar{r} - \hat{d}\right\}.$$

   The outcome of any cluster belonging to Type 2 contains the list of points from minority classes and radii of their neighborhoods.

3. If any cluster belongs to Type 3 then we continue to compute more clusters by adding one cluster at each iteration until the maximum number of clusters allowed. If still some clusters belong to this type we treat them similar to clusters belonging to Type 4.

4. Outcomes from clusters belonging to Type 4 are the list of points from both majority and minority classes as well as cluster centers and their radii. This outcome is passed to the next phase.

5. Clusters belonging Type 5 are treated similar to those belonging Type 2. However, in this case we compute neighborhood radii of data points from majority classes. Here the outcome includes not only the list of majority points and neighborhoods but also the center of the cluster and its radius.

6. Outcomes from clusters belonging to Type 6 are the list of data points from minority classes assigning to these clusters, centers of clusters and their radii.

As a whole outcome of Phase 2, we get a reduced dataset which is passed to Phase 3. Points from very minority classes are removed from the dataset

after Phase 1. All data points from clusters of Type 2 removed after Phase 2. We also remove all points from majority classes belonging to clusters of Type 5. Summarizing, we pass to Phase 3 the following subsets of the dataset:

1) Subset 1 which contains data points from clusters of Type 1;

1) Subset 2 which contains data points from clusters of Types 3 and 4;

2) Subset 3 which contains all data points from clusters of Types 5 and 6 by removing all points from .

Phase 3: Undersampling. In order to deal with clusters of Type 3 obtained using the incremental clustering algorithm we, in principle, can apply undersampling techniques to balance majority and minority classes in these clusters. This type undersampling is applied only to very few clusters not to the whole dataset. Applying these techniques to clusters of Type 3 we transform them into clusters of Type 4. If in some iteration of the incremental clustering algorithm there is no any cluster of Type 3, then the algorithm terminates and the undersampling technique is not applied in this case.

Phase 4: Supervised data classification. A classification algorithm is applied to data points in Subset 1, Subset 2 and Subset 3 separately.

- Subset 1 contains only points from majority classes. If there is only one majority class then the classification is not required. Otherwise a classification algorithm is applied to separate these classes. Note that this separation is computed for the whole input space.

- After application of the undersampling technique Subset 2 contains only clusters of Type 4 and in these clusters majority and minority classes are balanced. We apply the classification algorithm to separate majority classes from minority classes.

- Subset 3 contains only points from minority classes and the classification in this subset is carried out using only points belonging to clusters of Types 5 and 6.

Outcomes of this phase are boundaries between majority classes from Subset 1, boundaries between majority and minority classes in Subset 2 and boundaries between minority classes in Subset 3.

All outcomes of the algorithm. Summarizing, outcomes from all these four phases are as follows:

1. Points from very minority classes with radii of their neighborhoods;
2. Subset of points from minority classes with radii of their neighborhoods;
3. Subset of points from majority classes with radii of their neighborhoods;
4. Clusters of points from minority classes with their centers and radii;
5. Clusters of points from both majority and minority classes with their centers, radii and boundaries between classes
6. Boundaries between majority classes.

Classification rules. Assume that we have a new observation. The classification rule is defined according to the list of outcomes of all four phases formulated above and this rule consists of the following steps:

Step 1. Check whether this observation belongs to the neighborhood of any points from very minority classes. If yes, the observation is assigned to the corresponding class and stop. Otherwise go to Step 2.

Step 2. Check whether this observation belongs to the neighborhood of any points from the subset of points from minority classes. If yes, the observation is assigned to the corresponding class and stop. Otherwise go to Step 3.

Step 3. Check whether this observation belongs to neighborhood of any points from the subset of points from majority classes. If yes, the observations is assigned to the corresponding class and stop. Otherwise go to Step 4.

Step 4. Check whether this observation belongs to any cluster of Types 5 or 6. If yes, the observation is assigned to the corresponding class and stop. Otherwise go to Step 5.

Step 5. Check whether this observation belongs to any cluster of Types 3 or 4. If yes, using boundaries assign the observation to one of the classes and stop. Otherwise go to Step 6.

Step 6. Using boundaries between majority classes assign the observation to one of the majority classes.

5.4 Clustering

In this section we describe a clustering algorithm used in Phase 2 of the proposed classification algorithm.

1) $A^j \neq \emptyset,\ j = 1, \ldots, k$;

2) $A^j \bigcap A^l = \emptyset,\ j, l = 1, \ldots, k,\ j \neq l$;

3) $A = \bigcup\limits_{j=1}^{k} A^j$;

Let $A = \{a^1, \ldots, a^m\}$ be a given finite point set in the n-dimensional space $\mathbb{R}^n$. For a given $k > 0$ the clustering aims to find k groups (called also clusters) of points of the set A using predefined similarity measure. It is assumed that points from the same cluster are similar and points from different clusters are dissimilar to each other. In order to model the clustering problem it is important to formulate the similarity measure. Here we will use the squared Euclidean distance to define this measure:

$$d(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2, \quad x, y \in \mathbb{R}^n.$$

Denote by $A^1, \ldots, A^k$ the cluster distribution of the set A for a given k. Each cluster A^j is represented by its center which is denoted by $x^j,\ j = 1, \ldots, k$. Then the clustering problem is formulated as follows:

$$\text{minimize} \quad f(x^1, \ldots, x^k) \quad \text{subject to} \quad x^j \in \mathbb{R}^n,\ j = 1, \ldots, k, \tag{5.4.1}$$

where the objective function is defined as

$$f(x^1, \ldots, x^k) = \frac{1}{m} \sum_{i=1}^{m} \min_{j=1,\ldots,k} d(x^j, a^i). \tag{5.4.2}$$

The model (5.4.1) is called the nonconvex nonsmooth optimization model of the clustering problem [124, 125].

The most popular algorithm for solving the clustering problem is the k-means algorithm. However, this algorithm is very sensitive to the choice of starting cluster centers. Therefore, we apply the incremental clustering algorithm - the modified global k-means algorithm (MGKM) introduced in [125] and modified in [45]. This algorithm computes clusters gradually starting from one cluster, which is the whole dataset, and adds one cluster center at each iteration.

The most important component of the MGKM algorithm is the procedure for finding starting cluster centers. These points are found by minimizing the so-called auxiliary clustering function ([125]). The MGKM algorithm proceeds as follows.

Algorithm 2 The modified global k-means algorithm.

Input: Data set A, the number $k > 0$ the number of clusters to be calculated, the tolerances $\varepsilon > 0$.

Output: l-partitions of the dataset A, $l = 2, \ldots, k$.

Step 1: (Initialization). Select a tolerance $\varepsilon > 0$. Compute the center $x^1 \in \mathbb{R}^n$ of the set A. Let f^1 be the corresponding value of the objective function (5.4.2). Set $k := 1$.

Step 2: (Computation of the next cluster center). Set $k = k + 1$. Let $x^1, \ldots, x^{k-1}$ be the cluster centers for $(k-1)$-partition problem. Solve the auxiliary clustering problem to find a starting point $\bar{y} \in \mathbb{R}^n$ for the k-th cluster center.

Step 3: (Refinement of all cluster centers). Select $(x^1, \ldots, x^{k-1}, \bar{y})$ as a new starting point, apply k-means algorithm to solve k-partition problem. Let $y^1, \ldots, y^k$ be a solution to this problem and f^k be the corresponding value of the objective function (5.4.2).

Step 4: (Stopping criterion). If

$$\frac{f^{k-1} - f^k}{f^1} < \varepsilon$$

then stop, otherwise set $x^i = y^i,\ i = 1, \ldots, k$ and go to Step 2.

In its first step the MGKM algorithm calculates the center of the whole dataset. Assume that we have already calculated k cluster centers, that is solved the k-clustering problem. In order to solve the $(k+1)$-clustering problem

we first formulate the k-th auxiliary clustering problem and solve it to find the set of starting cluster centers for the $(k+1)$-th cluster center. Then each point from this set is used as a starting cluster center together with other k cluster centers to solve the clustering problem itself. Both the clustering and the auxiliary clustering problems are solved by applying the k-means algorithm.

5.5 Mainstream classifiers

In this chapters, we briefly discuss some mainstream classifiers for general data classification problems applied in Phase 4 of the proposed classification algorithm.

5.5.1 Random Forest

A decision tree is a tree-like structure. The internal nodes of the tree works as a test based on input feature and the leaves represents classes. A test sample is classified by testing each feature against the internal nodes of decision tree. A random forest is an ensemble classifier, which is a group of decision trees proposed by Leo Breiman [42]. A number of decision tree is made from the training data and each individual tree makes a prediction for a given test point. The highest number of votes determines the class label of test point.

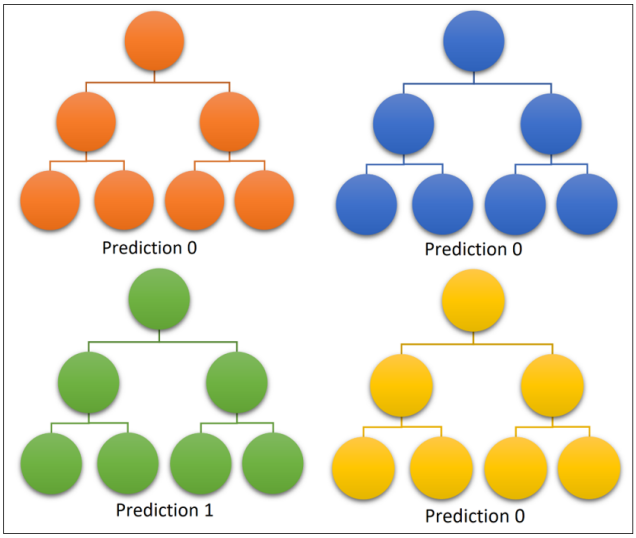

Figure 5.1: Random Forest classifier

5.5.2 K-NN

The k-nearest neighbours (K-NN) is a supervised machine learning technique which is capable of solving regression and classification problems. It calculates the distance from the testing point to all other points and k points are selected with the smallest distance. In case of regression the mean value of these k points are returned as result and in case of classification the majority votes are considered for the label of test point. The selection of k have big impact on performance of this algorithm. A suitable value of k can be selected by iterating through multiple value of k and select the one which has minimum error.

5.5.3 Adaboost

Adaboost which is short for "Adaptive Boosting" is an ensemble machine learning technique which creates a strong classifier by combining a set of weak classifiers. It fits a sequence of weak learners on different weighted training data. It starts by predicting original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an iterative process, it continues to add learners until a limit is reached or all the training points are correctly identified.

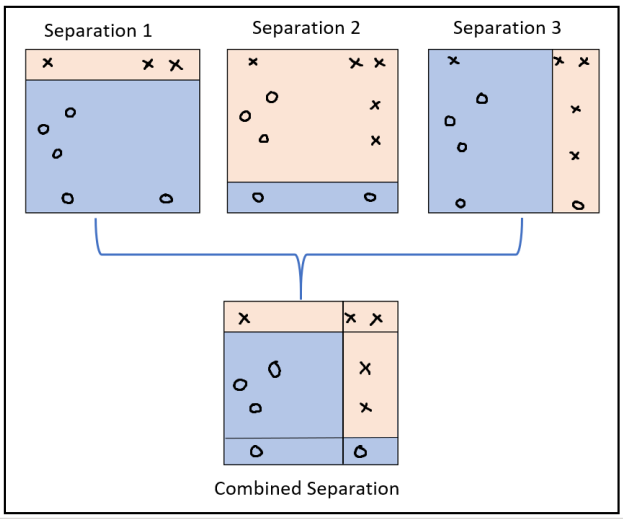

Figure 5.2: Working principle of Adaboost

5.5.4 SVM

Support Vector Machine (SVM) is a supervised machine learning technique that analyze data and recognize patterns, used for classification and regression analysis. An SVM training algorithm works on some train data. Consider a training set $(a^i, b_i),\ i = 1, \ldots, m,\ a^i \in \mathbb{R}^n, b_i \in \{-1, 1\}$. Assume that there is a hyperplane (x, y), where $x \in \mathbb{R}^n,\ y \in \mathbb{R}$, that separates positive elements from negative elements. This can be formulated as follows:

$$\langle x, a^i \rangle + y \geq 1 \quad \text{for} \quad b_i = 1$$

and

$$\langle x, a^i \rangle + y \leq -1 \quad \text{for} \quad b_i = -1$$

The linear SVM aims to maximize the margin, the distance between parallel supporting hyperplanes of positive and negative samples. Using kernels SVM is modified to find nonlinear functions separating these two sets of samples.

SVM is primarily designed for binary class separation. But it can be used for multiclass problem as well. Traditional methods for multiclass problems are one-vs-rest and pairwise approaches. Recently simultaneous classification and various loss functions are used for multiclass classification. One-vs-rest which is also known as one-vs-all (OVA) solves k different binary classification problems. For prediction of a new observation, each classifier is queried once and issues a vote. The class with the maximum number of votes is the winner.

