Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

Guilherme Jorge Cardoso

Metodologias para o uso de aprendizagem automática na classificação de entidades de rede com base em padrões de tráfego

Methodologies for machine learning classification of network entities based on traffic patterns

Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

Guilherme Jorge Cardoso

Metodologias para o uso de aprendizagem automática na classificação de entidades de rede com base em padrões de tráfego

Methodologies for machine learning classification of network entities based on traffic patterns

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Professor Doutor Paulo Jorge Salvador Serra Ferreira, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professora Doutora Susana Isabel Barreto de Miranda Sargento
Professora Associada C/ Agregação, Universidade de Aveiro

vogais / examiners committee

Professor Doutor Rui Jorge Morais Tomaz Valadas
Professor Catedrático, Universidade Técnica de Lisboa

Professor Doutor Paulo Jorge Salvador Serra Ferreira
Professor Auxiliar, Universidade de Aveiro

agradecimentos /
acknowledgements

Seria impossível ignorar o grande obrigado à minha família. Sem eles jamais teria conseguido a conclusão deste curso. À Paula Moreira, por acreditar até mais que eu neste caminho que revelou-se ser mais do que o certo, pela paciência e cumplicidade ao longo de todos estes anos que de outra forma jamais estaria aqui. Quero agradecer também ao professor Paulo Salvador por toda a orientação, dentro e fora da dissertação e no debate de tantos outros temas que de uma forma ou de outra direcionaram-me no percurso que hoje sigo. Estou extremamente agradecido aos meus amigos do #elite, por me mostrarem que por mais que as coisas mudem são uma constante fonte de apoio e alegria. Ao José Moreira, por me fazer descobrir um rumo que não imaginaria, Tiago Oliveira pela companhia, ao Eduardo e ao Marco, a todos os colegas do #workers, e a tantos outros que fizeram parte desta experiência académica.

Palavras Chave

reconhecimento de padrões, analise comportamental, arquitetura de aprendizagem automática, modelação de atividade de rede, modelação de atividade humana em redes, classificação de entidade de redes

Resumo

Nos últimos anos notícias sobre roubos e perdas de informação e de dados têm sido constante, levantando discussão sobre a segurança dos sistemas dos quais hoje dependemos. As comunicações são também cada vez mais privadas, pelo que os sistemas de segurança de última geração têm desenvolvido técnicas de reconhecimento de padrões para detetar e inferir a segurança sem a necessidade de processar conteúdos. Esta dissertação propõe metodologias para inferir os padrões de entidades considerando o seu tráfego de rede: se está enquadrado no comportamento de tráfego previamente conhecido, ou se a atividade gerada é incomum e, por isso, ser indicação de um possível problema. Há uma forte indicação de que o reconhecimento de padrões de comportamento continuará a liderar a investigação no domínio de soluções de segurança, não só para o tráfego de rede, mas também para outras atividades mensuráveis. Outros exemplos englobam a gestão de acesso de identidade ou programas em execução em um computador. As metodologias propõem a modelação de metadados da camada de rede OSI 3 a 5 em contagens que são posteriormente processadas por algoritmos de aprendizagem automática para classificar a atividade da rede. Esta classificação baseia-se em dois níveis: no primeiro o reconhecimento entidades ativas dentro de um domínio de rede e o segundo, se cada entidade corresponde ao padrão conhecido. As metodologias apresentadas para inferir se algo está de acordo com padrões conhecidos são transversais a outros domínios. Embora a agregação de metadados e modelação seja diferente, o processo descrito para inferir padrões é genérico o suficiente para ser aplicado a outros casos de uso, de rede ou não, ou ainda combinado em cenários mais complexos. O último capítulo inclui uma prova de conceito com dados sintéticos e algumas métricas de avaliação, para perceber se os algoritmos de classificação podem distinguir com sucesso padrões diferentes. Os testes mostraram resultados promissores, variando de 99% para classificação de entidades e 77% para 98% (dependendo da natureza da entidade) para deteção de anormalidades.

Keywords

pattern recognition, behavior analysis, network activity, machine learning pipeline, network behavioral modeling, human network behavioral modeling, network entity classification

Abstract

For the last years, constant news about information and data leaks are raising public discussion of the safety of the systems that we all nowadays depend on. Communications are increasingly more private; hence next-generation security systems rely on pattern recognition techniques to detect and infer the safety without the need for scrapping its content. This dissertation proposes methodologies to infer entity patterns and their nature according to their network traffic: if they are running according to their previously known safe pattern or if its behavior is uncommon, an indication of a possible breach. There is a strong indication that behavioral pattern recognition will continue to lead the research of security solutions, not only for the network traffic but also for other measurable activities. Other examples are identity access management or programs running on a computer. This dissertation proposes modeling network OSI layers 3 to 5 metadata in features that are later processed by machine learning algorithms to classify the network activity. The classification itself is divided into two groups: the first level is recognizing active entities operating within a network domain and the second if each entity is acting according to each known pattern. The presented methods of inferring if something is acting according to known patterns are transversal to other domains. Although aggregation of metadata and modeling differ, the described process of solving the problem of inferring patterns is generic and can be applied to user use cases rather than to the network, or combined with more complex scenarios. The last chapter includes a proof of concept with a few evaluation metrics using synthetic data, to evaluate if the classification algorithms can successfully distinguish different patterns. The tests showed promising results, ranging from 99% for entity classification and 77% to 98% (depending on the entity nature) for abnormality detection.

Contents

List of Figures

List of Tables

Glossary

ASR	Automatic Speech Recognition
ML	Machine Learning
OCR	Optical Character Recognition
AI	Artifficial Intelligence
SVM	Support Vector Machine
EC	European Commission
DDoS	Distributed Denial of Service
DNS	Domain Name Service
IoT	Internet of Things
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
SA	Situation Awareness
CSA	Cyber Situational Awareness
ECSA	Effective Cyber Situational Awareness
NSA	Network Situation Awareness
NSSA	Network Security Situation Awareness
IDS	Intruction Detection System
NN	Neural Network
EU	European Union
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
TAP	Terminal Access Point
ISP	Internet Service Provider
DPDK	Data Plane Development Kit
SDN	Software-defined Networking
NFV	Network function virtualization
CEO	Chief Executive Officer
PCA	Principal Component Analysis
QoS	Quality of Service
MLP	Multi-layer Perceptron
DTC	Decicion Tree Classifier
OvO	One-vs-One
OvR	One-vs-Rest
ROC	Receiver Operating Characteristic
PDF	Probability Density Function
PoC	Proof of Concept

CHAPTER 1

Introduction

When one thinks about the typical corporate organization structure, the traditional hierarchy in the form of a pyramid comes to mind. This chain starts with Chief Executive Officers (CEOs) or the association that owns the business on the top, followed by the executive president, vice presidents, and so on. There's a definite image of a top-down model of a chain of command down that goes from the president down to the engineers. Each layer also has its objective in the overall view of running a business: the ultimate goal of CEOs is to keep the company healthy and running, trying to guarantee the capitalization of the whole operation. In a layer below, executive officers have a different role, trying to manage ways to keep this goal of capitalization possible. The chain continues down to the engineers who implement the base of the applicational product. In this whole structure, management policies try to define each one's responsibility and tasks, guiding working habits. Usually, the bigger the corporation is, more protocols and policies exist. It becomes harder to manage so many people, especially for identity and access management. It is understandable the vast investment and effort in just managing people, and it is not by accident that *IBM X-Force® Research 2016 Cyber Security Intelligence Index* found out that 60% of all corporate attacks were led by insiders, being three-quarters of malicious intent and one-quarter involved inadvertent actors. The primary affected industries of inside attacks include healthcare, manufacturing, and financial services, due to the value of personal data, intellectual property, and massive financial assets. Despite the fact that these industries differ almost in everything, including their working field, politics, management policies and so on, the primary shared ground is people - all of whom have the potential to be an inside threat [1].

The last example is just one of the vectors threatening corporations. Each business has their threats and challenges: it may be competition, market shifts that corporate needs to quickly adapt, lack of research or even bad investments. However, new kinds of threats are surfacing with the continuous digitalization and increased assets and people mobility. Technology is affecting and empowering a digital economy, by allowing us to work smarter, faster and more safely. The enrichment of digital devices, networks, and the innovations are

now inherent in the way we live and do business. Nevertheless, to archive the full potential of a digital revolution, we need to trust the capability of these systems to provide our safety, security, and privacy. In the past several years highly publicized cybersecurity events brought the focus of people in general. These events, discussed in Section 2.1.1, are shaping the trustworthiness of what is believed to be the fourth digital revolution [2].

Like any of the revolutions that preceded it, this "fourth revolution" comes with changes and opportunities, including the social effects in a world of data and mobility, new labor conditions and the organization of labor itself. Technological developments also impact the standards of living, disrupting the concepts of privacy (and especially data privacy) since everything nowadays can be permanently recorded, correlated and inferred, in an era that there is more information than ever. Significant government efforts are being made to define new standards of privacy rights, as discussed in Section 2.1.3. The new digitization and data collection, reviewed in Section 2.2, allows the possibility of inferring information in ways that were previously not possible.

Also, as described in Section 2.3, it is possible to compile new information (and therefore knowledge) via correlation and massive data mining, resorting to "The Fourth Industrial Revolution" tools (such as Machine Learning (ML), discussed on Section 2.4). These tools can create solutions capable of moderately level awareness and can create new solutions that improve security, integrity, and availability of the systems that we now depend on. With this in mind, as Figure 1.1 shows, at a network level, there are entities that generate traffic, either human or machine to machine communication, internal or external. Each one, by definition, has its tasks that impact network activity in a specific way. This strategy of monitoring the generated activity itself and not directly the entity gives the opportunity of having security consciousness without interfering with privacy or limiting user freedom. Can be seen as a transversal macroscopic analysis resorting on the network.

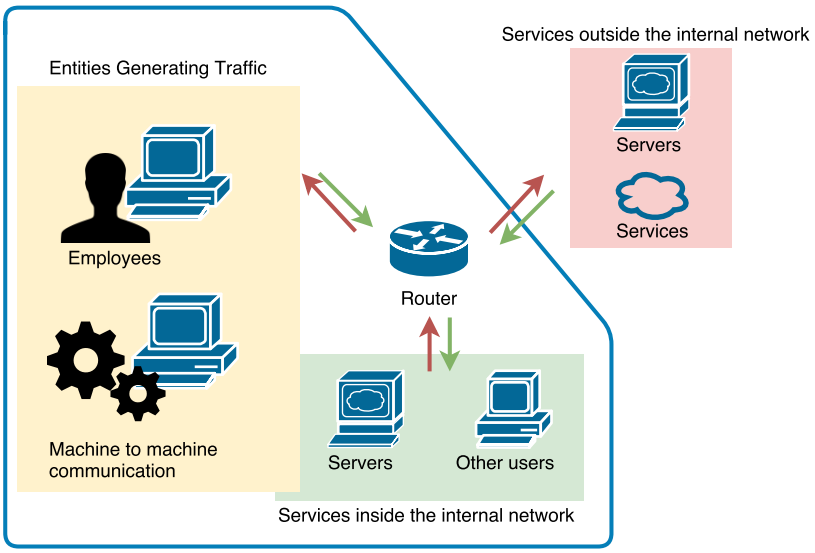

Figure 1.1: Representation of network entities (human or machine) generating traffic within an enterprise network environment

The work developed in this dissertation proposes a set of methodologies and strategies that together set an architecture which tries to identify corporate threats emerging due to data mobility and availability, taking into account the increasing rules of user and data privacy. The possibility of using ML as a tool to reveal the subtle patterns of each entity activity that identifies each one typical behavior is a solution compatible with corporations by not restricting or invading employees privacy. Information gathered from learning and recognizing these activity patterns ultimately provide a new level of awareness of the security situation in such environments. Such security, known as Network Security Situation Awareness (NSSA) and discussed in Section 2.5.3, can identify each entity/user/employee normal activity behaviors, and therefore revealing compromised services, machines or even exposing insiders. Lastly, due to the nature of ML, this approach used recursively can identify sub-activities, revealing in depth what activity is usual and which is not, rather than flagging the whole entity as misbehaving from the pattern. Developed work in this dissertation includes two analysis in two layers. The first more generalistic discussed in Section 3.4.2, is classifying network entities within a network space and thus proving a high-level view of the network players, providing entity identity authenticity by recognizing its behavior and labeling it to a known matching profile and thus working at a networking overview level. The second, more in-depth layer is recognizing and tracking entities activity and matching it to its known normal behavior, which in turn provides misbehaving or suspecting activity - discussed in Section 3.4.3.

The testbed setup and the whole infrastructure to generate pseudo-synthetic data was kindly provided by *IT - Instituto de Telecomunicações* at Universidade de Aveiro. The complete collection tools, data set, preprocessing methods, ML training processes, evaluations, results, testings and data structures are opensource and available at https://github.com/luminoso/th-ml-patternrecognition/.

CHAPTER 2

State of Art

Corporations have always been under the threat of industrial espionage, information theft, and sabotage, to disrupt the economy and competition from inside. Until the usage of computers and the internet, these kinds of attacks manifested somewhat in a physical context: the source had to be someone or something visible. Business security came from trust and ultimately by a perimeter. However, data stopped to have a physical format. Theft of information is now possible and imperceptible to human senses, requiring new defense mechanisms. In this chapter, an overview of the current security situation, with examples of these new (cyber) type of attacks and their increasing impact on the economy, reactions, and efforts to avoid it. It is presented how the use of tools such as ML to process more-than-ever amounts of monitoring data, trying to archive NSSA to provide security of data, people, and businesses.

2.1 Widespread Security view

2.1.1 Next generation threats

Each year, information and data leakage sets new records, and it has been one of the most prominent ways of threatening corporations, with high-profile attacks that have a direct influence on global politics and economics. Notable attacks include the leaked documents from the Panamanian law firm Mossack Fonseca, which exposed offshore accounting of thousand of prominent people around the globe. Dubbed as "Panama Papers", the leakage included financial information of several ongoing and former USA state heads, friends, and family. The whole range of people included celebrities and business people[3]. While this leak did not reveal any illegal content per se, they raised suspicion due to the nature of offshore accounts being used for tax evasion and money laundering, leading to criminal investigations in 79 countries[4], and driving to anti-government protests in several countries including Pakistan[5] and UK[6]. In April 2016, the Prime Minister of Iceland stepped down from office after the leak[7].

Another significant data leak attack occurred with Equifax[1], considered by many the most significant security breach in history[8]–[10], in which more than 143 million records of American citizen personal data[2] were stolen. Databases were downloaded over a few months without being detected, by using a known unpatched vulnerability[11].

Even though 2017 unprecedented level of attacks and disruption, like Equifax breach, attackers often used simple tools and tactics to successfully archive their objective, rather than practicing exploitation complex Zero-day vulnerabilities or using sophisticated malware. Instead, the usage of straightforward approaches such as phishing emails, legitimate network administration tools or even operating system features is common[12]. The rationale for performing attacks, by investing time and money is also lucrative since industrial espionage is more profitable that research and development, which initiates an economic incentive to cyber espionage[13].

Not all attacks include theft of information. Service disruption also affects corporations reputation and obviously their revenue stream. Mirai's botnet caused disruption of the internet in the United States by deploying a Distributed Denial of Service (DDoS) to one of the biggest DNS providers. As Figure 2.1 shows, the attack led to *Level(3) Communications* Internet Service Provider (ISP) internet disruption over several hours, making many services unavailable nation-wide [14].

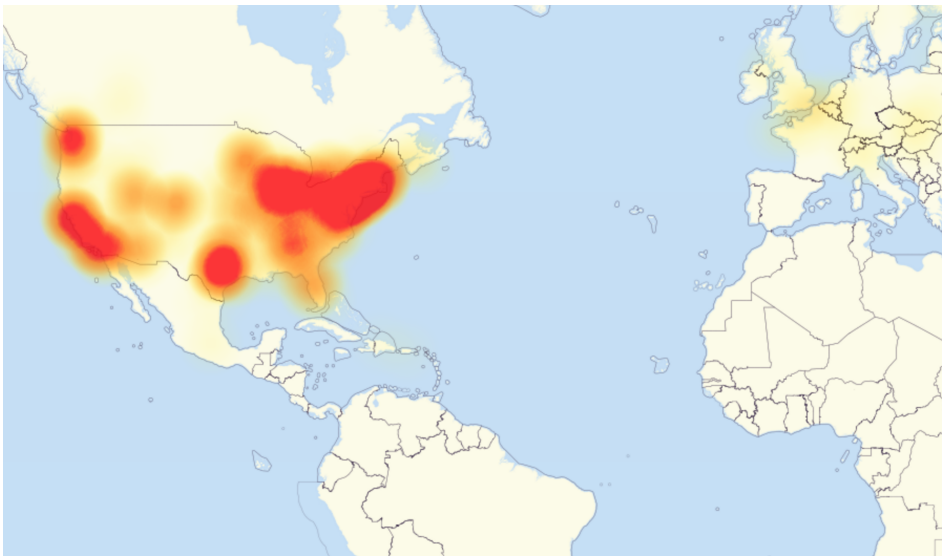

Figure 2.1: Map of areas most affected by DNS attack and without DNS resolution services, 16:45 UTC, 21 October 2016

The attack proliferated without any sophisticated dictionary attacks or brute-force, by using just Internet of Thingss (IoTs)[3] default device's passwords, ranging from an infection rate from 1% to 12% in some countries[15]–[18]. Although this kind of devices often has low processing power, the disperse nature of such infection led to the ideal and unstoppable

[1]Private company that collects and computes credit card scores in United States

[2]Including full names, Social Security numbers, birth dates, addresses, and, in some cases, driver license numbers

[3]Mostly home routers and home IP cameras

DDoS attack. It affected not only DNS providers but also to considerable web hosting providers like OVH and other services[19]. It was later investigated by the United States Department of Homeland Security, and up to today the investigators still have no idea who carried out the cyber attack[20]. These examples raise the question of the availability and unpredictability of performing disruptive attacks, how easy it looks to provoke service disruption and unavailability, which may cause economic losses. The feeling of being helpless when they occur boosts the discussion of computer system security and reliability that we, day by day, depend more upon.

2.1.1.1 Vector of the IT-oriented threats

Technology companies are under immense pressure to innovate and release products to the market quickly. This accelerated development is often at the expense of cybersecurity, where being "first to the market" is a higher priority than being "secure to market". Development and implementation of security features take time and money, and as market competitiveness continue to intensify, companies try to avoid any risk of delaying the "development and release" cycle of a product. With less and less flexibility of tolerating these kind of release constraints, security features and implementation are often delayed to subsequent versions of the products, despite de fact of doing so results in a weak implementation and architecture when compared to a product that has security taken in mind in the initial design.

As the security measures are tightened, innovations in defense are often in pair with the offense. One example is the usage of ML (Section 2.4) to correctly identify new attack vectors, that will undoubtedly also be used for the improvements of the attack vectors for malicious activities [21].

Nonetheless, attackers have often the advantage: frequently composed of highly skilled and motivated people that can spend months in preparation and plan to carry out and intrusion. Attacks may be sponsored by nations, criminal organizations or just activist groups that invest the time to deploy such act. Organizations have the burden of blocking all those attempts, knowing that all it takes is one lucky successful attempt to compromise the whole infrastructure of a corporation[22], [23].

As the computer systems continue to grow in power and number, software complexity is also increasing. The need of interoperate and to seamlessly integrate with different components and systems in ubiquitous computing is exposing more attack vectors and entry points. Managing all vulnerabilities with such a complicated scenario becomes a challenge to security experts. The rising of IoT devices that often have an antiquated, release-oriented software development cycle becomes a nightmare to manage due to the fact of exploding combination of software and hardware exposed to attacks.

Ultimately, the success of a transformation to a Digital Economy relies on the trust that organizations and corporations can meet the demands of protecting the customer. Guarding user data, network infrastructure, intellectual property without any incidents is the top priority to ensure people trust the continuous digitalization of society. As the dependency on

technology continues to increase, concerns about the consequences of data manipulation and success of attackers create new challenges for the cybersecurity to address.

To sum up, with the emergence of communication in a world where people choose to be connected and linked to an enormous range of devices, the definition and the state of cybersecurity is gaining a major role in the tech world. To void a safe cyber state of a machine or service can range from its development or even to intentional abuse, as discussed in 2.1.2.

2.1.2 Social-oriented threats

Although the immense range of possible threats to business corporations, current working habits are also a challenge for cybersecurity. The concept of employees working in a fixed place or a job in the company headquarters is changing to employees working with mobility. Personal devices that allow access from any location at any time are now a standard, originating a challenge on how should one keep the data secured. The need of improving interoperability of different systems between vendors, partners, and customers are inevitably removing barriers to access to information. The definition of the classic concept that if something needs to be secure, then a perimeter set is now obsolete, as devices, people and even networks that access data are no longer in physical control - most of these things are now remote. Nonetheless, attempts of malicious activity try to take advantage not only changing methodologies of work but also of the reluctance to prioritize basic cybersecurity practices or staff training. Failure of mitigating primary attack vectors allow attempts of malicious actors of any expertise levels to perform successful attacks.

In a study published by the Defense Personnel Security Research Center of the United States Department of Defense, found that globally there's an increase in the opportunity and motivation for espionage. People now have an unprecedented level of access to sensitive information due to the technological advances and data availability, being able to store and retrieve such information. This opportunity of easily establishing contact and transfer colossal volume of information effortlessly allows "insiders" to more types of information globally. The opportunity for this kind of crime increases in the same magnitude as computers are becoming more linguistically accessible and heterogeneous. Additionally, loyalty to organizations is diminishing, as the globalization of employees has emotional and financial ties to other countries, making them less inclined to have moral dilemmas when conducting espionage or even considering it justifiable if sharing in the benefit of "world community" or preventing conflicts[24].