5.6 Conclusion

The mainstream classifiers fail to reach high accuracy for minority classes in imbalanced datasets. Undersampling method (RUS) and oversampling method (SMOTE) are developed to improve classification accuracy for such classes. But in some cases this improvement is achieved in the expense of the majority class. Therefore, it is imperative to develop supervised data classification algorithms which are able to obtain high accuracy in the minority classes without sacrificing too much on the classification accuracy of the majority class. One such method is developed in this chapter. The new method is based on the combination of clustering and supervised data classification algorithms.

The clustering algorithms are applied to identify regions in the input space where minority classes are located and then the supervised data classification algorithms are applied to separate classes in these regions.

Chapter 6

Numerical results

In this chapter we describe the datasets used in our experiments and present numerical results from the experiment with both of our proposed methods discussed in the Chapter 4 and Chapter 5.

6.1 Datasets

In this subsection we first define imbalanced datasets and then provide the description of datasets used in numerical experiments.

6.1.1 Definition of minority classes

We define the minority classes in imbalanced datasets as follows. Let a dataset A with N classes $C_i,\ i = 1, \ldots, N$ be given. Set $n_i = |C_i|$. Compute the following number

$$n_{avg} = \left(\sum_{i=1}^{N} |n_i| \right) \Big/ N \tag{6.1.1}$$

which is the average number of data points per class. Using n_{avg} we define the number $n_{th} = cn_{avg}$, where $c \in (0, 1)$. This number is used as a threshold to determine minority classes. In this paper we use $c = 0.5$. Then we get

$$i - th\ class\ is\ minortiy = \begin{cases} true, & n_i \leq n_{th} \\ false, & otherwise. \end{cases} \tag{6.1.2}$$

6.1.2 Description of datasets

The performance of the proposed algorithms is demonstrated using both binary class and multi-class imbalanced datasets. The binary datasets created by the authors of [126] are obtained from the python library named "imbalanced-learn" [127]. Note that some of these datasets are originally multi-class datasets and the authors transformed these dataset into binary class by selecting one target class and labelling it as positive class and rest of classes are considered as a nontarget and labelled as negative class.

Binary class dataset information

We picked twenty four datasets out of twenty seven datasets from the library. The brief description of binary class imbalanced datasets is given in Table 6.1 where we include the number of features, instances as well as the ratio of the number of instances in the large class to that in the small class.

Multiclass dataset information

The multiclass imbalanced datasets used in our experiments are from the Keel dataset repository [128]. The brief description of these datasets are given in Table 6.2 where we include the number of features, instances, number of classes and imbalance ratio.

6.2 Numerical results for piecewise linear classifier

In this section we present results obtained by our first proposed algorithm presented in Chapter 5, using both the binary class and multi-class imbalanced data sets. Before presenting these results we discuss two main issues related to the application of the proposed algorithm. The first one is the choice of the penalty parameter and the second one is the choice of the number of hyperplanes and their distribution to separate classes.

Table 6.1: Information on selected binary class datasets from Python library

Name	Number of Features	Number of Instances	Imbalance Ratio
Ecoli	7	336	8.6 : 1
Optical_digits	64	5620	9.1 : 1
Satimage	36	6435	9.3 : 1
Pen_digits	16	10992	9.4 : 1
Abalone	10	4177	9.7 : 1
Sick_euthyroid	42	3163	9.8 : 1
Spectrometer	93	531	11 : 1
Car_eval_34	21	1728	12 : 1
Isolet	617	7797	12 : 1
Us_crime	100	1994	12 : 1
Yeast_ml8	103	2417	13 : 1
Scene	294	2407	13 : 1
Libras_move	90	360	14 : 1
Thyroid_sick	52	3772	15 : 1
Coil_2000	85	9822	16 : 1
Arrhythmia	278	452	17 : 1
Solar_flare_m0	32	1398	19 : 1
Oil	49	937	22 : 1
Car_eval_4	21	1728	26 : 1
Wine_quality	11	4898	26 : 1
letter_img	16	20000	26 : 1
yeast_me2	8	1484	28 : 1
ozone_level	72	2536	34 : 1
mammography	6	11183	42 : 1

Table 6.2: Description of multi-class imbalanced datasets

Dataset	Number of Instances	Number of Features	Number of classes	Imbalance Ratio
Ecoli	336	7	8	71.50
Arrhythmia	452	279	13	122.50
Balance	625	4	3	5.88
Yeast	1484	8	10	92.60
Car_eval_34	1728	6	4	18.62
Shuttle control	2175	9	5	853.00

6.2.1 The choice of the number of hyperplanes

The seperation of piecewise linear classifier depends on the pair (m_x, m_n), where m_x is the number of minimum functions under each maximum and m_n is the number of linear functions under each minimum. In the implementation of normal piecewise linear classifer we restrict ourselves with the following set

of hyperplane structure

$$\{(1,1),\ (2,1),\ (3,1),\ (4,2)\}.$$

This choice is reasonable as the larger number of linear functions may lead to overfitting in most datasets. The use of the different number of linear functions leads to the design of the different piecewise linear classifiers. In order to select the right classifier for each dataset we use the validation set which constitute 20% of the training set.

6.2.2 The choice of the penalty parameter

First, we discuss how to select the penalty parameter in finding boundaries between classes. For this purpose we consider Balance, Yeast, Ecoli and Arrhythmia datasets. We select these datasets because the number of minority classes in these datasets is different and this number varies from 1 to 8.

The choice of the penalty parameter may affect the final result. In this paper, we consider two different approaches to choose this parameter. In the first approach all minority classes have the same penalty parameter whereas in the second approach this parameter depends on the number of points in minority classes. In the case of the first approach we choose $c_p = 0, 1$ and 5. In the second case we identify the minority class with smallest number of points and assign to this class the penalty $c_p = 5$. For other minority classes the penalty parameter is defined as follows:

$$c_p^j = \max\left\{\frac{n_s}{n_j}c_p, 1\right\}$$

where n_s is the number of points in the smallest minority class and n_j is the number of points in the j-th class. All results with the penalty parameter with first approach are labeled as $\{(m_x, m_n), c_p\}$ and second approach are labeled as $\{(m_x, m_n), c_p f\}$. For example the configuration ((3,1),5) indicates that the penalty parameter is 5 with m_x and m_n values are 3 and 1 respectively. The configuration ((3,1),5f) indicates the similar setup as the previous configuration with the exception of calculating penalty values. In this case, the smallest class

is assigned the penalty value 5 and rest of the minority classes are assigned fraction of this penalty value based on the number of instances for that class.

6.2.3 Results for binary class imbalanced datasets

We evaluate the performance of the proposed algorithm using above described binary and multi-class real-world imbalanced datasets and compare its performance with that of several mainstream classifiers such as k-NN, Random Forest, SVM and Adaboost as well as with the Easy-Ensemble classifier developed specifically for solving supervised data classification problems in imbalanced datasets. Python implementations of k-NN, Random Forest, SVM and Adaboost algorithms, available from [129], are used in experiments. The Easy-Ensemble classifier is available from [127]. All these algorithms are applied by default.

Table 6.3: Results for overall classification accuracy

Dataset Name	KNN	Random Forest	SVM	Adaboost	PWL (1,1)	PWL (2,1)
Ecoli	89.87	92.84	89.58	86.59	89.07	88.11
Optical_digits	99.41	97.10	90.14	97.46	94.16	96.49
Satimage	92.29	92.79	90.27	91.24	59.49	60.91
Pen_digits	99.79	99.30	90.40	98.37	93.32	98.75
Abalone	88.91	89.87	90.64	90.23	68.85	73.11
Sick_euthyroid	90.10	96.90	91.43	96.74	88.50	89.16
Spectrometer	96.99	97.74	91.53	97.18	97.00	96.80
Car_eval_34	90.80	86.26	80.46	93.46	92.36	90.74
Isolet	98.27	95.78	97.76	96.79	96.79	97.06
Us_crime	92.83	93.68	92.58	93.38	86.82	88.57
Yeast_ml8	92.47	92.59	92.64	91.11	65.21	72.53
Scene	92.52	92.40	92.65	90.40	77.77	83.71
Libras_move	91.44	92.40	93.34	94.72	93.78	93.78
Thyroid_sick	93.40	97.88	94.04	97.43	87.12	91.99
Coil_2000	93.58	92.86	94.02	93.87	68.87	71.12
Arrhythmia	94.25	94.91	94.47	96.46	86.79	86.79
Solar_flare_m0	94.39	91.09	95.10	90.95	69.91	73.33
Oil	95.52	92.20	95.63	94.66	73.43	86.02
Car_eval_4	96.41	95.66	91.43	95.08	92.36	90.74
Wine_quality	95.73	95.94	96.20	95.71	74.82	79.84
Letter_img	99.82	95.58	99.82	99.24	94.22	95.51
Yeast_me2	96.43	96.43	96.56	95.69	84.22	85.09
Ozone_level	97.04	96.89	97.12	96.53	85.88	87.98
Mammography	98.57	98.62	98.51	98.50	90.24	93.48

Table 6.3 presents the overall accuracy on selected binary datasets using

four mainstream classifers and two configurations [(1,1) and (2,1)] of the piecewise linear classifiers. We observe that the overall accuracy is very good for all the datasets and results are similar across different classifiers as well.

Table 6.4: Results on classification accuracy for minority class in binary class datasets

Dataset Name	KNN	Random Forest	SVM	Adaboost	PWL (1,1)	PWL (2,1)
Ecoli	45.71	40.00	0.00	34.29	75.00	75.00
Optical Digits	95.31	70.40	0.00	82.49	92.07	92.43
Satimage	56.87	43.77	0.00	47.60	83.81	82.38
Pen Digits	98.96	93.46	0.00	88.25	92.64	97.92
Abalone	18.93	9.21	0.00	2.56	88.10	84.56
Sick Euthyroid	16.04	74.74	9.56	82.94	87.83	88.97
Spectrometer	71.11	80.00	0.00	73.33	68.00	68.00
Car Evaluation34	24.63	52.99	40.30	84.33	97.04	88.15
Isolet	90.17	48.17	75.83	75.83	83.15	81.98
Us Crime	28.67	30.00	1.33	43.33	70.97	58.06
Yeast ml8	2.25	0.00	0.00	2.25	42.22	28.89
Scene	6.21	5.08	0.00	11.30	34.44	22.22
Libras Movement	37.50	29.17	0.00	50.00	93.23	93.23
Thyroid Sick	12.99	71.00	3.46	77.49	85.96	87.23
COIL 2000	2.05	5.63	0.00	1.37	64.07	63.22
Arrythmia	0.00	8.00	0.00	68.00	43.33	43.33
Solar Flare	8.82	11.76	0.00	13.24	55.71	51.43
Oil	2.44	19.51	0.00	46.34	60.00	44.44
Car Evaluation 4	20.00	18.46	20.00	55.38	75.71	62.86
Wine quality	1.64	7.10	1.64	15.85	65.40	70.27
Letter Image	97.41	90.05	96.19	87.06	96.33	96.73
yeast_me2	13.73	11.76	0.00	21.57	70.91	70.91
Ozone Level	0.00	1.37	0.00	17.81	68.00	54.67
Mammography	53.08	53.46	46.54	51.15	83.40	84.15

As we observe in Table 6.4, the accuracy of minority classes for all mainstream classifiers are significantly low compared to the overall accuracy. This happens because of the trained model being biased towards the majority classes. The results of the piecewise linear classifiers are comparatively better than others.

The mainstream classifiers obtains a overall high accuracy by classifying most of the test points into majority classes, leading into higher accuracy for majority classes. This is reflected by the results presented on Table 6.5. This table presents the accuracy of majority classes on the same datasets as the previous results. We observe that accuracy of majority class is very high, due to the bias of trained model towards majority classes.

Table 6.5: Results on classification accuracy for majority class in binary class datasets

Dataset Name	KNN	Random Forest	SVM	Adaboost	PWL (1,1)	PWL (2,1)
Ecoli	95.02	97.34	100	92.69	87.87	86.89
Optical Digits	99.86	99.88	100	99.09	94.30	96.84
Satimage	96.11	98.69	100	95.94	56.80	58.52
Pen Digits	99.88	99.96	100	99.45	93.32	98.76
Abalone	96.14	97.49	100	99.29	88.10	84.56
Sick Euthyroid	97.67	99.30	99.79	98.15	91.86	88.81
Spectrometer	99.38	99.38	100.00	99.18	98.16	97.96
Car Evaluation34	96.36	90.72	83.88	94.23	91.85	90.85
Isolet	98.94	99.71	99.58	98.54	97.83	98.22
Us Crime	98.05	98.64	100.00	97.45	87.86	90.84
Yeast ml8	99.64	99.87	100.00	97.95	66.96	75.94
Scene	99.37	99.69	100.00	96.68	80.98	88.37
Libras Movement	95.24	97.92	100.00	97.92	93.23	93.23
Thyroid Sick	98.64	99.72	99.94	98.73	86.99	92.10
COIL 2000	99.38	98.30	99.99	99.74	69.11	71.56
Arrythmia	99.77	100	100.00	98.13	88.14	88.14
Solar Flare	98.79	95.46	100.00	94.93	70.26	74.11
Oil	99.78	98.44	100.00	96.88	73.22	87.22
Car Evaluation 4	99.40	98.08	94.23	96.63	93.61	93.87
Wine quality	99.38	99.38	99.87	98.81	75.06	80.08
Letter Image	99.91	99.97	99.96	99.70	94.06	95.44
yeast_me2	99.37	99.72	100.00	98.33	84.39	85.29
Ozone Level	99.92	99.80	100.00	98.86	86.29	88.84
Mammography	99.65	99.82	99.74	99.62	90.35	93.65

Tables 6.6, 6.7 and 6.8 present precision, recall and F-measure for Random forest, KNN, and Naive Bayes, respectively. As we saw higher accuracy of majority classes and lower accuracy for minority classes which results in high precision values and low recall values for these mainstream classifiers. Although we obtain good overall accuracy, the poor precision value leads to poor F-measure values, which shows the drawbacks of these mainstream classifiers.