Nonetheless, industrial espionage can also work in a social, legal way. These methods include the purchase of companies or products, having the results of inevitably transferring the technology or knowledge of a competitor, leaving almost nothing to avoid this kind of "attack" to a country or business, working as information acquisition. Other methods include, for example, when evaluating the cost of expansion to other countries or just by scaling up. Training new workers to use critical technology that may later leave their current position to competition. Corporations must evaluate the kind of risk of what is worth to expose business intelligence with the collaboration of security managers, although the final decision is not

made by the security but by the CEOs. The hiring of key employees from the competition results in the inevitable transfer of knowledge to a competitor even if employees do not intend to divulge sensitive information. To avoid such leakage, tech giants like Google, Apple and Adobe often sign deals of antipoaching, that each company is not allowed to hire engineers from each other. This leads to antipoaching deals that often result in lawsuits in the order of the millions of dollars[25]. As a final example, there are also legal approaches to legitimately access information of a competitor, including the practice of joint ventures or analyze annual reports. Patent submissions also provide a vast opportunity for knowing the current state of the art of the technology developed internally [26]. The concept of legitimate social-oriented attacks is so abstract that researchers sometimes have difficulty to define what is intelligence gathering or industrial espionage. Knowing the precise definition of legitimate intelligence gathering versus the industrial espionage - what practices make each one[27].

The repercussions of the fear of industrial espionage via social engineering lead to attempts to stop such activities. Highly controlling work environments that try to stop espionage, usually deploy complicated security measures and intensive employee monitoring that create distrust in the workspace, as described in Figure 2.2 [28].

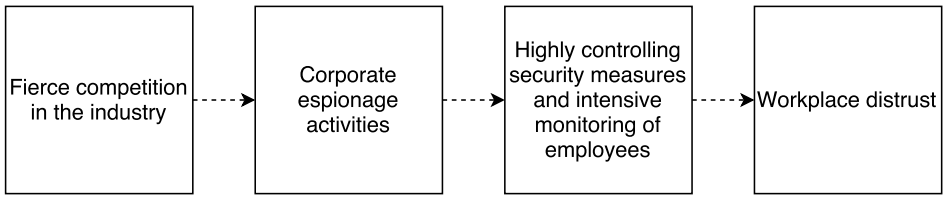

Figure 2.2: Relationship of corporate competition and how it leads to work distrust

Such suspicion in working spaces eventually reduces employee loyalty and respect to the company, creating a loop of possible new information-stealing attack vectors, since the moral and ethic dilemma of stealing from someone who does not trust their employees is lessen[28]. Consequently, research into security by other means has been shown possible, due to the data age, lead by monitoring in alternative ways as described Section 2.2. Awareness (Section 2.5) is therefore possible through the use of next-generation extraction tools (2.4), making possible to know if in fact there are indicators that lead to the suspicion and consequently investigation of cases of information leakage. These new techniques alleviate the problems created by highly restrictive policies, reinforcing the trust between the company and the worker.

In the same level of social-oriented attacks, the independent research group *The Information Warfare Monitor* concluded in an extensive and exhaustive investigation between June 2008 and March 2009 focused on allegations of Chinese cyber espionage against the Tibetan community. In a short version, the investigation led to the discovery of *GhostNet*, an espionage network spread in 103 countries of high-value systems. The list of infected systems included computers of ministries of foreign affairs, the ASEAN (Association of Southeast Asian Nations) Secretariat, SAARC (South Asian Association for Regional Cooperation), and the Asian Development Bank; news organizations; and an unclassified computer located at NATO headquarters. The vector of spreading the *GhostNet* infection leverages on social means, by

sending contextually relevant emails to targets with attached documents packed with the exploit code and the trojan. The successfulness of the network was then highly dependant on the social factor and staff training to detect and evade phishing attacks [29].

Social engineering can itself be in many forms. As used by spreading and growing of the *Ghostnet*, phishing is the most common and effective. Luring victims into revealing information based on the tendency to believe the security and trustworthiness of a brand, spies could also be insiders such as employees, ex-employees, vendors, consultants, and so on, leading the victim to expose secrets to a trustworthy entity - or person[30]. Although security training in order to detect and avoid phishing attacks works in most cases, it does not eliminate the risk[31]. Consequently, there is an apparent relationship between the classic social methods of information gathering and the digitalization of working habits. In one hand social engineering attacks need to take into account the methods of extracting information using these actors in the digital medium. On the other security needs to be able somehow to have the awareness of what is or what is not the typical activity to prevent the success of these attack attempts.

2.1.3 Formal view of security

Computer Security (or *IT Security*) defines that a system can only be secure if a set of policies delineating what information is available to be read or written by each entity is delineated. Those same policies need mechanisms capable of enforcing such policies, with the goal of preserving the integrity[4], confidentiality, and the disruption or misdirection usage [32], [33].

The process itself of determining if something is, in fact, secure is more profound than the policy itself. It requires the necessity of having evaluators who can accurately and consistently assess the state of the security of a computer/system/service/etc[32]. Considering that the mechanisms for enforcing such policies can range from different fields (for example having better administrative procedures or laws *versus* implement technical solutions) the technology of computer security becomes controversial and somewhat evolved over the years[32].

Due to the magnitude of the general *IT Security* term, it was segmented and it is now subdivided into many fields, including *Internet Security*[5][34]. Since 2013, it has been under burdensome regulation by the European Union (EU), that defines strategies and methodologies to combat cybercrime and better shape the cybersecurity principles[35]–[37].

According to Symantec Threat Report, cybercrime has grown every year, especially in 2016 with extraordinary multi-million dollar virtual bank heists, electoral process disruption, and the biggest DDoS recorded, causing unprecedented levels os disruption[12]. With this in mind, it is understandable why EU keeps accelerating the measures and regulation[38].

The interest in the field is also growing due to the comprehensiveness of the attacks itself. Although the information is often restricted or not publicly disclosed due to economic implications, or just because of corporate image. Two of the most famous known attacks in 2017 include a ransom attack of the HBO's episodes of Game of Thrones[39]. The episode

[4]Preventing damage, modification, destroying, etc.
[5]Also known as *Cyber Sec* or *Net Sec*.

larceny is not by accident. This show has an average of 30 million viewers[40] and a production cost estimated at 100 million dollars per season. The attackers stole the complete season demanding to HBO to pay in order not to release the episodes before the official premiere time.

The broad nature of the attacks is dividing *Internet Security* in two other subtopics: *Network Security* and *Information Security*. The first, combines the policies, practices to prevent the disruption, snooping, modification or denial of a computer network or network-accessible resources. It also evolves the mechanisms to enforce access (such IDs and passwords) or to the resources under the network authority, both public and private[41]. The second, is the process of preventing unauthorized access, modification or destruction of information. This definition is valid regardless of the medium of the format that the data has [42]. European Commission (EC) also developed regulation for the general data protection and it is now being implemented with the deadline of May/2018 for full implementation deadline, where the regulation on the protection of natural people concerning the processing of personal data and the free movement of such data [43].

Blending of *Information Security* and *Network Security* are the two topics that compose the main aspects of this thesis. Given the rapid changes and continuous growth of threats, current techniques are barely up to the level of such attacks, requiring new methods and approaches to ensure a holistic view security of a system. Studies suggest that bringing awareness to the network is the next generation approach to solve the next challenges of cybersecurity and data protection [44].

2.1.4 Organizational security assumptions

There's currently plenty of ideally implementations that aim to provide a full organizational security. For example, ISO/IEC 27000-series[6] composes information security standardization published by International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC)[45], in which best practices for implementing an information security management system within an organization, including the assessment of risks are specified. They are intended to be generic allowing it to accommodate the nature, size, and type of each organization[46].

The standardization is extensively wide and includes security aspects of the structure, policies, organization of information security, human resources, asset management and control, cryptography, physical, environmental security, communications, and many others [47], [48].

Alongside with the standardization, there is risk evaluation *threat models*, widely available and encouraged by the giants in the tech world. Microsoft's *S.T.R.I.D.E.*[7] mnemonic introduced in 1999, that intends to help developers with suggestions for practices and approaches when evaluating risk [49]. *PASTA*, takes a different approach, by defining a seven-stage process of a risk-centric methodology, aligning organizational business objectives with technical requirements, where a few of the main goals are to improve the visibility of cyber-threat

[6] Also known as the 'ISMS Family of Standards' or 'ISO27k' for short

[7] Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege

risks, extending the protection domains, reducing risk, collaboration with stakeholders, among others [49], [50].

Due to each *threat model* limited comprehensiveness and due to the scope under each was created, there are also proposals to combine multiple *threat models* and frameworks with the security standardization itself, providing a more extensive overall coverage[51].

Altogether, there is a wide availability of norms, frameworks, and tools organizations are implementing and using their own security model, covering preventive and reactive aspects of security. However, as threats are evolving, models are also shifting to a proactive philosophy, with the assumption that attackers already gained access to the network[52]. ML is helping to shape a new kind of network security health perspective by providing misuse and abnormality detection which was not yet possible via signature-based methods due to the nature of increasingly network complexity[44].

This thesis relies the on the assumption above that organizations follow the best practice regarding the security of an organization either via the standardization certification or thread models evaluation. It is intended to provide a more abstract comprehension of security with the concept of matching normal and abnormal network patterns via ML (2.4), facilitating the response time to yet unknown attack vector (or unusual), adapting to the age of volume, variety, and velocity of data[44] without the need of implementing new monitoring architectures.

2.2 Monitoring

On a corporate level, avoiding and detecting threats and especially preventing them is inherently related to monitoring. It happens at all levels of corporate activity in the most straightforward ways: the concierge who watches an entrance for known people, logging entering and leaving hours, access levels and clearance, among others. In the terms of business security, as discussed in Sections 2.1.2 and 2.1.1, Cyber activity is no different. Employees have often restricted activity authorization, and their activity monitored at some level. The ultimate goal is to have some awareness (as discussed in Section 2.5) to act when required. Monitoring allows knowing that something happened or at least trying to know that something happened.

When it comes to the network specifically, monitoring has a whole different approach. Unless in an environment with low volume traffic, there is some limitation of data network collection. It is essential to store the minimum set of information, since storing headers and contents of every packet is too resource-intensive. It is possible to perform initial traffic analysis by scanning the characteristics of packets and by ignoring the payload altogether. This analysis, however, has some limitations. For example, it is unable to verify if communication if a port matches the expected service (e.g., a Trojan using port 21 to disguise as FTP activity). In corporate environments, the focus of analysis typically insides in TCP, UDP and ICMP traffic. Obviously not despising other protocols in use in the network, but for discussion of monitoring concepts these big 3 represent the fundamental idea of such analysis which are:

- Source and destination IP addresses.

- For TCP or UDP traffic, the source, and destination ports.
- Only for TCP or UDP traffic, the source, and destination ports, as for ICMP traffic only the contents of Destination Unreachable are useful for identifying blocked connection attempts. Depending on the tools it is possible to capture information on TCP traffic by packet or by connection, ideally the two. Looking at connection level, gather data should be considered at four different levels:
  - Successful connection: three-way handshake completed successfully
  - Failed connection: no response after a connection attempt. Usually repeated two or three times.
  - Blocked connection: client gets its connection refused by either a TCP RST packet, ICMP host Unreachable or unreachable port packet.
  - Aborted connection: the three-way handshake is started but does not finish.

2.2.1 Types of network monitoring

Depending on the environment and the analysis goals, it is necessary to collect information from a network probe, for a specified period and the analysis of such collection depends on the scope. For example, in such network that everything that isn't explicitly denied is permitted, then the system should look for the explicitly denied items. If everything is denied except for the explicitly permitted items, the system should look for the items that aren't denied. In a mixed environment with no explicit permitted or denied items a more complex report should be built that shows the types of activity occurring. The work in this thesis focus in this scenario. The system performs data pattern analysis via ML (chapter 2.4) which performs unusual activity detection.

Since analyzing each packet is unpractical due to computation requirements to do so, a statistical view approach is preferred, such as the discovery of most used ports, server addresses, count of the number of exchanged packets, duplicates/failed/missing, along with others. There are other ways to look at the data. Analysis can include detecting port scans, network misconfiguration (for example private IPs coming from external network) or unauthorized accesses. In the scope of this thesis, network monitoring intends to perform network activity summarization, which ignores the more delicate details and reduces the volume of data to analyze to a manageable amount by including the parameters that describe de network model (as discussed in chapter 2.4.3 as feature engineering).

Network activity summarization has its limitations and difficulties. For example, the connectionless nature of UDP packets presents in a network activity perspective, unclear which one of the two peers exchanging packets is the server. There are some ways to try to infer this information (e.g., who first initialized the exchange), but it is limited to guessing, narrowing the information flow context.

Macroscopic monitoring of network health requires data gathering. The processes can be split into two primary techniques:

- Active: it works by pushing test traffic to analyze how it travels through the network or just by generating traffic for data gathering;

- Passive: in which it analyzes the traffic over a period and reports the results;

Each one has their advantages and disadvantages. Active monitoring tests traffic and bases result in real-time data, can analyze network performance, and measures traffic both inside and outside the network environment. However, it increases the load on network hardware, and for forensics, this means introducing new traffic. On the other hand, passive network monitoring does not inject any traffic, relies on historical and bases the results on longer-term data, providing a more holistic view of the network health. However, it is limited to the datacenter and measures only traffic internally.

The scope of each type of monitoring is also distinct. Active monitoring can ensure and test Quality of Service (QoS), and by generating traffic into the network, it can emulate specific scenarios that generate information that could be observed just by looking at the data. Passive monitoring works better with large data volumes, analyzing the bandwidth (and therefore identifying possible abusers), provides a baseline of the network model due to historical records [53].

2.2.2 Active Monitoring

A few examples of information from monitoring in active monitoring mode include:

- SNMP: Simple Network Management Protocol is one of the most popular protocols for active monitoring. It operates in an agent-manager model, that can be used to "get" and "set" values. It generates traffic in the network and can adjust network parameters.
- ICMP: Internet Control Message Protocol. This protocol can ascertain the reachability and availability of a device and often other information like delay.
- Log data collection: querying logs from servers for offline analysis or scripting to simulate user activity, as they are running and generate information about the performance and reachability, gathering real-time data of the current network status.

In common all these metrics generate some input in the network, as a test or just to invoke information.

2.2.3 Passive Monitoring

As for passive monitoring, the most conventional technique is inserting probes into the network, can be a hardware device, usually know as Terminal Access Point (TAP), or a software solution that monitors a network interface data.

2.2.3.1 Hardware-based passive monitoring

A network TAP is a simple, well-known device, predating the network industry, although the term is now generally adopted as a device that provides access to the actual packets navigating in networks. It provides access to a running network, at the packet level. No other solution provides similar granularity. The simplicity of a TAP device enables a huge array of applications, being the most efficient way to copy the actual traffic running across a system as Figure 2.3 shows [54].

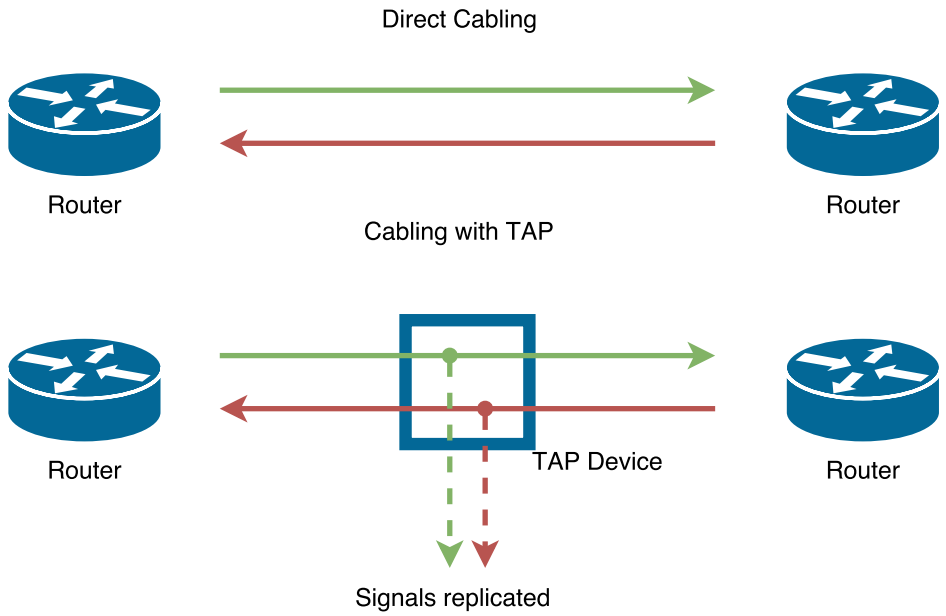

Figure 2.3: Simplicity of a TAP device and how it compares to direct cabling

There are two types of TAP devices: passive and active. Passive requires no power and doesn't interfere or interact with other components of the network. Just copies and relays the signal, being the disadvantage having split ratios and power attenuation.

On the other hand, an active TAP requires a power source and regenerates the signal, eliminating the signal and power attenuation. However, it becomes a point of failure because of possible power outages, leaving the TAP inoperational and consequently leading to connectivity disruption. Signal regeneration, however, can be strategically located where levels are too low[54], [55].

The simplicity of such devices makes them reliable that can run for years, and due to their monitoring, their placement is usually in secure locations. Inserting a TAP can disrupt the connectivity, so TAPs are usually considered in the original network architecture. The idea of using TAPs for security analysis, that works as an Intruction Detection System (IDS) is not new, by analyzing information in a communication line for unwanted intrusions without disrupting the traffic when something is detected[56].

Other solutions for mirroring traffic include *port mirroring*[8] . They both work with the same basics of TAP, supported at a hardware level, with the disadvantage of possible packet loss. As shown in Figure 2.4, it is not possible the aggregation of all traffic. Mirroring ports is impractical due to the limitation of the sum of the bandwidth of multiple ports overload the output port, in addition to requiring configuration and setting additional load in the network devices[55], [57].

[8]Also know as Switch Port Analyzer (SPAN)

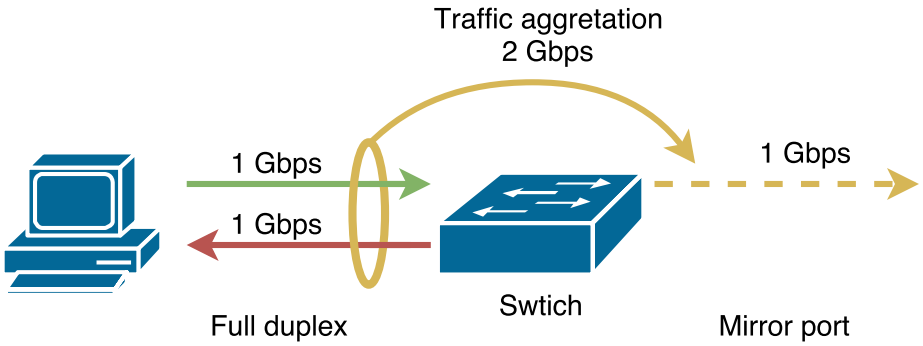

Figure 2.4: Representation of the hardware limits of bandwith aggregation on Switch's port mirroring

2.2.3.2 Software-based passive monitoring

Due to increased server virtualization (Figure 2.5), hardware TAP devices have the limitation of monitoring all the joint traffic of a node, that can aggregate multiple servers under a connection. For a finer, more passive traffic monitoring, a few software solutions have emerged. One example is the famous Wireshark that can TAP wired or wireless network interfaces [58].

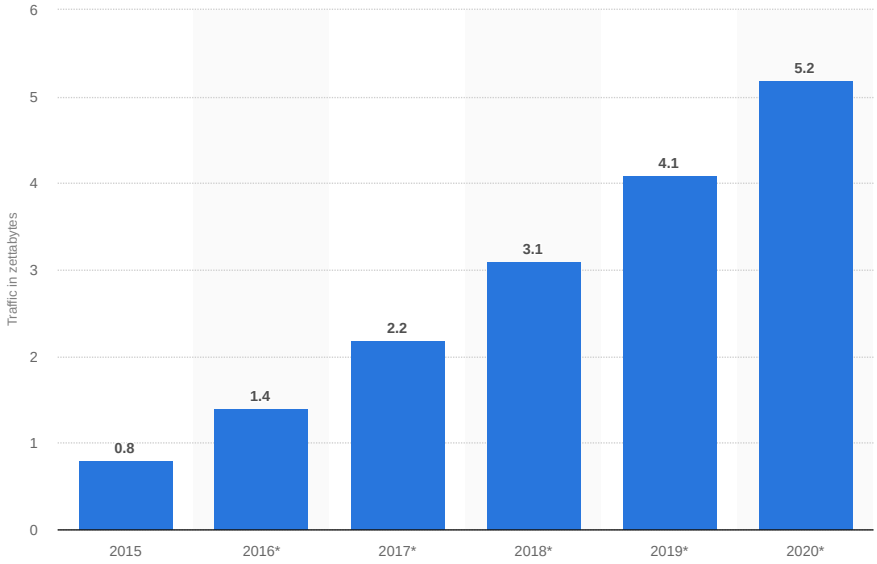

Figure 2.5: Volume of traffic via virtual switches over the years. By the end of 2019 it is expected 4.1 zettabytes worth of virtualized traffic within data centers.