In cyber security, minority classes are the representative of malicious activities and success rate of detecting minority classes are critical in this area. We improve the accuracy of minority classes by introducing penalty values to push the separating hyperplane towards minority classes to include more points from minority classes. This increases the accuracy of minority classes.

We apply our proposed method with several configurations on seven selected datasets from the python library and the results are presented in Table

Table 6.6: Accuracy, Precision, Recall and F-measure for Random Forest algorithms

Dataset name	Accuracy	Precision	Recall	F-Measure
ecoli	92.86%	0.62	0.62	0.62
optical_digits	97.65%	0.97	0.78	0.87
satimage	93.29%	0.78	0.45	0.57
pen_digits	99.53%	1.00	0.95	0.97
abalone	88.42%	0.31	0.10	0.15
sick_euthyroid	97.09%	0.90	0.80	0.85
spectrometer	97.74%	0.75	1.00	0.86
car_eval_34	97.22%	0.95	0.66	0.78
isolet	95.44%	0.96	0.47	0.63
us_crime	93.59%	0.50	0.28	0.36
yeast_ml8	92.40%	0.00	0.00	0.00
scene	92.36%	0.00	0.00	0.00
libras_move	95.56%	1.00	0.43	0.60
thyroid_sick	97.77%	0.93	0.70	0.80
coil_2000	92.71%	0.17	0.05	0.08
arrhythmia	94.69%	0.00	0.00	0.00
solar_flare_m0	95.69%	0.20	0.08	0.12
oil	95.32%	0.29	0.25	0.27
car_eval_4	98.84%	0.88	0.64	0.74
wine_quality	96.57%	0.54	0.16	0.25
letter_img	99.70%	1.00	0.91	0.95
yeast_me2	97.57%	0.33	0.12	0.18
webpage	98.62%	0.81	0.66	0.73
ozone_level	95.90%	0.00	0.00	0.00
mammography	98.35%	0.82	0.41	0.55

6.9. We can see that adding penalty parameters to the minority classes improve the accuracy for minority classes by a big margin. This is obtained by sacrificing the accuracy of majority class accuracy, which results in decrease in overall accuracy.

We summarize our results in the table 6.10. We can see that our proposed method performs significantly better compared to the mainstream classifiers and the conventional piecewise linear classifier. We also compared our results with EasyEnsemble classifier, which a special classification method to deal with imbalanced data. In some cases our method provides better results for minority class accuracy than EasyEnsemble but our proposed method obtains better accuracy in majority classes for all but except one dataset.

Table 6.7: Accuracy, Precision, Recall and F-measure for KNN algorithms

Dataset name	Accuracy	Precision	Recall	F-Measure
ecoli	90.48%	0.50	0.62	0.56
optical_digits	99.57%	0.99	0.97	0.98
satimage	92.98%	0.65	0.62	0.63
pen_digits	99.89%	1.00	0.99	0.99
abalone	87.27%	0.29	0.16	0.20
sick_euthyroid	88.75%	0.35	0.14	0.20
spectrometer	98.50%	0.89	0.89	0.89
car_eval_34	96.76%	0.88	0.66	0.75
isolet	97.74%	0.89	0.83	0.86
us_crime	92.99%	0.43	0.31	0.36
yeast_ml8	92.73%	0.67	0.04	0.08
scene	92.52%	0.33	0.02	0.04
libras_move	96.67%	1.00	0.57	0.73
thyroid_sick	93.11%	0.41	0.15	0.22
coil_2000	93.32%	0.14	0.02	0.04
arrhythmia	94.69%	0.00	0.00	0.00
solar_flare_m0	96.55%	0.00	0.00	0.00
oil	94.89%	0.17	0.12	0.14
car_eval_4	98.38%	0.83	0.45	0.59
wine_quality	96.41%	0.44	0.09	0.15
letter_img	99.88%	0.98	0.98	0.98
yeast_me2	97.57%	0.33	0.12	0.18
webpage	98.45%	0.87	0.52	0.65
ozone_level	96.06%	0.00	0.00	0.00
mammography	98.35%	0.74	0.50	0.60

6.2.4 Results for multi-class imbalanced datasets

We tested four mainstream classifiers using six different multi-class datasets and the results are presented in Table 6.11. Similar to binary class datasets, in multiclass datasets mainstream classifers suffer from the same problem, where the classifer's training model becomes heavily biased towards majority class and obtains accuracy for overall class by assigning most of the points to majority classes. This leads to very low accuracy for minority classes. We observe that the classes containing least number of points suffers most in terms off accuracy. We also calculated the weighted overall accuracy by assigning weights for minority classes. The low weighted overall accuracy indicates the failure of mainstream classifiers detecting minority classes.

Table 6.8: Accuracy, Precision, Recall and F-measure for Naive Bayes algorithms

Dataset name	Accuracy	Precision	Recall	F-Measure
ecoli	69.05%	0.24	1.00	0.38
optical_digits	27.40%	0.12	0.98	0.21
satimage	81.73%	0.32	0.79	0.46
pen_digits	92.98%	0.67	0.54	0.60
abalone	73.68%	0.26	0.87	0.41
sick_euthyroid	38.69%	0.14	0.99	0.25
spectrometer	89.47%	0.37	0.78	0.50
car_eval_34	91.67%	0.47	1.00	0.64
isolet	88.15%	0.41	0.93	0.56
us_crime	86.77%	0.31	0.88	0.46
yeast_ml8	73.06%	0.07	0.22	0.11
scene	68.44%	0.16	0.77	0.26
libras_move	93.33%	0.57	0.57	0.57
thyroid_sick	39.87%	0.09	0.89	0.16
coil_2000	16.16%	0.07	0.97	0.12
arrhythmia	47.79%	0.07	0.67	0.12
solar_flare_m0	82.76%	0.14	0.75	0.23
oil	88.94%	0.15	0.50	0.24
car_eval_4	98.38%	0.61	1.00	0.76
wine_quality	94.78%	0.27	0.28	0.27
letter_img	96.06%	0.43	0.76	0.55
yeast_me2	19.41%	0.03	1.00	0.05
webpage	74.19%	0.10	0.98	0.17
ozone_level	66.88%	0.09	0.80	0.16
mammography	95.24%	0.30	0.69	0.41

We applied our proposed method with eight different configuration on the same multiclass dataset. Results are presnted in Table 6.12 and 6.13. Our method demonstrates significant improvement on accuracy for minority class. Similar to binary datasets this is achieved by sacrificing accuracy in majority classes. That is why although we got improvement on minority class accuracy, the weighted overall accuracy remains similar as mainstream classifers. We address this issue with our second proposed method.

Table 6.9: Results on classification accuracy obtained by the piecewise linear classifier in binary class datasets

Class Label	Instances	((1,1),0)	((1,1),1)	((1,1),5)	((1,1),5f)	((2,1),0)	((2,1),1)	((2,1),5)	((2,1),5f)
Ecoli									
Class 1	301	87.19	87.19	87.19	87.19	87.19	87.19	87.19	87.19
Class 2	35	85.71	85.71	85.71	85.71	88.57	88.57	88.57	88.57
Average	336	87.02	87.02	87.02	87.02	87.32	87.32	87.32	87.32
Abalone									
Class 1	3786	66.67	66.67	66.67	66.67	71.87	71.87	71.87	71.87
Class 2	391	88.95	88.95	88.95	88.95	86.13	86.13	86.13	86.13
Average	4177	68.76	68.76	68.76	68.76	73.21	73.21	73.21	73.21
Spectrometer									
Class 1	486	98.13	98.13	98.13	98.13	98.13	98.13	98.13	98.13
Class 2	45	75.56	75.56	75.56	75.56	80.00	80.00	80.00	80.00
Average	531	96.20	96.20	96.20	96.20	96.58	96.58	96.58	96.58
Us Crime									
Class 1	1844	87.85	87.85	87.85	87.85	90.13	90.13	90.13	90.13
Class 2	150	73.33	73.33	73.33	73.33	57.33	57.33	57.33	57.33
Average	1994	86.76	86.76	86.76	86.76	87.66	87.66	87.66	87.66
Yeast_me2									
Class 1	1433	84.15	84.15	84.15	84.15	84.78	84.78	84.78	84.78
Class 2	51	77.14	77.14	77.14	77.14	74.29	74.29	74.29	74.29
Average	1484	83.88	83.88	83.88	83.88	84.41	84.41	84.41	84.41
Mammography									
Class 1	10923	90.44	90.44	90.44	90.44	93.77	93.77	93.77	93.77
Class 2	260	84.23	84.23	84.23	84.23	85.38	85.38	85.38	85.38
Average	11183	90.30	90.30	90.30	90.30	93.57	93.57	93.57	93.57
Isolet									
Class 1	600	84.00	95.50	97.50	97.50	82.33	94.00	96.83	96.83
Class 2	7197	97.79	92.82	85.59	85.59	98.37	96.90	93.68	93.68
Average	7797	96.73	93.02	86.51	86.51	97.14	96.68	93.92	93.92

Table 6.10: Classification accuracy for binary class datasets

Dataset Name	KNN	Random Forest	SVM	Adaboost	Piecewise 1:1	Piecewise 2:1	Easy Ensemble (Cross Validation)	Easy Ensemble (One Train-Test)	Piecewise (With Cost)
Minority Class Accuracy									
Ecoli	45.71	40.00	0.00	34.29	75.00	75.00	91.43	87.50	87.19
Abalone	18.93	9.21	0.00	2.56	88.10	84.56	86.45	89.81	71.87
Spectrometer	71.11	80.00	0.00	73.33	68.00	68.00	93.33	100.00	98.13
Isolet	90.17	48.17	75.83	75.83	83.15	81.98	96.67	94.41	90.13
Us Crime	28.67	30.00	1.33	43.33	70.97	58.06	86.00	90.62	84.78
Yeast_me2	13.73	11.76	0.00	21.57	70.91	70.91	82.35	50.00	93.77
Mammography	53.08	53.46	46.54	51.15	83.40	84.15	86.92	89.71	96.83
Overall Accuracy									
Ecoli	89.87	92.84	89.58	86.59	89.07	88.11	85.07	78.57	87.32
Abalone	88.91	89.87	90.64	90.23	68.85	73.11	73.09	73.01	73.21
Spectrometer	96.99	97.74	91.53	97.18	97.00	96.80	93.79	91.73	96.58
Isolet	98.27	95.78	97.76	96.79	96.79	97.06	93.74	94.36	87.66
Us_crime	92.83	93.68	92.58	93.38	86.82	88.57	82.30	81.56	84.41
Yeast_me2	96.43	96.43	96.56	95.69	84.22	85.09	81.47	74.12	93.57
Mammography	98.57	98.62	98.51	98.50	90.24	93.48	88.17	87.63	93.92

6.3 Numerical results for hybrid method

This section presents and discusses the results of our experiment applying the second classifier - the hybrid method. We assigned variable penalties for false-negative and false-positive results and calculated overall cost for the classification. The failure to detect threats (False-negative) is assigned more penalty than misclassifying a benign sample (False-positive). True-negative and true-positive results do not add any cost. A higher penalty is assigned for false-negative prediction. This value can be set based on the importance of a particular classes in a specific problem domain and will be reflected in the overall cost. For example if we consider failing to detect a threat is ten times more harmful than failing to detect a normal activity, then we can set the penalty (P) to 10. For our experiment we used $P = 10$ and $P = 50$. Table 6.14 shows the cost matrix used in our experiment.

Classification accuracy of conventional classifiers is presented in Table 6.15. Results show that the four mainstream classifiers have very high accuracy for the majority classes and achieve a good overall accuracy. But the performance of classifiers on minority classes is very poor compared to the performance of classifier on majority classes. Support Vector Machine (SVM) classifier provides comparatively better results among these four mainstream classifiers.