There are a lot of other options and techniques. With this in mind another example of passive software network monitoring, and due to the increased requirement for network monitoring, projects like Data Plane Development Kit (DPDK) became strategic due to the evolution of Software-defined Networking (SDN) and processors extensions like Network function virtualization (NFV). Both NFV and SDN allow the execution of the control of a router on a general processor like the Intel Xeon series. The increased flexibility for data centers to change routing policies on the fly in the software with DPDK allows the creation of virtual switched in solutions using NFV\SDN, with the higher advantage of running in

the user space, avoiding the associated overhead of context switching for rules not in the user space. One of the many advantages is also the increased monitoring capabilities on high-performance networks.[59]–[61].

2.2.3.3 Passive data processing

The data collected from passive monitoring is usually stored in time series events. The goal is to infer network health, performance, availability, and to detect unusual network behavior at a macroscopic level[62]. To archive network information and to detect abnormal behavior, the average values of the network running are used against thresholds to detect when something abnormal starts to happen. Network interference or unusual patterns is detected by frameworks that often combine multiple sources of data, from different protocols.

The deepness of monitoring can also be an issue. Directly monitoring the data exchanged between internet peers[9] raises legal questions[63], leaving the monitoring just to the network activity and health; becoming harder to detect malicious activity directly. With the reveal in 2013 of NSA's PRISM surveillance program that monitored email, video and voice chat (such as Skype), file transfers, and so many others[64], led to the doubling of encrypted Hypertext Transfer Protocol (HTTP) (formally know as HTTP Secure (HTTPS)) traffic[65], [66]. Other projects like *Let's Encrypt* with the sponsorship of internet giants (like Mozilla, Akamai, Cisco, EFF, Google, Facebook, OVH, etc.), are allowing the smooth implementation of HTTPS with dedicated deployment and configuration programs for the most used HTTP servers, resulting in a higher availability of HTTPS websites and services as Figure 2.6 shows [67].

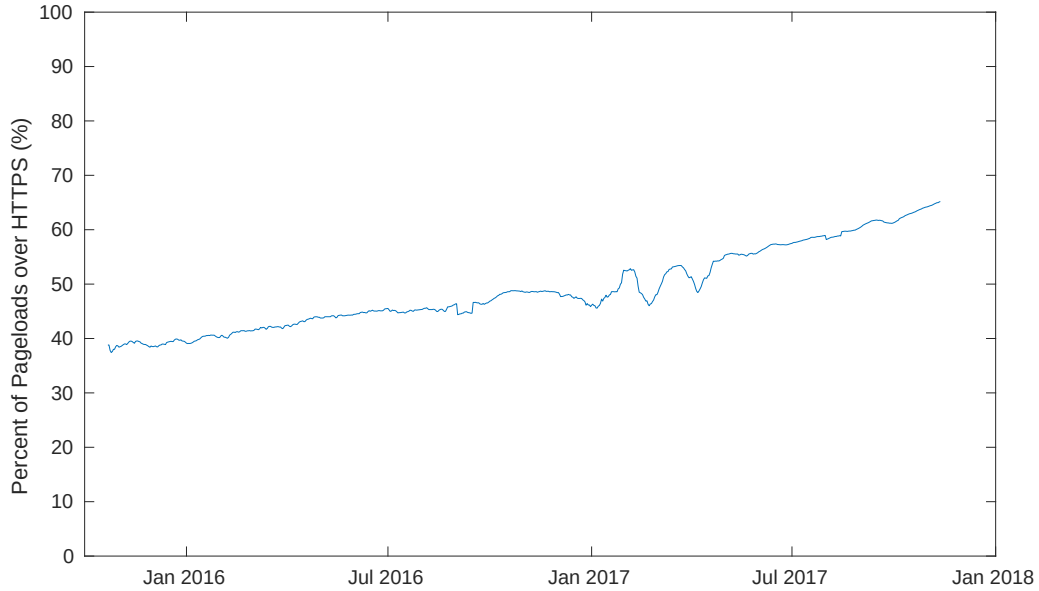

Figure 2.6: The increasing number of HTTPS requests over the years. More than 50% of the requests nowadays are encrypted. 14-day moving average from Mozilla Firefox Telemetry

[9]Meaning: Deep Packet Inspection

Google also pushed HTTPS adoption by ranking higher domains that support it[68]. More than increasing infrastructure costs, webpage loading time and data consumption[66], the rise of HTTPS usage raises new concerns about security and auditing. Information exchanged between two peers cannot be scanned, and although improving privacy, it also masks malicious activity. To overcome unreadable data new techniques for analyzing traffic have surfaced. As the work of B. Miller suggests, applying correlation and ML for pattern analysis, it is possible to reveal nature of the data by examining packet burst count exchanged between the client and the server[69], reducing the privacy provided by HTTPS. Classification and pattern analysis is one of the most used procedures to overcome obfuscation of data. Traffic classification implementation suggested to classify and prioritize webmail traffic[70].

All the suggested methods require high-level network monitoring to then perform an analysis and correlation of patterns (techniques of pattern discovery and correlation discussed in Section 2.4). Gathering information from switches, routers and network devices can be used using the protocols (discussed in Section 2.2.2) before or using a TAP device.

In the context of the work developed in this thesis, monitoring is assumed as a TAP device, without deep packet inspection, limited to statistics from OSI level 3 to 5. The intention of the collected network data is to infer awareness (as discussed in Section 2.5) to provide full NSSA, detecting imperceptible network patterns via ML tools (as discussed in Section 2.4)

2.3 Business Intelligence from Data

The continuous increase of networking and computing is producing a society that feeds in information. However, information must be inferred from its raw format: data. Data mining is one of the ten emerging technologies according to MIT Technology Review that will change the world[71]. Gartner Group claims "Data mining is the process of discovering meaningful new correlations, patterns, and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques"[72]. Therefore, data mining is the extraction of implicit, previously unknown and potentially useful information from data, representing such important field. When scrubbing the data, algorithms can automatically find irregularities or patterns, and when found, can generalize and make predictions on future data. Although, many of the findings can be uninteresting, banal, spurious or accidental coincidences of a particular dataset. Therefore algorithms need to be robust enough to cope with imperfect data, and there must always be a critical analysis of each finding.

This inundation of data across all fields, which costs firms millions to collect and collate, are often stagnant in warehouses and repositories. The main problem is the lack of trained human analysts skilled at translating data into knowledge. As data mining becomes more widespread, companies choose not to adopt these techniques become in danger of losing market share, when market competitors advance into using data mining to gain a market edge. Therefore these fields appeal to managers, CEOs, CIOs, and team leaders, to keep up to date with the latest method for enhancing return on investment, pushing the interest

of data mining as business intelligence. However, the surface of software platforms for data mining is kindled a new kind of danger, due to the easiness of applications to manipulate data, combined with formidable data mining algorithms that work in a black-box making misuse and erroneous conclusions, inappropriate analysis and wrong assumptions that if deployed can lead to expensive failures[73].

Most data analysis is performed under "Business intelligence" that, despite being a generalization, is a synonym of companies that leverage their data and analytics under hidden insights that have a real and measurable impact in their business. This leads to a new concept of doing business, as described by some as "data-driven" management.[74], [75]

> "FedEx and UPS[10] are well known for using data to compete. UPS's data led to the realization that, if its drivers took only right turns (limiting left turns), it would see a large improvement in fuel savings and safety, while reducing wasted time. The results were surprising: UPS saved an astonishing 20.4 million miles off routes in a single year.[75]"

The insights and patterns uncovered in data by using statistical and probabilistic methods, helps organizations to have information that would have been unnoticed. As in the case of UPS and FedEx lead to significant increase in revenues while driving down the costs of running a business. The widespread concept of data mining and the chase of industries moving into this can involve some confusion when delineating the lines between a Statistician, Data Scientist, and Business Intelligence. According to *Revolutions*, a blog from Microsoft dedicated to this area, Table 2.1 tries to describe the differences between a Statistician and Data Scientist terms[76]:

Table 2.1: Informal proposition of the differences between a Statistician and a Data Scientist

	Statistician	**Data Scientist**
Mode	Reactive	Consultative
Works	Solo	In a team
Inputs	Data File, Hupothesis	A business problem
Data	Pre-prepared, clean	Distributed, messy, unstructured
Data Size	Kilobytes	Gigabytes
Focus	Inference (why)	Prediction (what)
Output	Report	Data app/ Data Product
Latency	Weeks	Seconds

On the other hand, as seen in Table 2.2, this is how Business Intelligence will always be a part of an enterprise, using data science as support for decisions to have a competitive advantage [76].

[10] Shipping and delivery companies

Table 2.2: Informal comparison between the topics Business Intelligence and Data Science

	Business Intelligence	**Data Science**
Prespective	Looking backwards	Looking forwards
Actions	Slice and Dice	Interact
Expertsie	Business User	Data Scientist
Questions	What happened?	What will happen? What if?
Output	Table	Answer
Scope	Unlimited	Specific business questions

The embrace of statistics and probability in the form of advanced analytics will continue to be the critical point of a successful business.

2.3.1 Basic Statistical Tools

There are plenty of statistical network models trying to predict network links, probability, and patterns. A survey of statistical network models suggests a division into two big groups [77]:

- **Static Network Models** that focus into a single network snapshot and their essentially static nature, assuming that nodes peering are somewhat stable realized networks. These models try to govern networks change over time and therefore are the primary focus of researchers for last years. The view is at the node-link level: meaning that the observations are each node peering at a particular point in time and how those changes evolve. Models are sustained with graphs analysis, Stochastic Blockmodels and Bayesian inferences.
- **Dynamic Network Models** surfaced with the emergence of online networks and more data available for dynamic analysis, focusing on the dynamics of birth and death of edges and nodes. For example, new nodes can be introduced at any point in time, and old ones can be dropped due to inactivity. This kind of modeling is gaining traction due to its focus on the underlying dynamics of today's networks.

Both of these network models groups are used to infer information from the network itself, such as links or routes changes, attempting to predict the resulting behavior. In the context of this thesis, uncovering patterns will be supported by ML (discussed in 2.4), however, and as described by Table 2.1, as ML heavily depends on statistics, data must be pre-prepared and cleaned before processing. Counters collected from TAPs devices (as described in 2.2) are transformed into values that have some statistical meaning that ultimately are converted into knowledge via ML (as awareness - section 2.5).

With this in mind, the collected network statistical data have to prepared to be consumed by ML algorithms, as an approach to classifying the nature of the underlying network activity.

2.4 Machine-learning

Contrary to common perception, ML is around for decades, present in specialized applications. One of the most common examples are Optical Character Recognition (OCR) [78] (around the 90s) and spam-filters that improved the lives of hundreds of millions of people. It is now spread across many applications including product recommendations according to a user profile and voice searching. It is now expected that ML will transform radiology in the next years by improving detection, image quality, and therapeutic profiles[79].

It is also one of the hot topics in nowadays computer science and at the peak of inflated expectations[80]. The booming of interest in ML started in 2006 with a publication by Geoffrey Hinton et al., that drew attention when demonstrated that a multi-layered neural network could be trained to recognize handwritten digits with a precision of 98% [81].

Despite the elementary concepts and techniques dated to early decades, at the time Geoffrey published his paper it was widely considered impossible to train a deep neural network, an idea that was abandoned by most of the researchers. This publication with mind-blowing results revived the interest of the scientific community, due to the possibility of using Deep Learning for general purpose.

However, what does all of this means? According to Arthur Samuel, 1959:

> "[ML is the] a field of study that gives computers the ability to learn without being explicitly programmed"[82]

Another, more detailed, common definition:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."[83]

ML however is commonly confused with Artifficial Intelligence (AI). ML is a subset of AI, whereas AI is the broader concept of machines being able to perform tasks in a way that can be considered "smart" and ML is the technique used to archive that intelligence[84], [85].

As Figure 2.7 exemplifies, DL is therefore the technique used to perform the analysis, joining algorithm and statistics, calibrating hundreds of algorithms weights that represent the data[84], [86].

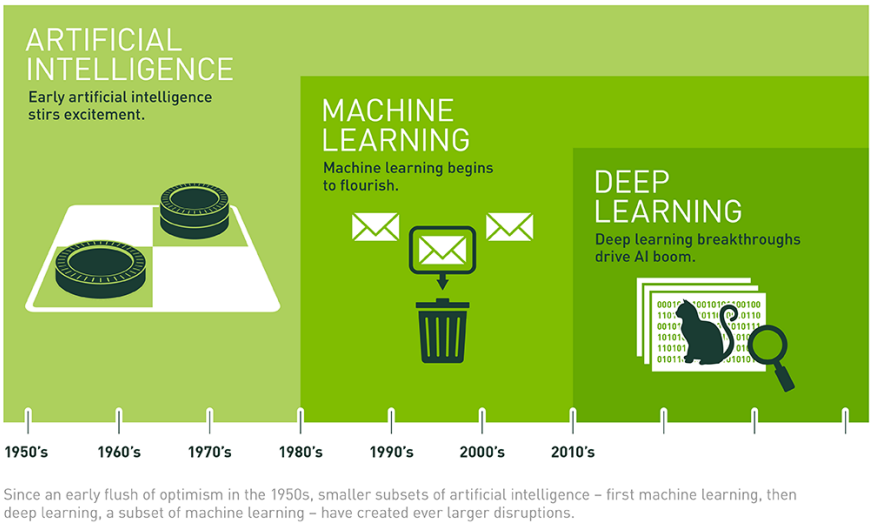

Figure 2.7: Relationship between Artificial Intelligence, Machine Learning and Deep Learning

Consequently, ML can be seen as an algorithm that has tremendous amount of array numbers that we are incapable of setting, and we set those by looking at data.

2.4.1 Typical Scenario

One of the most used examples for explaining the concepts of ML is an email spam filter that can learn how to flag spam. The system is given examples provided by users (e.g., when someone flags an email) and examples of non-spam[11] email. From this data, the examples that the system uses to learn are called *training set* and each training example (every single email) is named *training instance*[87].

In this ML user case illustration, using Mitchel's definition: task T is the flagging spam of new emails, the experience E is the *training data*, and the performance P can be the ratio of correctly classified emails[12][83].

The same detection system written in traditional programming techniques would require[88]:

1. Recognize the most common spam patterns: emails containing words such as "Free" "Amazing", "Incredible",
2. Write a detection algorithm that searches for this typical spam word dictionary
3. Test the program and calibrate the algorithm

Figure 2.8 represents this method of writing algorithms in the classic way [89]:

[11] "In other words "ham" or "good email"
[12] Also known as accuracy. Used in classification tasks.

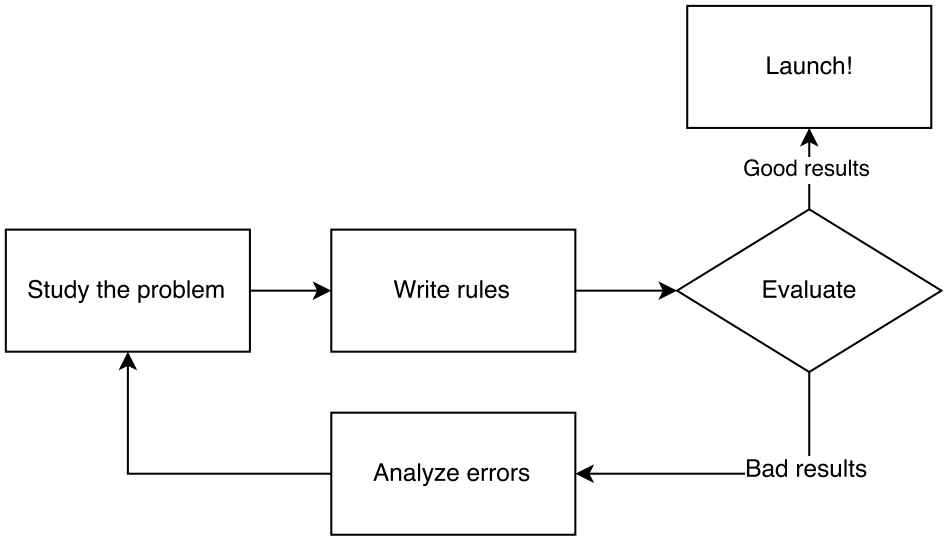

Figure 2.8: Traditional architecture approach for rule-based spam filters

Considering that non-spam emails can contain words of the spam dictionary and vice versa, more complex rules would be required. The system grows in complexity with rules harder to maintain. Also, once spammers notice that the emails are being blocked, they would start to use replacement phases (e.g. "4U" instead of "For you") to avoid being classified as spammers by the system. Every change of the spam patterns would require an update to the detection algorithm. Spammers keep writing and finding new words, and the system would require a never-ending update cycle for its rules[90].

In contrast, and as Figure 2.9 shows, spam filter using ML approach, allows to adapt to change (learning new words substitutions), recognizing which new words and phases patterns are good predictors to flag an email as spam or non-spam. The anti-spam system becomes easier to maintain, more straightforward and probably more efficient[88], [89].

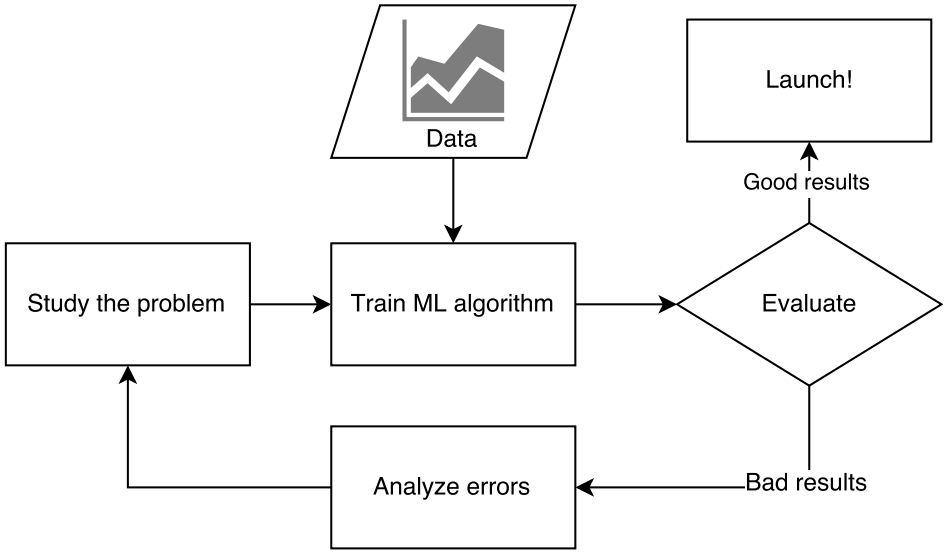

Figure 2.9: Machine-Learning semi-automatic architecture approach for spam filters

The added benefit is that the learning from data and evaluation of system performance can be automated, as Figure 2.10 shows, creating an almost automatic system that only needs monitoring [89].

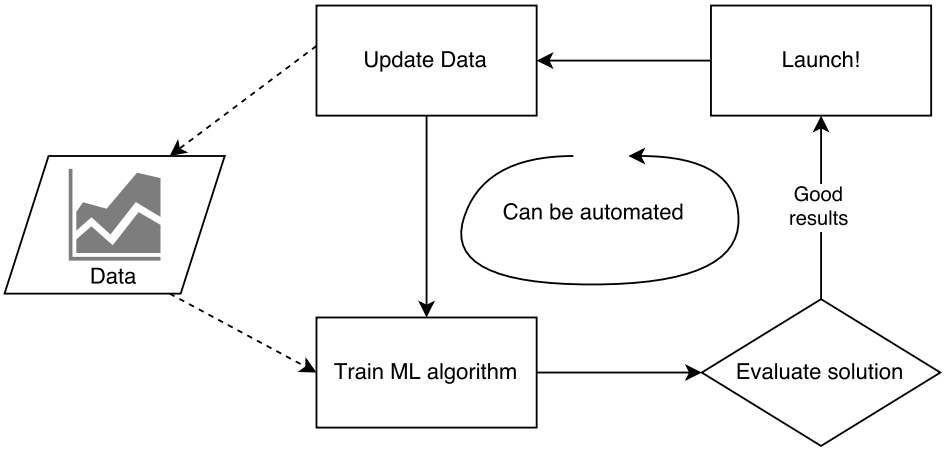

Figure 2.10: Fully automatic Machine-Learning architecture for spam filters

Another strong area where ML shines, are problems that are too complex for traditional approach or where no good algorithm is known. Automatic Speech Recognition (ASR) is one of the research domains taking great advantage of an algorithm learning by itself. A traditional approach to this problem would include, for example, writing rules for special cases for *two* or *too*, speed variations, pitch differences and many others, making it impossible to scale to thousands of words spoken by millions of different speakers. Using ML for ASR outperforms current classical solutions such as *Hidden Markov Model* and *Gaussian Mixture Model* [91].

Last but not least, ML algorithms have been helping to understand problems that initially there was no solution, by studying what they have learned. Learning from ML solutions usually results by applying pattern-recognition algorithms to huge amounts of data and discovering patterns that were not apparent.[13]. Although inspecting the solution is not always straightforward, and can be difficult to decode and understand.

Training ML systems can also generate impressive results. One of the phenomena recurrently happening is when this kind of systems learn and improve current solutions, such as spoken languages, where new and not understandable (by humans) dialects surface when there is no reward sticking with the English language[92]–[94]. This leads to new paradigms how should AI be controlled. Although efforts about understanding AI solutions are emerging[95], should we allow AI to evolve its dialects that evolve speaking to each other in ways humans cannot understand?

[13] Also know as *data mining*

2.4.2 Types of Machine Learning Systems

There are many variations on ML systems. Therefore it is required to classify different techniques in broad categories based on[89]:

- If the training is performed under human supervision (supervised, unsupervised, semi-supervised and Reinforcement Learning)
- If incremental learning is possible (online vs. batch learning)
- If the internal working is based on comparing new data to old data or instead there is the detection of patterns in the training data, and a predictive model is built (instance-based vs. model-based learning)

This criterion is not strict. There are other ways to classify ML solutions and algorithms, for example, instead of classifying how the algorithm works, classify them by the outcome each one provides[96].