Table 6.16 shows the accuracy of SMOTE, RUS and our proposed methods combined with the previously mentioned four classifiers. RUS obtains higher accuracy for minority classes in many cases, but it has a lower accuracy rate for majority class. As majority class has large number of points, so small drop in the majority class. Our proposed method has higher accuracy for majority

Table 6.11: Results on classification accuracy by other algorithms in multi-class datasets

Class Label	#Instances	Weight	KNN	Random Forest	SVM	Adaboost
Data set: balance						
Class: 0	49	5.0	10.2	6.12	0.0	24.49
Class: 1	288	1.0	86.11	84.03	95.49	91.32
Class: 2	288	1.0	84.38	80.9	95.83	93.06
Average	625	N/A	79.36	76.48	88.16	86.88
Weighted Average	625	N/A	62.85	59.68	67.11	71.99
Data set: yeast						
Class: 0	463	1.0	60.69	58.1	88.34	79.05
Class: 1	5	5.0	100.0	20.0	0.0	60.0
Class: 2	35	5.0	54.29	57.14	0.0	57.14
Class: 3	44	5.0	72.73	65.91	0.0	63.64
Class: 4	51	5.0	27.45	21.57	0.0	9.8
Class: 5	163	1.0	72.39	80.37	3.07	72.39
Class: 6	244	1.0	53.69	51.64	49.18	2.87
Class: 7	429	1.0	44.29	53.15	20.05	17.72
Class: 8	20	5.0	55.0	20.0	15.0	20.0
Class: 9	30	5.0	0.0	3.33	0.0	0.0
Average	1484	N/A	53.98	55.26	41.98	42.25
Weighted Average	1484	N/A	50.58	48.74	28.55	38.98
Data set: car_eval_34						
Class: 0	384	1.0	34.38	53.12	48.96	63.54
Class: 1	69	5.0	15.94	55.07	31.88	40.58
Class: 2	1210	1.0	78.76	88.1	88.26	79.26
Class: 3	65	5.0	23.08	76.92	80.0	40.0
Average	1728	N/A	64.29	78.59	76.97	72.74
Weighted Average	1728	N/A	53.67	75.53	71.82	65.06
Data set: ecoli						
Class: 0	143	1.0	98.6	98.6	100.0	54.55
Class: 1	77	1.0	77.92	84.42	89.61	85.71
Class: 2	2	5.0	0.0	0.0	0.0	0.0
Class: 3	2	5.0	0.0	0.0	0.0	0.0
Class: 4	35	1.0	48.57	37.14	0.0	0.0
Class: 5	20	5.0	80.0	70.0	0.0	40.0
Class: 6	5	5.0	80.0	80.0	0.0	20.0
Class: 7	52	1.0	88.46	84.62	46.15	44.23
Average	336	N/A	84.52	83.63	70.24	52.38
Weighted Average	336	N/A	80.53	78.1	52.21	46.9
Data set: shuttle						
Class: 1	1706	1.0	99.71	100.0	100.0	92.15
Class: 2	2	5.0	0.0	50.0	0.0	0.0
Class: 3	6	5.0	0.0	50.0	0.0	100.0
Class: 4	338	1.0	97.63	99.41	12.72	99.11
Class: 5	123	5.0	97.56	100.0	4.07	100.0
Average	2175	N/A	98.9	99.72	80.64	93.61
Weighted Average	2175	N/A	97.48	99.18	65.73	94.55
Data set: arrhythmia						
Class: 1	245	1.0	98.37	95.51	100.0	98.37
Class: 2	44	1.0	15.91	50.0	0.0	0.0
Class: 3	15	5.0	40.0	66.67	0.0	0.0
Class: 4	15	5.0	6.67	26.67	0.0	0.0
Class: 5	13	5.0	0.0	15.38	0.0	0.0
Class: 6	25	1.0	0.0	0.0	0.0	0.0
Class: 7	3	5.0	0.0	0.0	0.0	0.0
Class: 8	2	5.0	0.0	0.0	0.0	0.0
Class: 9	9	5.0	0.0	55.56	0.0	44.44
Class: 10	50	1.0	12.0	40.0	0.0	46.0
Class: 14	4	5.0	0.0	0.0	0.0	0.0
Class: 15	5	5.0	0.0	0.0	0.0	0.0
Class: 16	22	1.0	0.0	0.0	0.0	0.0
Average	452	N/A	57.74	65.71	54.2	59.29
Weighted Average	452	N/A	40.36	53.21	34.22	39.66

Table 6.12: Results on classification accuracy obtained by the piecewise linear classifier in multi-class datasets

Class Label	Instances	((1,1),0)	((1,1),1)	((1,1),5)	((1,1),5f)	((2,1),0)	((2,1),1)	((2,1),5)	((2,1),5f)
Balance									
Class 1	49	55.11	73.56	92.00	92.00	90.00	92.00	92.00	92.00
Class 2	288	92.70	91.66	90.96	90.96	90.63	90.97	90.63	90.63
Class 3	288	91.69	91.69	91.33	91.33	91.69	91.69	91.69	91.69
Average	625	89.29	90.25	91.20	91.20	91.06	91.38	91.22	91.22
Weighted Average	821	81.13	86.26	91.38	91.38	90.79	91.51	91.39	91.39
Yeast									
Class 1	244	59.40	59.40	56.93	59.40	63.87	62.24	54.87	59.79
Class 2	429	42.68	41.51	39.42	41.51	46.65	46.88	43.62	46.65
Class 3	463	42.33	39.09	24.62	39.09	40.84	35.43	26.37	34.13
Class 4	44	72.50	72.50	63.33	72.50	65.83	68.33	63.61	68.33
Class 5	35	60.00	57.14	54.29	57.14	65.71	57.14	68.57	57.14
Class 6	51	36.62	43.12	42.08	43.12	42.86	47.53	41.30	47.53
Class 7	163	81.62	79.77	74.92	79.77	81.02	79.20	73.02	79.84
Class 8	30	20.00	20.00	36.67	20.00	16.67	16.67	23.33	16.67
Class 9	20	55.00	55.00	55.00	55.00	55.00	55.00	50.00	45.00
Class 10	5	60.00	40.00	60.00	60.00	60.00	40.00	60.00	60.00
Average	1484	50.48	48.99	43.00	49.06	51.97	49.81	44.16	48.94
Weighted Average	2224	50.22	49.38	45.58	49.62	51.57	49.94	46.20	49.20
Car_Eval_34									
Class 1	1210	50.83	47.77	36.61	36.61	35.70	25.95	30.25	30.25
Class 2	384	15.68	8.87	0.00	0.00	59.86	51.17	36.72	36.72
Class 3	65	75.38	75.38	73.85	73.85	40.00	20.00	46.15	46.15
Class 4	69	25.93	61.10	95.71	95.71	27.14	44.07	54.40	54.40
Average	1728	42.95	40.69	32.24	32.24	40.92	32.04	33.28	33.28
Weighted Average	2264	44.63	47.15	44.76	44.76	39.21	32.06	37.42	37.42
ecoli									
Class 1	143	93.04	93.04	93.04	93.04	93.05	93.05	93.74	93.74
Class 2	77	68.79	68.79	68.79	68.79	60.90	60.90	62.18	62.18
Class 3	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class 4	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class 5	35	68.46	68.46	68.46	68.46	74.35	74.35	74.35	74.35
Class 6	20	80.00	80.00	80.00	80.00	75.00	75.00	75.00	75.00
Class 7	5	83.33	83.33	100.00	100.00	66.67	66.67	100.00	100.00
Class 8	52	88.46	88.46	88.46	88.46	84.62	84.62	84.62	84.62
Average	336	82.13	82.13	82.43	82.43	79.75	79.75	80.93	80.93
Weighted Average	452	78.73	78.73	79.82	79.82	75.22	75.22	77.83	77.83
shuttle									
Class 1	1706	97.60	97.54	97.54	97.54	98.89	98.89	98.89	98.89
Class 2	338	96.15	95.86	96.15	96.15	99.70	99.70	99.70	99.70
Class 3	123	95.94	89.38	82.83	88.56	98.37	99.18	98.36	98.36
Class 4	6	83.33	83.33	50.00	83.33	83.33	83.33	50.00	83.33
Class 5	2	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00
Average	2175	97.20	96.74	96.32	96.74	98.90	98.94	98.80	98.90
Weighted Average	2699	96.70	95.14	93.33	94.99	98.52	98.70	98.15	98.52
arrhythmia									
Class 1	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class 2	25	8.33	8.33	8.33	8.33	8.33	8.33	8.33	8.33
Class 3	50	58.00	58.00	58.00	58.00	58.00	58.00	58.00	58.00
Class 4	245	66.94	65.71	67.34	66.11	66.94	65.71	67.34	66.11
Class 5	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class 6	4	50.00	50.00	50.00	25.00	50.00	50.00	50.00	25.00
Class 7	15	80.36	80.36	45.54	74.11	80.36	80.36	45.54	74.11
Class 8	22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class 9	44	34.09	36.36	38.64	36.36	34.09	36.36	38.64	36.36
Class 10	15	75.00	67.86	67.86	67.86	75.00	67.86	67.86	67.86
Class 11	13	33.33	25.00	39.29	25.00	33.33	25.00	39.29	25.00
Class 12	9	42.50	32.50	62.50	45.00	42.50	32.50	62.50	45.00
Class 13	5	0.00	16.67	0.00	16.67	0.00	16.67	0.00	16.67
Average	452	53.81	52.90	53.76	52.91	53.81	52.90	53.76	52.91
Weighted Average	716	52.65	50.89	50.67	50.42	52.65	50.89	50.67	50.42

Table 6.13: Results on classification accuracy obtained by the piecewise linear classifier in multi-class datasets (cont.)

Class Label	Instances	((3,1),0)	((3,1),1)	((3,1),5)	((3,1),5f)	((4,2),0)	((4,2),1)	((4,2),5)	((4,2),5f)
Balance									
Class 1	49	74.00	88.00	88.00	88.00	92.00	92.00	92.00	92.00
Class 2	288	92.01	90.63	89.59	89.59	97.56	97.56	96.87	96.87
Class 3	288	92.03	92.03	92.03	92.03	92.71	91.33	92.38	92.38
Average	625	90.58	91.06	90.58	90.58	94.88	94.25	94.41	94.41
Weighted Average	821	86.57	90.31	89.94	89.94	94.18	93.69	93.82	93.82
yeast									
Class 1	244	57.35	55.71	51.60	55.31	63.89	61.42	58.15	60.19
Class 2	429	48.98	48.51	46.87	48.75	46.88	45.72	43.62	45.95
Class 3	463	42.75	40.79	27.39	41.01	46.45	43.89	35.42	41.96
Class 4	44	63.61	59.17	61.39	59.17	72.78	68.06	61.11	68.06
Class 5	35	62.86	62.86	60.00	62.86	57.14	60.00	60.00	62.86
Class 6	51	46.75	46.75	50.65	46.75	45.71	42.08	38.44	42.08
Class 7	163	79.73	79.16	73.02	79.80	77.30	74.88	71.09	74.88
Class 8	30	10.00	16.67	23.33	16.67	20.00	20.00	6.67	20.00
Class 9	20	55.00	55.00	45.00	50.00	40.00	40.00	45.00	40.00
Class 10	5	60.00	40.00	80.00	60.00	80.00	80.00	80.00	80.00
Average	1484	51.82	50.67	44.87	50.82	53.38	51.38	46.64	50.71
Weighted Average	2224	50.93	49.99	46.68	50.10	52.50	50.66	46.04	50.39
car_eval_34									
Class 1	1210	39.50	37.44	37.69	36.20	45.21	44.96	37.85	45.12
Class 2	384	72.61	76.49	54.41	62.46	64.03	62.20	43.23	50.24
Class 3	65	29.23	20.00	40.00	38.46	20.00	26.15	58.46	56.92
Class 4	69	62.86	69.89	69.89	71.43	49.23	63.74	54.62	57.47
Average	1728	47.39	46.75	42.77	43.52	48.62	48.85	40.52	47.23
Weighted Average	2264	47.16	46.49	45.74	46.31	45.44	48.09	44.37	49.66
ecoli									
Class 1	143	93.74	93.74	94.43	94.43	91.61	90.91	90.91	90.22
Class 2	77	63.53	63.53	64.81	64.81	67.48	70.04	68.76	70.04
Class 3	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class 4	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class 5	35	62.58	62.58	62.58	62.58	56.86	56.86	59.64	59.64
Class 6	20	75.00	75.00	75.00	75.00	75.00	85.00	80.00	85.00
Class 7	5	100.00	100.00	100.00	100.00	66.67	66.67	66.67	66.67
Class 8	52	86.54	82.69	84.62	84.62	82.69	80.77	82.69	80.77
Average	336	80.33	79.73	80.61	80.61	78.53	79.13	79.13	79.13
Weighted Average	452	77.38	76.94	77.59	77.59	74.30	76.55	75.65	76.55
shuttle									
Class 1	1706	99.53	99.53	99.47	99.53	99.71	99.65	99.71	99.65
Class 2	338	99.70	99.70	99.70	99.70	99.70	100.00	99.70	100.00
Class 3	123	98.37	99.18	98.36	98.36	97.55	97.55	97.55	97.55
Class 4	6	83.33	83.33	50.00	83.33	50.00	66.67	50.00	66.67
Class 5	2	50.00	50.00	50.00	50.00	0.00	0.00	0.00	0.00
Average	2175	99.40	99.45	99.26	99.40	99.36	99.40	99.36	99.40
Weighted Average	2699	98.92	99.11	98.52	98.92	98.30	98.48	98.30	98.48
arrhythmia									
Class 1	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class 2	25	8.33	8.33	8.33	8.33	4.17	4.17	4.17	4.17
Class 3	50	58.00	58.00	58.00	58.00	52.00	50.00	54.00	50.00
Class 4	245	66.94	65.71	67.34	66.11	74.70	75.10	75.51	75.92
Class 5	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class 6	4	50.00	50.00	50.00	25.00	50.00	50.00	50.00	50.00
Class 7	15	80.36	80.36	45.54	74.11	79.46	93.75	93.75	93.75
Class 8	22	0.00	0.00	0.00	0.00	4.55	9.09	4.55	9.09
Class 9	44	34.09	36.36	38.64	36.36	40.91	43.18	40.91	38.64
Class 10	15	75.00	67.86	67.86	67.86	74.11	66.96	60.71	66.96
Class 11	13	33.33	25.00	39.29	25.00	39.29	33.33	33.33	33.33
Class 12	9	42.50	32.50	62.50	45.00	70.00	22.50	55.00	22.50
Class 13	5	0.00	16.67	0.00	16.67	16.67	0.00	0.00	0.00
Average	452	53.81	52.90	53.76	52.91	58.87	58.20	58.87	58.21
Weighted Average	716	52.65	50.89	50.67	50.42	57.96	54.81	56.36	54.82