2.4.2.1 Supervised/Unsupervised Learning

Organizing ML systems according to how much amount and type of human supervision is required during the training can be split into four major categories: Supervised learning, Unsupervised learning, Semi-supervised learning, and Reinforcement Learning.

One of the main applications of *supervised learning* is *classification*, where a *label* is attached to each *training instance* of the whole *training set*. In the spam filter example each email (*training instance*) has a *label* that classifies each sample in a *class*: spam or non-spam so the system learns how to classify new email[89].

Another example of *supervised learning* usage includes prediction of *target* numeric values, such as the price of a car or a house given a set of *features* (location, age, brand, mileage, etc) called *predictors*. In this case, the training of *supervised learning* system will require data that includes the *predictors* and their *labels* (price), allowing the ML to learn from past examples and then to predict new values for new instances[89].

Some regression algorithms are also used for classification, in which the output value corresponds to a probability of belonging to a class (e.g., 20% chance of a received email belonging to the spam).

Some common supervised algorithms include[89]:

- K-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural Networks[14]

[14]Some *Neural Networks* can be also unsupervised (e.g., *autoencoders* and *Boltzman machines*)

In contrast, in *unsupervised learning*, the training data is *unlabeled* and the system tries to learn by itself, performing *clustering* of the data[87].

For example, given the data of visitors to a blog, when a *clustering* algorithm runs on data, the *unsupervised learning* tries to detect groups of similar visitors. Clustering the visitors of a blog can reveal group patterns. The result of such analysis can, for example, reveal that a percentage of the visitors that read a blog on weekdays at night also like comic books and those who read in the weekends like sci-fi. *Hierarchical clustering* subdivides groups into smaller groups and may help to target each post or topic to each type of users[89]: blog posts on the weekends can be related more to sci-fi and at the weekdays to comic books, optimizing the content to users.

Visualization of data can also benefit from *unsupervised learning*. When a lot of unlabeled and complex data is used, representing it in a 2D or 3D representation may help to understand how organized and structured data is, or even discover new unsuspected patterns detected by *unsupervised algorithms*[89], [97].

As a middle ground, *Semisupervised learning* refers to algorithms deals with partially labeled data. Image recognition is one of the user cases benefiting from automatic classification. Large data sets are used and grouped in the form of clusters. If a cluster is formed, then the data in it must be somehow of the same category, then it can belabeled[98]. One example of this is Google Photos (storage of peoples photographs in the cloud), where the system recognizes different people, group them together but then needs human input to match a face to a name or to calibrate wrongly labeled faces[89].

An entirely different approach is *Reinforcement Learning*. In this situation the system is an *agent* that can perform actions in a closed environment without the need to specify how the task is archived. As input, it receives *rewards* when an action tested by the agent is closer to the desired solution or goal. It works as a trial and error mechanism: the agent scans the environment for behaviors that give the higher rewards[99].

Recently, a successful case of *Reinforcement Learning* was Google's Deepmind project, where the agent is a simple humanoid with proprioception and simplified vision. The agent starts in a world with obstacles, where the only reward how much the agent managed to move forward. Using *Reinforcement Learning*, allowed the humanoid to learn to balance, walk, turn, run, jump and crouch as required to avoid obstacles and to keep improving the reward: more distance travelled[15] [100].

2.4.2.2 Batch and Online Learning

Another criterion to classify ML systems can also be the ability to or not to learn incrementally from a stream of incoming data. In *batch learning* the system keeps its calibration constants while computing each sample in the input. Training data must be used all at once, and the algorithms are unable to learn incrementally from new data: when there are new instances in the *training set* to learn from, all data must be reprocessed. Depending on the volume,

[15]Video from the paper: https://youtu.be/hx_bgoTF7bs

this limitation can be unpractical to maintain: storage, processing time and the inability to quickly adapt to change could be too limiting in some scenarios.

As for *Online Learning* the system trains incrementally, data from the training set can be feed to ML algorithms in single *training instances* or divided into small groups called *mini-batches*. In this case, the weight changes are made at each stage, updating the constants according to the current system state and to the change that each new sample implies[101]. Online learning has advantages when the *training set* is so huge that it cannot fit in the system. The data is divided into smaller sets and served to the ML system in chunks. Another vantage of *Online Learning* is that since the training algorithm does not require all the data to be processed again when new data is available, it works by just processing new chunks, adapting rapidly to the new conditions or environment.

An important parameter of online learning is the *learning rate*, which as the name implies, is how fast the system adapts to change. When the system has a high learning rate it quickly adapts to change, but also quickly forgets the old learned data. This can lead to quick ML performance degradation when wrong data is feed to the system. For example, when in robots if some sensor is malfunctioning and outputting bad values, or in the case of search engines when someone spams search terms. In both cases, faulty data will quickly deteriorate the correct output. In a spam filter, if a high learning rate is used, the system would also quickly forget the old spam models and will only recognize new ones.

On the other hand with low learning rate, the system will have more inertia, meaning that the system will be more tolerant of abnormal data (noise or errors), but also learn more slowly. This means that *online learning* requires constant and more attention to the system performance since it can degrade with time. Typical solutions are constant monitoring and filtering the input data, to avoid some of the performance degradations. Snapshots of system states are also common, since it can be inevitable to restore a previous working system state[89].

2.4.2.3 Instance-Based Versus Model-Based Learning

Additionally to the previous categorizations, most of the ML systems are about making predictions for new instances or values, and the quality of such prediction depends on how well the system can generalize new data. Existing generalization techniques can be divided in two categories: *instance-based* and *model based*.

Using the spam filter example, for an *instance based* learning the system would learn by heart. It would measure the similarity (world count and frequency, number of links, etc.), then new identical emails to *spam emails* would be flagged as spam. On the other hand, a *model-based* system builds a spam model from the *training set* and then performs a regression. The system then has performance measures: *fitness* and *cost* functions, which tells how the new samples fit in the learned models. This approach gives a more measurable analysis of the prediction and the actual result[89].

2.4.3 Main Challenges of Machine Learning

Many things can go wrong when applying ML to a problem. Since it works by processing huge amounts of *training data* there are many aspects regarding the input instead of the ML algorithm itself.

For example, according to SciKit-Learn framework recommendations, some algorithms require a minimum of 100K samples for classifiers and regression, and at least 10K for clustering and dimensionality reduction[102].

Therefore, choosing the right data and how the data is feed to ML systems requires a few precautions.

2.4.3.1 Insufficient Quality of Training Data

Microsoft researchers Michele Banko and Eric Brill demonstrated in a study published in 2001, that the effectiveness on the problem of natural language disambiguation[16] highly depends on the volume of the data. Even simpler algorithms performed identically to most complex for a large corpus [103].

The authors describe the idea that data plays a significant role in complex problem:

> "These results suggest that we may want to reconsider the trade-off between spending time and money on algorithm development versus spending it on corpus development." [103]

The importance of data for complex problems was later strengthened by Peter Norvig in 2009, and revised in 2017 for performance for vision tasks increases logarithmically sorely based on the volume of training data size, showing that performance can be improved just by using better base models[104], [105].

2.4.3.2 Non-representative Training Data

In order to ML algorithms perform good generalizations, the samples of the *training set* must represent the cases to be generalized. This is valid both for instance based or for model-based systems.

A famous case of sampling bias occurred during the preparation for the 1936 presidential election of the United States. An unambiguous and clear result of 2.4 million respondents (biggest in history) giving governor Alfred Landon a projected victory. However, Roosevelt won 46 of the then 48 states. The debate over what went wrong in such a miscalculation escalated.

It turns out that the poll of Literary Digest magazine, surveyed its readers and the readership was skewed in a subgroup that supported Landon. This demonstrated that although the sample size was big, it did not represent the voting class and created a nonrepresentativeness model, unable to provide a correct generalization.

Same assumptions can also be tricky in ML, and the fallacy of thinking that if a data set is big enough, then it is not biased. Despite the volume, the *training set* must be representative

[16] Ability to distinguish "to" from "two" or "too" according to the context

of the data it intends to generalize, and it is always possible that even huge data sets even in the era of the "big data" can be biased[106].

2.4.3.3 Poor-Quality Data

Analogously to biased or non-representative generalized data, if the data-set is full of errors or noise, the guarantee of a system to have good performance is reduced, making harder to detect the possible patterns of data. A few solutions can are used in this cases:

- Clean up the data and discard the noisy or invalid values or samples, to have fewer samples, but with higher value and lower noise
- When there are missing or incomplete features, decide if it is possible to discard it altogether according to its influence on the comprehensive view of the model. Another possibility is to fill the missing data with average values.

Cleaning up the input before running ML algorithms is often worth the hassle when compared compensating in the system algorithms parameters: it is more difficult to calibrate, adjust and test multiple algorithms that just to clean up the data[89].

2.4.3.4 Irrelevant Features

Likewise, ML can only learn from features that indeed can model a problem. If no relevant features are used, or too many are irrelevant or insignificant, then surely *garbage in, garbage out*. The process of choosing the right features is called *feature engineering* that describes the process of[89]:

- Feature Selection: of relevant and useful features
- Feature Extraction: Process data to create relevant features from existing ones
- Create new features that characterize the problem to be generalized

2.4.3.5 Over-fitting and Under-fitting the Training Data

Also, a common phenomenon in ML is how the data of the [training set] is learned adequately according to *fitness functions*[17] but then it fails to generalize when facing new instances. This happens when the prediction comes from an inferred function that perfectly fits the training data, falling into *over fitting*[107], [108].

In the same degree, *under fitting* turn up when the data is too simple, and the problem is more complicated than the model, effecting in inaccurate predictions. Solutions to over and underfitting starts in the *feature engineering* layer, where more parameters, better features or avoiding regularization can help to evade such problems[89].

[17] Fitness functions measure how good the system is generalizing

2.4.4 Testing and Validating

Evaluating the performance of a ML system includes to figure out a balance in how well the system generalizes in the *training set* but also in new data. To archive this, the *training set* is often split in the *training set* and *testing set*.

The error of the system after training, and when experimenting with the *testing set* is named *generalization error*. This is especially useful when comparing two models that behave identically. The *generalization error* can be used both as a discriminatory selector or a calibration for parameters.

Nonetheless, this may be a subtle situation, since validating against different models to the same *testing set* falls into *over fitting* - testing algorithms excessively to the same *training set* that always contains the same data.

To avoid this, the *training set* is split in smaller sets and each model is trained with a different, working as *cross validation*. The *testing set* is all other unused sets, providing a more accurate *generalization error*. Supplementary to avoid *over fitting* also allows to re usage of the *training set*. Finally, a *validation set* is used for a final verification of the generalization performance for a complete new set of instances[89].