Table 6.14: Cost matrix

	prediction $y = 1$	prediction $y = 0$
label $h(x) = 1$	$C_{1,1} = 0$	$C_{0,1} = P$
label $h(x) = 0$	$C_{1,0} = 1$	$C_{0,0} = 0$

*0 = Clean Sample and 1 = Infected Sample

Table 6.15: Classification accuracy of mainstream classifiers

Dataset	Class	Rnd. Forest	KNN	Adaboost	SVM
Us-Crime	Majority	98.92	98.92	97.29	99.19
	Minority	30.00	26.67	43.33	36.67
	Overall	93.73	93.48	93.23	94.49
Ecoli	Majority	98.36	96.72	96.72	96.72
	Minority	14.29	85.71	28.57	85.71
	Overall	89.71	95.59	89.71	95.59
Libras Move	Majority	100.00	100.00	100.00	100.00
	Minority	20.00	40.00	80.00	80.00
	Overall	94.44	95.83	98.61	98.61

classes than RUS in most of the cases and it improves the accuracy on minority classes compared to conventional classifying methods. SMOTE provides better accuracy for the majority class but the performance on minority class is worse than both of RUS and our proposed method.

We calculated costs for all our experiment for both $P = 10$ and $P = 50$ and the results are presented in Table 6.17. For all of the cases penalty of misclassifying majority class is set to 1. From the results we can observe that, the cost for mainstream classifier is significantly high for most of the cases. The hybrid method achieved lowest cost for some of the cases and for other cases it provides a competitive results but not the lowest of all. The hybrid method obtains a better accuracy for minority classes for those cases, but the classifying accuracy of majority suffers more than other methods resulting in a higher cost. This suggests the scope of improvement of our proposed hybrid method to obtain a better classifying accuracy for minority classes without sacrificing too much in majority classes.

Table 6.16: Classification accuracy with RUS, SMOTE and the proposed hybrid method

Dataset	Class	Rnd. Forest	KNN	Adaboost	SVM
Applying Random Undersampling (RUS)					
Us Crime	Majority	81.30	80.22	80.49	82.38
	Minority	86.67	93.33	83.33	93.33
	Overall	81.70	81.20	80.70	83.21
Ecoli	Majority	83.61	78.69	83.61	85.25
	Minority	57.14	100.00	71.43	85.71
	Overall	80.88	80.88	82.35	85.29
Libras Move	Majority	88.06	91.04	71.64	95.52
	Minority	100.00	100.00	80.00	100.00
	Overall	94.44	91.67	72.22	95.83
Applying SMOTE					
Us-Crime	Majority	94.04	82.11	91.06	90.24
	Minority	63.33	80.00	60.00	83.33
	Overall	91.73	81.95	88.72	89.72
Ecoli	Majority	95.08	91.80	93.44	90.16
	Minority	85.71	85.71	71.43	85.71
	Overall	94.12	91.18	91.18	89.71
Libras Move	Majority	100.00	95.52	100.00	97.01
	Minority	60.00	100.00	80.00	80.00
	Overall	97.22	95.83	98.61	95.83
Applying the hybrid method					
Us-Crime	Majority	89.70	87.26	83.74	87.80
	Minority	73.33	76.67	83.33	90.00
	Overall	88.47	86.47	83.71	87.97
Ecoli	Majority	95.08	91.80	86.89	93.44
	Minority	71.43	85.71	57.14	85.71
	Overall	92.65	91.18	83.82	92.65
Libras Move	Majority	94.03	86.57	85.07	92.54
	Minority	80.00	100.00	100.00	100.00
	Overall	93.06	87.50	98.61	93.06

Table 6.17: Cost for various classification methods including the proposed hybrid method

Dataset	Penalty	Rnd. Forest	KNN	Adaboost	SVM
Cost for conventional methods					
Us-Crime	P = 10	214	224	180	193
	P = 50	1054	1104	860	953
Ecoli	P = 10	61	12	52	12
	P = 50	301	52	252	52
Libras Move	P = 10	40	30	10	10
	P = 50	200	150	50	50
Cost for RUS					
Us-Crime	P = 10	109	93	122	85
	P = 50	269	173	322	165
Ecoli	P = 10	40	13	30	19
	P = 50	160	13	110	59
Libras Move	P = 10	8	6	29	3
	P = 50	8	6	69	3
Cost for SMOTE					
Us-Crime	P = 10	132	126	153	86
	P = 50	572	366	633	286
Ecoli	P = 10	13	15	24	16
	P = 50	53	55	104	56
Libras Move	P = 10	20	3	10	12
	P = 50	100	3	50	52
Cost for the hybrid method					
Us-Crime	P = 10	118	117	110	75
	P = 50	438	397	310	195
Ecoli	P = 10	23	15	38	14
	P = 50	103	55	158	54
Libras Move	P = 10	14	9	10	5
	P = 50	54	9	10	5

6.4 Conclusion

In this chapter we presented results from both of our proposed methods using several publicly available dataset. The results prove superiority of our methods over mainstream classifier in terms of achieving higher accuracy for minority classes in an imbalanced dataset. Our first method obtains these results by sacrificing accuracy of majority classes. We overcome this limitation in our second method using restricted undersampling. This method obtains better accuracy in minority classes over mainstream classes without sacrificing too much accuracy in majority classes.

Chapter 7

Application of proposed methods in internet security

In this chapter we discuss the importance of machine learning in cyber security and present general architecture for implementing both of our proposed classification method introduced in Chapter 4 and Chapter 5. We also present some numerical results applying our proposed methods using our own generated dataset and five publicly available datasets related to cyber security.

7.1 Introduction

Machine learning techniques are used in various areas of cybersecurity. These techniques have been applied for malware detection, anomaly detection, network traffic monitoring, generating security alert and many more. Cyber security landscape is heavily influenced by machine learning in recent years. Thousands of new variants of cyber attacks are being introduced every hour and traditional signature based detection solution are unable keep up with the increasing number of attacks. With the help of advanced sophisticated technique, some of the new malware can bypass end-user's detection mechanism. Machine learning based solutions are becoming more effective as it has the capability of detecting new samples based on analyzing previously trained patterns.

We developed two classification methods in earlier chapters which can deal with imbalanced datasets. As real-world security datasets are often imbalanced due to having a smaller number of infected samples compared to clean samples, these methods can be applied in cyber security domain.

7.2 Architecture for implementing proposed algorithms

We propose a one-stop solution by implementing our classification algorithms at server side. This approach allows the solution to be available for every user of a particular organization. To apply a machine learning based algorithm, a training model is generated using training data and saved in the server. A new observation is tested using the previously trained model and based on the prediction a decision is made. We propose two different architectures for our proposed classification algorithms to apply them to solve cyber security problems.

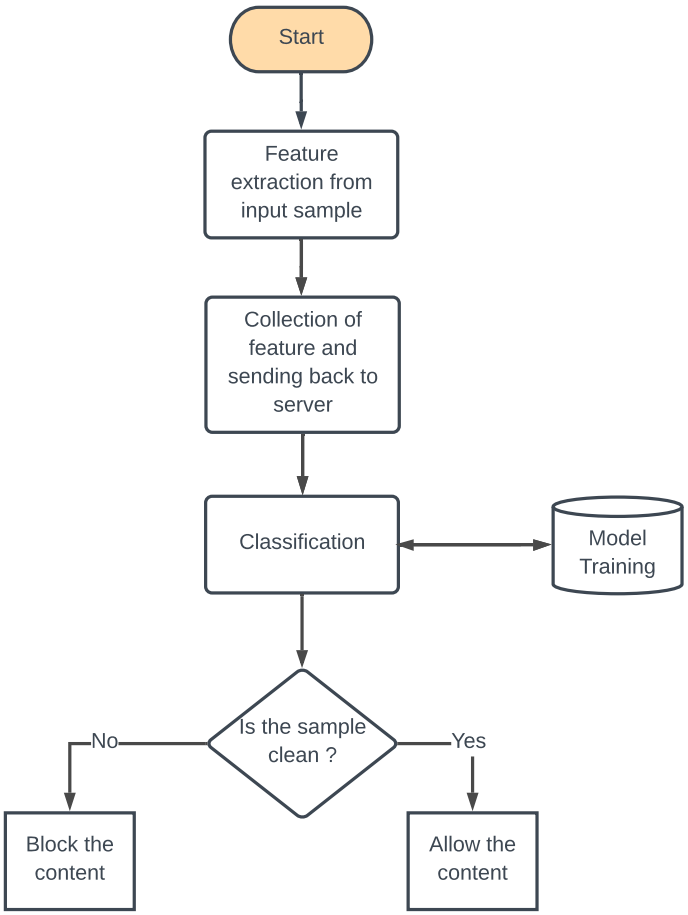

Figure 7.1: Implementation architecture with one training model.

7.2.1 Architecture for the first proposed method

Our first method is implemented with a single trained model. First, a classification model is trained with our proposed piecewise linear classifier for imbalanced data and stored in the server.

In the first phase of testing, we extract similar features as the training dataset from the input sample and the collection of features are sent back to the server for classification. The server then used previously trained model to predict the nature of the input sample. If the provided sample is classified as clean sample, then the content is allowed to execute further, otherwise it is blocked and warning signal is generated. Figure 7.1 presents the flow chart of

our implementation architecture for the first method.

7.2.2 Architecture for second proposed method

For our second method, we train multiple models for one training data. In the training phase, the input dataset is categorized into multiple clusters and a training model is generated for each clusters.

Similar to our first architecture, the features from test input is extracted and sent back to the server. The distance between the testing point and the centres for each cluster is calculated and the closest cluster is selected for classifying the input sample. Depending on the cluster type, appropriate classification method is applied using the training model for the representative cluster. Figure 7.2 presents the proposed architecture for our second method.

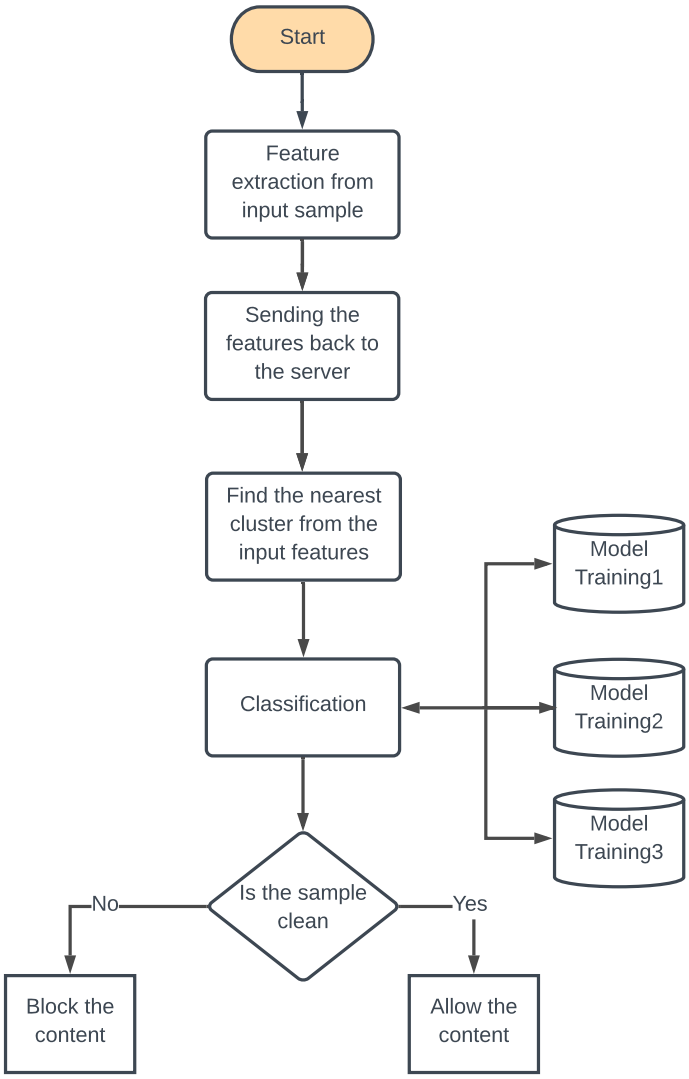

Figure 7.2: Implementation architecture with multiple training model.

7.3 Dataset description

We applied our proposed methods on six datasets in the cybersecurity domain. Five of these datasets are publicly available dataset including the categories of internet traffic analysis, mobile malware, fraudulent credit card transfer and spam emails. We also tested the methods with our own generated dataset, that was created by simulating web inject using google chrome extension.

7.3.1 NSL-KDD dataset

This dataset contains the information about internet traffic observed by an intrusion detection network. This is an improvement over and older dataset named KDD'99. This dataset removed the redundant records from the training and test sets. Each sample points in the dataset contains 41 traffic input features, one label and one last feature indicating the difficulty level of prediction for that input sample. The class label is a text file containing the name of the attack or the word "normal" for a clean sample. We converted the class label into numeric value as well by encoding the label. There are four classes of attacks are listed in the dataset, they are: Denial of Service (DoS), Probe, User to Root(U2R), and Remote to Local (R2L).

There are four sub data sets:

- KDDTest+
- KDDTest-21 (subset of KDDTest+)
- KDDTrain+
- KDDTrain+_20Percent (subsets of the KDDTrain+)

KDDTrain+ is the main train dataset and KDDTest+ is the main test dataset. The subset KDDTest-21 is obtained by removing the most difficult traffic records from the main test dataset. The KDDTrain+_20Percent is obtained by selecting 20% of the entire train dataset. These two subsets do not contain any points that are not present in main train and test datasets. We conduct our experiment by selecting one target attack sample and all normal samples from both train and test datasets, which makes it a binary imbalanced dataset. This subset contains 77054 clean samples and 737 attack samples, with an imbalanced ratio of 104.55.

Although this dataset is widely used by researchers, it is not a perfect representative of real network scenario. As network traffic, operating system, applications and attack behaviour has changed over time, this old dataset became outdated in modern era. To protect the integrity and continuity of a business, an enterprise often does not want to disclose their network data,

which makes it difficult to create a real-world dataset for intrusion detection system (IDS). To overcome these limitation, a synthetic realistic dataset is proposed by Haider *et al.* [130]. They evaluated the realism of existing IDS dataset using fuzzy logic system based metric and the dataset was generated based on this metric.

7.3.2 Credit card dataset

This dataset is obtained by collecting credit card transaction information occurred in two days in September 2013 by European cardholders. Among all the transactions, 492 were fraudulent transaction and 284315 normal transaction. The dataset is highly imbalanced with imbalance ratio of 577.88.

The dataset is processed so that all input variables contain numeric values. PCA transformation is used on all features except two, (Time and Amount) to convert all inputs into numeric values. There are thirty features for each data point. The features are labelled as V1,V2 …V28, Time, and Ammount. Actual feature labels and background information about the data is not provided with the dataset due to privacy concern. These information is not relevant for classification process. There are two labels for the class: 0 and 1. The fraud transaction are labelled as 1 and normal transaction is represented with 0.

7.3.3 Drebin

The Drebin dataset is designed to experiment with android malware [131, 132]. The dataset is created by collecting samples in the period of August 2010 to October 2012 and it was made public for both academia and industry upon requesting the access to the authors. We used Drebin dataset for malware detection which can be collected by requesting the access from the authors. The Drebin dataset features are generated from 129013 Android apk files. Among these samples 123453 are benign applications and 5560 samples are malware of different families, which makes this an imbalanced dataset. Eight different types of information is collected from each samples. The categories are: Hardware components, Requested permissions, App components, Filtered intents, Restricted API calls, Used permissions, Suspicious API calls and Network addresses. App components have four sub categories (activities, services, content providers and broadcast receivers). There are around 545,000 unique values

for these 11 categories and the original authors used each value as a feature, which creates a very high dimensional dataset, which is not suitable for real time detection. Instead of these unique values, we used the categories as features and counted each one's number of occurrences as the feature value. This reduces the number of features to 11.

7.3.4 Spambase

For spam detection we used Spambase dataset, which is available at UCI repository. Spambase dataset contains 2788 clean and 1813 spam email samples. To create a more imbalanced situation we used all normal emails and a first 200 spam samples. The collection of spam e-mails came from our postmaster and individuals who had filed spam. The collection of non-spam e-mails came from filed work and personal e-mails.

7.3.5 NB-15

The raw network packets of the UNSW-NB 15 dataset was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviours.

Tcpdump tool is utilised to capture 100 GB of the raw traffic (e.g., Pcap files). This dataset has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The Argus, Bro-IDS tools are used and twelve algorithms are developed to generate totally 49 features with the class label. The number of records in the training set is 175,341 records and the testing set is 82,332 records from the different types of attacks and normal activities.

NB-15 is a relatively modern dataset compared to other datasets used in our experiment. It uses nine modern attack types and realistic normal traffic to generate the dataset. One of the problem of this dataset is that, it does not contain any attack related to cloud computing, which is an essential part of the most web based solution now-a-days. It also contains a large number of duplicate records in training set [133].

7.3.6 User generated dataset

We created one dataset by simulating web injection using developed a customized google chrome extension. In practical scenario, number of infected instances are very low compared to the number of clean instances. This creates an imbalanced dataset for training the machine learning techniques. We generated an imbalanced dataset by collecting features from 793 web pages and among them 693 pages were clean, and rest 100 pages contains injection. A reduced size dataset was generated which contains 307 clean and 100 infected samples.

7.4 Experimental setup

In this section we present the numerical results of our first method on the datasets described above. The results prove the consistency of our proposed algorithms. We observe significant improvements of minority class accuracy by adding the penalty parameter. For obtaining the optimal results for second proposed algorithm various parameters are needed to be set and it varies from dataset to dataset. Finding the optimal parameters are challenging task and further research can be done to select the proper parameters.

We compared the performance of four mainstream classifiers with our proposed methods. A python machine learning library named “scikit-learn” is used to implement the mainstream classification models. The proposed piecewise linear classifier for imbalanced datasets is implemented in Fortran 77 and compiled using freely available “gfortran” compiler. The second algorithm, the hybrid algorithm, is implemented in python. We used one training and one test sets to calculate accuracy of different methods. 20% of each class is used for testing and rest is used for training purpose. Various penalty parameters are used for our first method and we present results of three different penalty values applying on two datasets.

7.5 Numerical results

We applied the piecewise linear classifier with various penalty parameter to obtain higher accuracy for minority class. Higher penalty value results in higher accuracy for minority classes. It comes at a price of sacrificing accuracy in majority class. Table 7.1 shows the results with various penalty values on creditcard and spambase dataset.

Table 7.1: Classification accuracy for the piecewise linear classier

Dataset	Class	No penalty	Penalty=1	Penalty=2	Penalty=3
Creditcard	Minority	83.84	88.89	91.92	89.90
	Majority	99.17	96.22	91.82	93.68
	Overall	99.14	96.21	91.82	93.67
Spambase	Minority	85.15	94.06	95.05	92.08
	Majority	84.77	72.94	63.44	54.30
	Overall	84.83	76.18	68.29	60.09

The results show that increasing the penalty values leads to increase of accuracy for minority class. Applying higher penalty value pushes the separating boundary of the minority class further apart to include more minority points inside it. This results in acquiring majority points inside the minority area and decrease the accuracy of the majority class. Our second proposed method address this issue.

Table 7.2 compares the results of our proposed method with four other mainstream classifiers. We can observe that in all cases the proposed piecewise obtains a higher accuracy for minority classes. The cost for achieving this is a drop in accuracy of majority classes which is reflected in the results. The subset of NSL-KDD dataset that we used in our experiment has a very clear separating boundary and there was no room to improve the classification accuracy. Performance of mainstream classifiers and our proposed methods are very similar for this dataset. For our own generated dataset the results obtained by our method was not the best suitable solution. We obtain the highest classification accuracy in minority class, but it suffers the classification accuracy of majority classes by a big margin.

Table 7.3 presents results obtained using the hybrid classification algorithm

Table 7.2: Comparison of mainstream classifiers with our first proposed method

Dataset	Class	KNN	Rnd. Forest	SVM	Adaboost	Proposed
Creditcard	Minority	7.14	80.61	4.08	69.39	91.92
	Majority	100.0	99.99	100.0	99.96	91.82
	Overall	99.84	99.95	99.83	99.91	91.82
Spambase	Minority	36.0	74.0	28.0	72.0	94.06
	Majority	95.52	99.46	99.28	98.03	72.94
	Overall	86.47	95.59	88.45	94.07	76.18
Drebin	Minority	77.07	80.04	67.81	35.88	86.94
	Majority	99.18	99.71	99.81	99.33	86.68
	Overall	98.23	98.86	98.43	96.6	86.47
NB-15	Minority	75.82	58.46	71.37	59.83	80.78
	Majority	97.78	94.47	97.25	95.11	88.48
	Overall	87.91	78.28	85.62	75.69	88.27
NSL-KDD*	Minority	92.52	100.0	55.78	100.0	99.32
	Majority	99.96	100.0	100.0	100.0	99.81
	Overall	99.89	100.0	99.58	100.0	99.80
Generated	Minority	20.0	75.0	25.0	75.0	85.71
	Majority	97.12	98.56	99.28	94.96	23.74
	Overall	99.89	100.0	99.58	100.0	99.80

introduced in Chapter 5. Note that any general purposed classifier can be used as a part of this algorithm. We use k-NN, Random Forest, SVM and Adaboost classifiers. The use of the hybrid algorithm allows to apply these classifiers locally, not globally to the whole dataset. This way we can improve the classification accuracy for minority classes without sacrificing too much on classification accuracy for majority classes.

Here we present results with only Spambase dataset as results with other datasets demonstrate the similar trend.

One can see from results given in Table 7.3 that in all cases the use of the hybrid algorithm significantly improves classification accuracy for the minority class. However, this improvement in the case of Random Forest and Adaboost algorithms does not lead to much worsening of the classification accuracy for the majority class.

Table 7.3: Comparison of mainstream classifiers with and without applying the hybrid method on Spambase dataset

Class	KNN	Rnd. Forest	SVM	Adaboost
Without applying hybrid method				
Minority	36.0	74.0	28.0	72.0
Majority	95.52	99.46	99.28	98.03
Overall	86.47	95.59	88.45	94.07
After applying hybrid method				
Minority	68.90	88.81	65.89	83.36
Majority	66.00	85.50	62.50	91.00
Overall	68.71	88.59	65.66	83.87

7.6 Conclusion

In this chapter we presented general architecture for implementing our proposed methods in detection of cyber threats. Both of our approach focuses on server-side implementation using pre-trained model. Our first method uses one training model and second method uses multiple training model for classifying the input sample. The performance of our systems heavily depend on the training dataset and selected features.

Numerical results presented in this chapter demonstrate that both proposed algorithms: the piecewise linear and the hybrid algorithms significantly improve the classification accuracy for minority classes. This means that they are accurate algorithms for detecting malware activities. Furthermore, in the case of the hybrid algorithm this accuracy is not achieved on the expense of the majority classes.

Chapter 8

Conclusion

In this chapter we present a summary of our thesis, provide a concluding remarks, discuss the limitation of our study and possible directions for future research.

8.1 Summary

In this digital world online security is a major concern. Various types of attacks are emerging everyday causing huge financial damage to business and individuals. Researchers are developing new techniques to fight against those attacks. Machine learning based detection methods have gained popularity in recent times due to its capability of detecting unexplored threats.

Various type of cyber attacks and their working principles are discussed in Chapter 1. Some attacks spread automatically and some needs a carrier to spread. Banking malwares are specialized attacks targeting particular banking organization. A banking malware named Zeus, caused huge financial damage in 2009. Many of its versions were introduced later. The objective of our research is to design an effective solution to detect cyber threats.

Chapter 2 discusses various methods of detecting cyber attacks and their shortcomings. In broad sense, the detection methdos are divided into three categories. They are: signaturebased techniques, anomaly-based techniques and hybrid techniques. Signature based methods works on some pre-defined

rules based on explored threats. It cannot detect new and unknown attacks, but it has higher detection rate for known attacks and has a very less false positive rate. Anomaly based detection methods work by generating model based on behaviours of known malwares. These trained models are used to detect attacks including new and unexplored threats. Machine learning based techniques are examples of anomaly based detection methods. There are four main categories of machine learning techniques, namely supervised learning, unsupervised learning, regression analysis and reinforcement learning. Our research involves supervised and unsupervised learning techniques.

In Chapter 3 we explored the use of machine learning techniques in cyber security area. We simulated a web inject by designing a google chrome extension and applied various supervised learning models to detect the attack. Our initial finding guides us to design more sophisticated machine learning techniques to deal with imbalanced data in cyber security domain.

Chapter 4 discusses our first proposed method which is a modification of piecewise linear classifier fro imbalanced datasets. In cyber security area detecting minority classes have higher priority as minority classes indicate threats and failing to detect a threat causes high damage and in particular, the huge financial damage. We designed a cost sensitive piecewise linear classifier by applying penalty for minority classes in the dataset which shifts the separating lines to include more minority points which results in increasing classification accuracy of minority classes.

Our second proposed method was discussed in Chapter 5. This method is a partial undersampling technique which combines both supervised and unsupervised learning. This method is an improvement over previous method. Our first method obtains higher accuracy in minority class by sacrificing a lot in majority class. This second method improves the accuracy of minority classes without sacrificing too much in classification accuracy of majority classes. Numerical results from both of these methods are presented in Chapter 6.

In Chapter 7 we discuss application of machine learning in cybersecurity. We present two general architectures for implementing our proposed algorithms.

We also discuss 5 publicly available dataset in cybersecurity area and present numerical results of the experiments performing on them.

8.1.1 Summary of cost-sensitive piecewise linear classifier

We defined minority classes by calculating imbalance ratio between different classes. We applied higher penalty value for the most smallest class and assigned fraction of that penalty values to other minority classes based on the ratio between their numbers. We compared our method with 4 other mainstream classifier namely KNN, Random Forest, SVM and Adaboost and in most of the cases we obtained higher classification accuracy in minority classes.

8.1.2 Summary of partial undersampling

For this method we used a clustering method, the modified global k-means algorithm (MGKM), and isolated the pure clusters. For the mixed clusters we train a model using several classification techniques. These results in improvement in minority class accuracy, as the imbalance ratio in mixed clusters are less than the imbalanced ratio of the original dataset.

8.2 Conclusion

We found from our experiment that mainstream classifiers provide poor performance in achieving higher classification accuracy in minority classes in imbalanced data, which is crucial in cyber security area as minority classes represent the threat. The cost sensitive piecewise linear classifier partially solves the problem but comes with a price of big sacrifice in majority classes. The undersampling method (RUS) and oversampling method (SMOTE) provide comparatively better solutions. These methods modify the dataset by adding or removing points. But in some cases the solution provided by these methods is achieved in the expense of the majority classes. The objective is to obtain higher classification accuracy in the minority classes without sacrificing too much in the majority classes. Our second proposed method obtains relatively

higher classification accuracy in minority classes without sacrificing too much in majority classes.

8.3 Limitations and future work

The main drawback of our first proposed method is sacrificing a big accuracy percentage in majority classes in order to achieve higher accuracy in minority class. We tried to overcome this drawback in our second method, and there is still room for improvement. The selection of penalty parameter for each dataset is different and chosen manually by iterating through different values. A future research can be done on how to find the best penalty parameter for a particular dataset. Our second method reduces the drop of accuracy in majority classes by using clustering algorithm on dataset before applying classification algorithms. Depending on points distribution inside each clusters we divided them in three categories, which are: only majority clusters, only minority clusters and mixed clusters. In future work clusters can be divided into more sub categories based on imbalance ratio and different classification rules can be applied depending on cluster types.

Combining the cost-sensitive approach with partial undersampling may lead to a better solution. We believe that an approach based on the combination of clustering, supervised data classification, undersampling and oversampling techniques will lead to the design of more efficient and accurate algorithms for solving the supervised data classification problems in imbalanced datasets. In turn, this direction of research will lead to the design of accurate algorithms to detect malware activities.

Feature extraction plays important role in data classification, as the performance of a classification algorithm will vary depending on selection of appropriate features. Developing a good feature extractor is another part of the research, which is not addressed in our study. Future works can be done on feature selection to increase the efficiency of the classification algorithm and improving the classification results.

Bibliography

[1] N. Milosevic, "History of malware," *arXiv preprint arXiv:1302.5392*, 2013.

[2] Kaspersky, "Financial cyber threats in 2013," 2013.

[3] P. Cucu. (2017) How a banking trojan does more than just steal your money. [Online]. Available: https://heimdalsecurity.com/blog/banking-trojan/

[4] Accenture, "Cost of cyber crime study," 2017.

[5] McAfee, "The economic impact of cybercrime no slowing down," 2017.

[6] S. Morgan, "2017 cybercrime report," 2017.

[7] M. Stahlberg, "The trojan money spinner," in *Virus bulletin conference*, vol. 4, 2007.

[8] M. Heiderich, T. Frosch, and T. Holz, "Iceshield: Detection and mitigation of malicious websites with a frozen dom," in *Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection*, ser. RAID'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 281–300.

[9] L. Kharouni, "Automating online banking fraud, automatic transfer system: The latest cybercrime toolkit feature," *Trend Micro Incorporated, Tech. Rep*, 2012.

[10] J. Kałużny and M. Olejarka, "Script-based malware detection in online banking security overview," *Black Hat Asia*, 2015.

[11] S. Saha, S. Jin, and K.-G. Doh, "Detection of dom-based cross-site scripting by analyzing dynamically extracted scripts," in *The 6th International Conference on Information Security and Assurance*, 2012.

[12] A. K. Dalai, S. D. Ankush, and S. K. Jena, "Xss attack prevention using dom-based filter," in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, P. K. Sa, M. N. Sahoo, M. Murugappan, Y. Wu, and B. Majhi, Eds. Singapore: Springer Singapore, 2018, pp. 227–234.

[13] A. W. Marashdih and Z. F. Zaaba, "Detection and removing cross site scripting vulnerability in php web application," in *2017 International Conference on Promising Electronic Technologies (ICPET)*, Oct 2017, pp. 26–31.

[14] B. Sullivan, "Server-side javascript injection," *Black Hat USA*, 2011.

[15] O. Eisen, "Catching the fraudulent man-in-the-middle and man-in-the-browser," *Network Security*, vol. 2010, no. 4, pp. 11 – 12, 2010.

[16] E. Kalige, D. Burkey, and I. Director, "A case study of eurograbber: How 36 million euros was stolen via malware," *Versafe (White paper)*, vol. 35, 2012.

[17] S. Specht and R. Lee, "Taxonomies of distributed denial of service networks, attacks, tools and countermeasures," *CEL2003-03, Princeton University, Princeton, NJ, USA*, 2003.

[18] "Man-in-the-middle (mitm) attacks: Techniques and prevention." [Online]. Available: https://www.rapid7.com/fundamentals/man-in-the-middle-attacks/

[19] W. G. Halfond, J. Viegas, A. Orso *et al.*, "A classification of sql-injection attacks and countermeasures," in *Proceedings of the IEEE international symposium on secure software engineering*, vol. 1. IEEE, 2006, pp. 13–15.

[20] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey

and research directions," *Computers and Security*, vol. 74, pp. 144 – 166, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016740481830004X

[21] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016.

[22] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," in *Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, 1990, pp. 296–304.

[23] C. S. Inc., "Netranger intrusion detection system technical overview," Dec 1998. [Online]. Available: http://storage.library.opu.ua/online/external/cisco/products/778/security/netranger/ntran_tc.htm#:~:text=TheNetRangerintrusiondetectionprocess,andpacketcontentstringmatches.&text=WhentheNetRangersystemanalyzes,looksforpatternsofmisuse.

[24] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration*, ser. LISA '99. USA: USENIX Association, 1999, p. 229–238.

[25] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic, "Noxes: A client-side solution for mitigating cross-site scripting attacks," in *Proceedings of the 2006 ACM Symposium on Applied Computing*, ser. SAC '06. New York, NY, USA: ACM, 2006, pp. 330–337.

[26] A. W. Marashdih, Z. F. Zaaba, and H. K. Omer, "Web security: Detection of cross site scripting in php web application using genetic algorithm," *International Journal of Advanced Computer Science and Applications (ijacsa)*, vol. 8, no. 5, 2017.

[27] L. Sun, S. Versteeg, S. Boztaş, and T. Yann, "Pattern recognition techniques for the classification of malware packers," in *Information Security and Privacy*, R. Steinfeld and P. Hawkes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 370–390.

[28] C. Criscione, F. Bosatelli, S. Zanero, and F. Maggi, "Zarathustra: Extracting webinject signatures from banking trojans," in *Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on*. IEEE, 2014, pp. 139–148.

[29] A. Gamachchi, L. Sun, and S. Boztas, "A graph based framework for malicious insider threat detection," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 01 2017.

[30] A. Continella, M. Carminati, M. Polino, A. Lanzi, S. Zanero, and F. Maggi, "Prometheus: Analyzing webinject-based information stealers," *Journal of Computer Security*, vol. 25, no. 2, pp. 117–137, 2017.

[31] S. Lekies, B. Stock, and M. Johns, "25 million flows later: large-scale detection of dom-based xss," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 1193–1204.

[32] B. Stock, S. Pfistner, B. Kaiser, S. Lekies, and M. Johns, "From facepalm to brain bender: exploring client-side cross-site scripting," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1419–1430.

[33] A. Fattori, A. Lanzi, D. Balzarotti, and E. Kirda, "Hypervisor-based malware protection with accessminer," *Computers & Security*, vol. 52, pp. 33–50, 2015.

[34] R. P. Lippmann and R. K. Cunningham, "Improving intrusion detection performance using keyword selection and neural networks," *Computer Networks*, vol. 34, no. 4, pp. 597 – 603, 2000, recent Advances in Intrusion Detection Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128600001407

[35] M. A. Aydın, A. H. Zaim, and K. G. Ceylan, "A hybrid intrusion detection system design for computer network security," *Computers & Electrical Engineering*, vol. 35, no. 3, pp. 517 – 526, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790609000020

[36] A. I. Saleh, F. M. Talaat, and L. M. Labib, "A hybrid intrusion detection system (hids) based on prioritized k-nearest neighbors and optimized svm classifiers," *Artificial Intelligence Review*, vol. 51, no. 3, pp. 403–443, Mar 2019. [Online]. Available: https://doi.org/10.1007/s10462-017-9567-1

[37] M. Kubat, *An Introduction to Machine Learning*. Springer, 2016.

[38] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[39] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: Association for Computing Machinery, 1992, p. 144–152. [Online]. Available: https://doi.org/10.1145/130385.130401

[40] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002200009791504X

[41] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[42] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[43] J. H. Friedman, "Multivariate adaptive regression splines," *The annals of statistics*, pp. 1–67, 1991.

[44] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[45] A. M. Bagirov, J. Ugon, and D. Webb, "Fast modified global k-means algorithm for incremental cluster construction," *Pattern recognition*, vol. 44, no. 4, pp. 866–876, 2011.

[46] L. Sun, S. Versteeg, S. Boztas, and A. Rao, "Detecting anomalous user behavior using an extended isolation forest algorithm: an enterprise case study," *arXiv preprint arXiv:1609.06676*, 2016.

[47] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ser. HotNets '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 50–56. [Online]. Available: https://doi.org/10.1145/3005745.3005750

[48] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128–135, 2010.

[49] X. Bu, J. Rao, and C. Xu, "A reinforcement learning approach to online web systems auto-configuration," in *2009 29th IEEE International Conference on Distributed Computing Systems*, 2009, pp. 2–11.

[50] M.-F. Balcan, A. Blum, P. P. Choi, J. Lafferty, B. Pantano, M. R. Rwebangira, and X. Zhu, "Person identification in webcam images: An application of semi-supervised learning," in *ICML 2005 Workshop on Learning with Partially Classified Training Data*, vol. 2. Citeseer, 2005, p. 6.

[51] M. Mahdaviani and T. Choudhury, "Fast and scalable training of semi-supervised crfs with application to activity recognition," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, ser. NIPS'07. Red Hook, NY, USA: Curran Associates Inc., 2007, p. 977–984.

[52] Y. Suzuki, H. Takamura, and M. Okumura, "Application of semi-supervised learning to evaluative expression classification," in *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 502–513.

[53] A. Astorino, A. Fuduli, M. Gaudioso, and E. Vocaturo, "A multiple instance learning algorithm for color images classification," in *Proceedings of the 22nd International Database Engineering & Applications Symposium*, ser. IDEAS 2018. New York, NY, USA:

Association for Computing Machinery, 2018, p. 262–266. [Online]. Available: https://doi-org.ezproxy.federation.edu.au/10.1145/3216122.3216144

[54] ——, "Multiple instance learning algorithm for medical image classification," 2019.

[55] A. Astorino, A. Fuduli, P. Veltri, and E. Vocaturo, "Melanoma detection by means of multiple instance learning," *Interdisciplinary Sciences: Computational Life Sciences*, vol. 12, no. 1, pp. 24–31, Mar 2020. [Online]. Available: https://doi.org/10.1007/s12539-019-00341-y

[56] A. Zakaryazad and E. Duman, "A profit-driven artificial neural network (ANN) with applications to fraud detection and direct marketing," *Neurocomputing*, vol. 175, pp. 121 – 131, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231215015015

[57] M. A. Ali, D. Svetinovic, Z. Aung, and S. Lukman, "Malware detection in android mobile platform using machine learning algorithms," in *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, Dec 2017, pp. 763–768.

[58] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang, "Machine learning based mobile malware detection using highly imbalanced network traffic," *Information Sciences*, vol. 433-434, pp. 346 – 364, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025517307077

[59] J. Cannady, "Artificial neural networks for misuse detection," in *National information systems security conference*, vol. 26. Baltimore, 1998, pp. 443–456.

[60] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," in *Proceedings of the 2004 ACM symposium on Applied computing*, 2004, pp. 420–424.

[61] J. Goodman and D. Heckerman, "Fighting spam with statistics," *Significance*, vol. 1, no. 2, pp. 69–72, 2004.

[62] S. Mukherjee and N. Sharma, "Intrusion detection using naive bayes classifier with feature reduction," *Procedia Technology*, vol. 4, pp. 119 – 128, 2012, 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212017312002964

[63] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar 1986. [Online]. Available: https://doi.org/10.1007/BF00116251

[64] ——, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[65] R. Amin, J. Ryan, and J. Van Dorp, "Detecting targeted malicious email," *IEEE Security & Privacy*, vol. 10, no. 3, pp. 64–71, 2012.

[66] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, Sep 1999.

[67] K. Ghanem, F. J. Aparicio-Navarro, K. G. Kyriakopoulos, S. Lambotharan, and J. A. Chambers, "Support vector machine for network intrusion and cyber-attack detection," in *2017 Sensor Signal Processing for Defence Conference (SSPD)*, 2017, pp. 1–5.

[68] E. Kabir, J. Hu, H. Wang, and G. Zhuo, "A novel statistical technique for intrusion detection systems," *Future Generation Computer Systems*, vol. 79, pp. 303 – 318, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17301371

[69] C. S. Dangi, R. Gupta, and G. S. Chandel, "Article: Cyber security approach in web application using svm," *International Journal of Computer Applications*, vol. 57, no. 20, pp. 30–34, November 2012, full text available.

[70] Y. Kaya and m. F. Ertuğrul, "A novel approach for spam email detection based on shifted binary patterns," *Security and Communication Networks*, vol. 9, no. 10, pp. 1216–1225, 2016, sCN-14-0580.R2.

[71] H. Brahmi, I. Brahmi, and S. Ben Yahia, "Omc-ids: At the cross-roads of olap mining and intrusion detection," in *Advances in Knowledge Discovery and Data Mining*, P.-N. Tan, S. Chawla, C. K. Ho, and J. Bailey, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 13–24.

[72] A. M. Bagirov, N. Karmitsa, and S. Taheri, *Partitional Clustering Via Nonsmooth Optimization: Clustering Via Optimization*. Springer Nature, 2020.

[73] M. Blowers and J. Williams, *Machine Learning Applied to Cyber Operations*. New York, NY: Springer New York, 2014, pp. 155–175.

[74] M. Xie, J. Hu, S. Han, and H. Chen, "Scalable hypergrid k-nn-based online anomaly detection in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1661–1670, 2013.

[75] S. Taheri, A. M. Bagirov, I. Gondal, and S. Brown, "Cyberattack triage using incremental clustering for intrusion detection systems," *International Journal of Information Security*, vol. 19, no. 5, pp. 597–607, Oct 2020. [Online]. Available: https://doi.org/10.1007/s10207-019-00478-3

[76] S. Seifollahi, A. Bagirov, R. Layton, and I. Gondal, "Optimization based clustering algorithms for authorship analysis of phishing emails," *Neural Processing Letters*, vol. 46, no. 2, pp. 411–425, Oct 2017. [Online]. Available: https://doi.org/10.1007/s11063-017-9593-7

[77] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, Sep. 2008.

[78] W. Li, "Using genetic algorithm for network intrusion detection," *Proceedings of the United States department of energy cyber security group*, vol. 1, pp. 1–8, 2004.

[79] A. S. Sendi, M. Dagenais, M. Jabbarifar, and M. Couture, "Real time intrusion prediction based on optimized alerts with hidden markov model," *Journal of networks*, vol. 7, no. 2, p. 311, 2012.

[80] M. Tang, B. S. U. Mendis, D. W. Murray, Y. Hu, and A. Sutinen, "Unsupervised fraud detection in medicare australia," in *Proceedings of the Ninth Australasian Data Mining Conference - Volume 121*, ser. AusDM '11. AUS: Australian Computer Society, Inc., 2011, p. 103–110.

[81] X. A. Hoang and J. Hu, "An efficient hidden markov model training scheme for anomaly intrusion detection of server applications based on system calls," in *Proceedings. 2004 12th IEEE International Conference on Networks (ICON 2004) (IEEE Cat. No.04EX955)*, vol. 2, 2004, pp. 470–474 vol.2.

[82] J. Hu, X. Yu, D. Qiu, and H. Chen, "A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection," *IEEE Network*, vol. 23, no. 1, pp. 42–47, January 2009.

[83] D. Iyer, A. Mohanpurkar, S. Janardhan, D. Rathod, and A. Sardeshmukh, "Credit card fraud detection using hidden markov model," in *2011 World Congress on Information and Communication Technologies*, 2011, pp. 1062–1066.

[84] C. Annachhatre, T. H. Austin, and M. Stamp, "Hidden markov models for malware classification," *Journal of Computer Virology and Hacking Techniques*, vol. 11, no. 2, pp. 59–73, May 2015. [Online]. Available: https://doi.org/10.1007/s11416-014-0215-x

[85] T. H. Austin, E. Filiol, S. Josse, and M. Stamp, "Exploring hidden markov models for virus analysis: A semantic approach," in *2013 46th Hawaii International Conference on System Sciences*, 2013, pp. 5039–5048.

[86] M. Gharacheh, V. Derhami, S. Hashemi, and S. M. H. Fard, "Proposing an hmm-based approach to detect metamorphic malware," in *2015 4th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*, 2015, pp. 1–5.

[87] M. Imran, M. T. Afzal, and M. A. Qadir, "Malware classification using dynamic features and hidden markov model," *Journal of Intelligent & Fuzzy Systems*, vol. 31, no. 2, pp. 837–847, 2016.

[88] A. Sarmanova and S. Albayrak, "Alleviating class imbalance problem in data mining," in *21st Signal Processing and Communications Applications Conference (SIU)*, April 2013, pp. 1–4.

[89] S. Vluymans, D. Tarraga, Y. Saeys, C. Cornelis, and F. Herrera, "Fuzzy rough classifiers for class imbalanced multi-instance data," *Pattern Recognition*, vol. 53, pp. 36 – 45, 2016.

[90] H. Guo, Y. Li, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220 – 239, 2017.

[91] B. Krawczyk, M. Galar, Łukasz Jeleń, and F. Herrera, "Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy," *Applied Soft Computing*, vol. 38, pp. 714 – 726, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494615005815

[92] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, July 2012.

[93] H. Haibo, "Introduction," in *Imbalanced Learning: foundations, algorithms, and applications*. Wiley-Blackwell, 2013, ch. 1, pp. 1–12.

[94] M. Koziarski, B. Krawczyk, and M. Woźniak, "Radial-based oversampling for noisy imbalanced data classification," *Neurocomputing*, vol. 343, pp. 19 – 33, 2019, learning in the Presence of Class Imbalance and Concept Drift. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231219301596

[95] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, June 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622407.1622416

[96] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, June 2008, pp. 1322–1328.

[97] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *Advances in Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 878–887.

[98] A. C. Liu, "The effect of oversampling and undersampling on classifying imbalanced text datasets," *The University of Texas at Austin*, 2004.

[99] X. Y. Liu, J. Wu, and Z. H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, April 2009.

[100] G. Nguyen, A. Bouzerdoum, and S. Phung, "A supervised learning approach for imbalanced data sets," in *2008 19th International Conference on Pattern Recognition*, December 2008, pp. 1–4.

[101] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, April 2012.

[102] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229 – 242, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231212006479

[103] K. Li, X. Kong, Z. Lu, L. Wenyin, and J. Yin, "Boosting weighted elm for imbalanced learning," *Neurocomputing*, vol. 128, pp. 15 – 21, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213009971

[104] Y. Zhang, B. Liu, J. Cai, and S. Zhang, "Ensemble weighted extreme learning machine for imbalanced data classification based on differen-

tial evolution," *Neural Computing and Applications*, vol. 28, no. 1, pp. 259–267, 2017.

[105] Xulei Yang, Qing Song, and A. Cao, "Weighted support vector machine for data classification," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, July 2005, pp. 859–864 vol. 2.

[106] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, July 2010, pp. 1–8.

[107] R. O'Brien and H. Ishwaran, "A random forests quantile classifier for class imbalanced data," *Pattern Recognition*, vol. 90, pp. 232 – 249, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320319300536

[108] B. S. Raghuwanshi and S. Shukla, "Generalized class-specific kernelized extreme learning machine for multiclass imbalanced learning," *Expert Systems with Applications*, vol. 121, pp. 244 – 255, 2019.

[109] ——, "Class-specific kernelized extreme learning machine for binary class imbalance learning," *Applied Soft Computing*, vol. 73, pp. 1026 – 1038, 2018.

[110] ——, "Underbagging based reduced kernelized weighted extreme learning machine for class imbalance learning," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 252–270, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197618301465

[111] S. Fotouhi, S. Asadi, and M. W. Kattan, "A comprehensive data level analysis for cancer diagnosis on imbalanced data," *Journal of Biomedical Informatics*, vol. 90, p. 103089, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1532046418302302

[112] L. Chen, G. Xu, Q. Zhang, and X. Zhang, "Learning deep representation of imbalanced scada data for fault detection of wind turbines," *Measurement*, vol. 139, pp. 370 – 379, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0263224119302386

[113] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen, "Effective detection of sophisticated online banking fraud on extremely imbalanced data," *World Wide Web*, vol. 16, no. 4, pp. 449–475, Jul 2013. [Online]. Available: https://doi.org/10.1007/s11280-012-0178-0

[114] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 434–443, June 2013.

[115] D. Goodin. (2018) Google chrome extensions with 500,000 downloads found to be malicious. [Online]. Available: https://arstechnica.com/information-technology/2018/01/500000-chrome-users-fall-prey-to-malicious-extensions-in-google-web-store/

[116] M. Mimoso. (2017) Google removes chrome extension used in banking fraud. [Online]. Available: https://threatpost.com/google-removes-chrome-extension-used-in-banking-fraud/127469/

[117] A. Bagirov, "Max-min separability," *Optimization Methods and Software*, vol. 20, pp. 277 – 296, 2005.

[118] A. Bagirov, J. Ugon, and D. Webb, "An efficient algorithm for the incremental construction of a piecewise linear classifier," *Information Systems*, vol. 36, pp. 782 – 790, 2011.

[119] A. Bagirov, J. Ugon, D. Webb, and B. K. zen, "Classification through incremental max-min separability," *Pattern Analysis and Applications*, vol. 14, pp. 165–174, 2011.

[120] A. Bagirov, N. Karmitsa, and M. Mäkelä, *Introduction Nonsmooth Optimization*. Cham, Springer, 2014.

[121] A. Bagirov, B. Karasozen, and M. Sezer, "Discrete gradient method: derivative-free method for nonsmooth optimization," *Journal of Optimization Theory and Applications*, vol. 137, pp. 317–334, 2008.

[122] A. Bagirov, "Continuous subdifferential approximations and their applications," *Journal of Mathematical Sciences*, vol. 115, no. 5, pp. 2567–2609, 2003.

[123] A. Bagirov, N. Karmitsa, and S. Taheri, "Discrete gradient methods," in *Numerical Nonsmooth Optimization*, A. Bagirov, M. Gaudioso, N. Karmitsa, M. Mäkelä, and S. Taheri, Eds. Cham, Springer, 2020, pp. 621–654.

[124] A. Bagirov and J. Yearwood, "A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems," *European Journal of Operational Research*, vol. 170, no. 2, pp. 578–596, 2006.

[125] A. Bagirov, "Modified global k-means algorithm for minimum sum-of-squares clustering problems," *Pattern Recognition*, vol. 41, no. 10, pp. 3192–3199, 2008.

[126] Z. Ding, "Diversified ensemble classifiers for highly imbalanced data learning and its application in bioinformatics," Ph.D. dissertation, College of Arts and Sciences, Atlanta, GA, USA, 2011, aAI3486649.

[127] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[128] J. Alcala-Fdez, A. Fernández, J. Luengo, J. Derrac, S. Garc'ia, L. Sanchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, pp. 255–287, 01 2010.

[129] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[130] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, 2013, pp. 4487–4492.

[131] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "Drebin: Effective and explainable detection of android malware in your pocket," in *28th Annual Network and Distributed System Security Symposium (NDSS)*, February 2014.

[132] M. Spreitzenbarth, F. Echtler, T. Schreck, F. C. Freiling, and J. Hoffmann, "Mobilesandbox: Looking deeper into android applications," in *28th International ACM Symposium on Applied Computing (SAC)*, March 2013.

[133] M. S. Al-Daweri, K. A. Zainol Ariffin, S. Abdullah, and M. F. E. Md. Senan, "An analysis of the kdd99 and unsw-nb15 datasets for the intrusion detection system," *Symmetry*, vol. 12, no. 10, 2020. [Online]. Available: https://www.mdpi.com/2073-8994/12/10/1666