Additionally, *No Free Lunch Theorem* is often used as an example. Since each model is a simplified version of observations, meaning that when entirely no assumption of the data is made, there's no reason to prefer a model over any other. For some, it may be a linear regression and for another a neural network. There is no guarantee *à priori* that any model is preferred over another and to work better and the only way to know which one works for each case is to test them all[18][109].

2.5 From awareness to cyber security

As discussed in Section 2.3, collection of statistical information from network activity *per si* is not useful. It most the converted in information first (in the case of this work, by detecting patterns using tools discussed in Section 2.4). However, detecting patterns only provides an output - a result of some data processing. To convert the results of ML recognition is necessary to have the concept of *context*, which combined with the results of data processing, can be latter converted into *information* that ultimately results in *awareness*.

The study objective of this thesis is to infer knowledge about the possible information leakage or malicious network activity; the two combined defines the *awareness of network state* in a certain network context - or view. Both concepts, *context* and *awareness*, are broad and range from many different study fields. Webster's dictionary defines *context* as "the interrelated conditions in which something exists or occurs". The formal definition may seem simple and intuitive, but when this concept is projected in ubiquitous computing, "context" is any information about the circumstances, conditions or surroundings that are relevant to the interaction of the Human with the machine. When Humans communicate with each other,

[18] However, a few assumptions can be made to narrow down to not all the algorithms

a wide variety of contextual information is used (e.g., location, temperature, time, current political situation, and so forth) but to enable computers to make use of such contextual information may be a challenging task. Work has been developed in order to provide physical (e.g. location, time), environmental (e.g. weather, light, temperature), informational (e.g stock price, game scores), personal (e.g. health, mood) social (e.g. group activity, relationships) and so on contexts to computer applications, providing a more ubiquitous interactions[110].

Complementary to *context*, the concept of *awareness* is also well established in human studies. Endsley describes as

> "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future"[111]

. It becomes an important component for supporting decisions, and the evolution of the concept itself led to awareness models with a goal-oriented perspective.

As such, *Situation Awareness (SA)* modeling was latter introduced by Schilit and Theimer with his work of developing software that adapts to locations of usage, nearby people and surrounding objects, including changes over time of these factors [112]. As the technology evolved and the interest in mobile and wearable devices, the *context aware* modeling research become an area of increasing interest, due to the fact of the possibility of developing adaptable user profiles and preferences to each user. This adaptation allows a more efficient interaction of users with its devices, actively customizing and changing according to the current situation of its user. Note that the same information may have different meanings for different people or situations. Therefore the system must be aware of the *context* information provided by its user[113], increasing the complexity needed for such *awareness*.

Introduction and rise of SA become a hot topic in military context for security surveillance of facilities with a low probability of intruders. Mainly for situations where although the chances of an attack being low, the consequence of such unlikely event could be great. In this scenarios, Human monitoring becomes a challenge due to the little or no activity leading to boredom and also unadaptation to a response, due to a continuous and too hardened expectation of not having events or intruders. SA research, therefore, lead to the combination of ML with SA, introducing a new ways blend information and decision support, providing mechanisms that continuously sense and infer the state of the environment, providing aid for better human response and decisions when some event happens [114].

Although not always in coherence[115], the combination of the notions and concepts how *awareness* applies in computation is an exploding topic. There are a few suggestions from different authors, as discussed in the following sections, how these views adapt to the cyberspace, and its applicability in cybersecurity on how the notion of awareness and *context* can be extracted with the help of ML [44].

A simple representation of currently levels of *awareness* can be seen in Figure 2.11. SA, is a subfield only possible due to the wider concept of *awareness*. In computing, Cyber Situational Awareness (CSA), is the baseline of Effective Cyber Situational Awareness (ECSA) and the two combined provide NSSA, a specific subfield of awareness applied to computer networking.

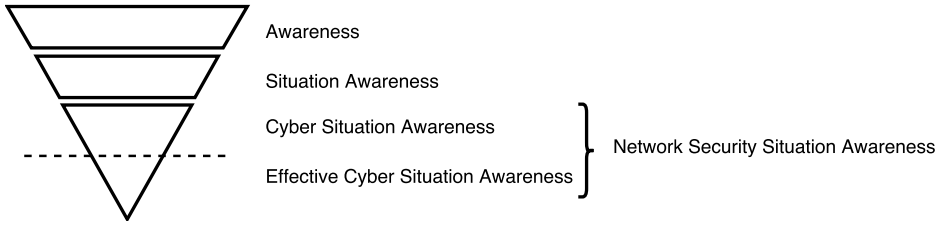

Figure 2.11: Different layers of awareness

Each layer adds more decision automation and response, relieving the human factor in the decision making. Despite the fact the immature nature of such definitions and awareness layering, in the case of NSSA is already being implemented according to the availability of tools. For example in computer science, despite not being fully aware or automated, simple statistical network analysis can provide hints of possible network intrusion and work as a IDS.

2.5.1 Cyber SA: Situational Awareness in Computing

SA is now adjusting to the digital world of cyber defense and is a broad term, capturing many aspects of *awareness* in a given situation. It is about being aware of what is happening and the relative importance of what is observed. The inclusion of actions, future states, and their likelihood, or even awareness of others'awareness. Like so, the concept of SA can be looked from different perspectives [116]. From a technical viewpoint, SA is the process of compiling, processing and fusing data. This data processing includes the ability of process fragments and a rational estimate of data quality, making possible to technically make relations of different pieces of inputs with each other[115]. On the other hand, cognitive view of SA concerns to the ability to comprehend implication and draw conclusions to perform informed decisions. Therefore, measuring the extension of *awareness* in a given situation and how it evolves as the time passes becomes an interesting research field. Endsley definition can be used as the base for such[115]:

> "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future"[117]

As suggested by Endsley, *preception*, *comprehension* and *projection* can be taken into consideration of incresingly awareness levels, ranging from[117]:

1. Basic perception of important data
2. Interpretation and combination of data into knowledge
3. Ability to predict future events and their implications

Overall, in Figure 2.12, the model suggested by Endsley, SA includes three levels and the work of this thesis is focused in *Individual Factors* and towards Level 1 and 2, with no projection of future status. The automated decision by ML and their respective *performance of actions* are not in the scope of this dissertation [117].

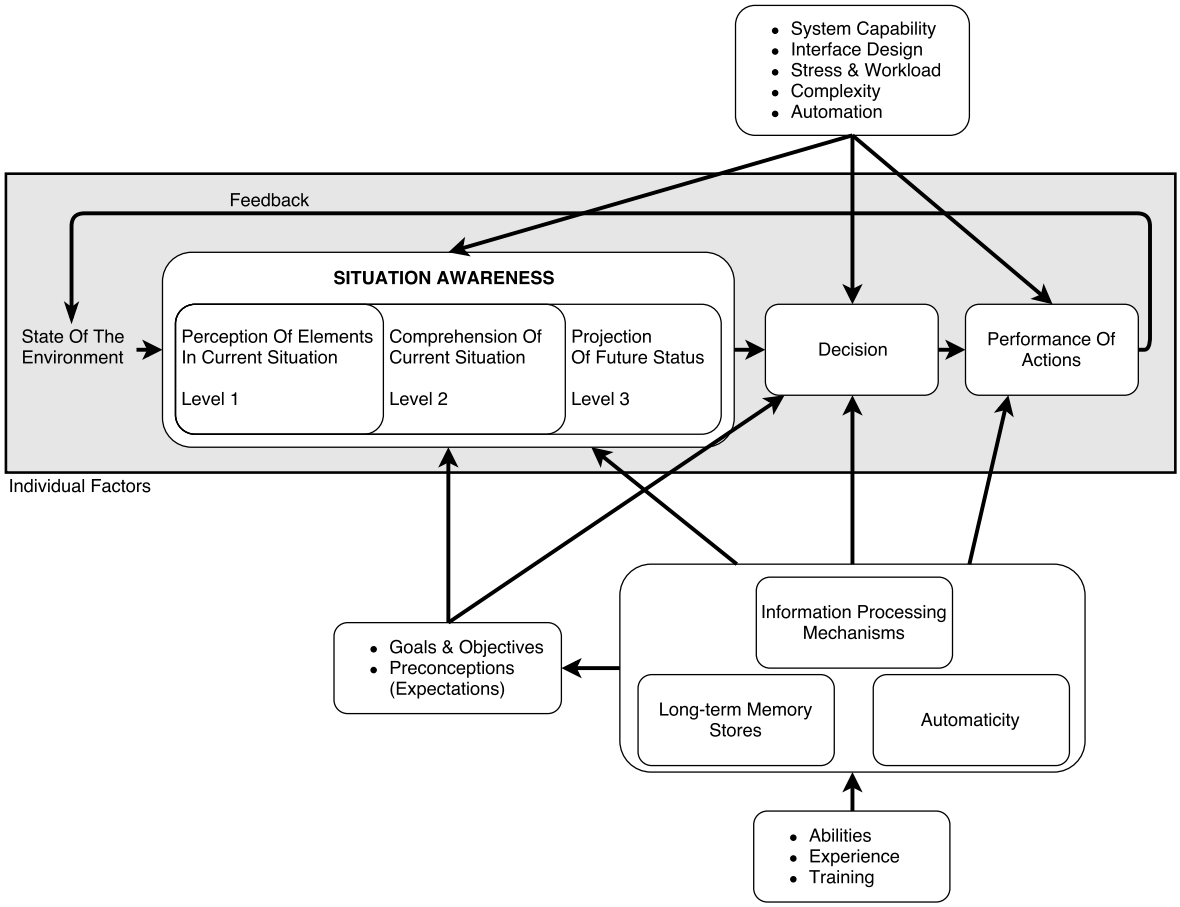

Figure 2.12: Conceptual model of situation awareness in dynamic decision making

As the concept is maturing, SA adaptation to computing field now consists of a more well-defined topic with propositions of the capabilities of what is a CSA system. Suggestions that define the ability of such system are converging into the follow requirements[118]:

1. Be aware of the current situation
2. Be aware of the impact of the attack
3. Be aware of how situations evolve
4. Be aware of actor (adversary) behavior
5. Be aware of why and how the current situation is caused
6. Be aware of the quality (and trustworthiness) of the collected situation
7. Assess plausible futures of the current situation.

The whole CSA is also described as a three-phase process[118]:

(a) Situation recognition (aspect 1 and 6)
(b) Situation comprehension (aspect 2, 4 and 5)
(c) Situation projection (aspect 2, 4 and 5)

SA involves the technical and cognitive challenges needed to estimate the state of the world, according to the processing of the input data. The development of SA is the result of data fusion, i.e., the process of combining multiple sources of data to define estimates of the

current state and possible predictions of the future [119]. Therefore, data fusion is the process of making large amounts of data into comprehensible information, and the development of systems able to provide decision support for the decision-makers. For this reason, CSA is considered a subset of SA, being the situational awareness in the cyber environment. Such situational awareness is therefore built on data (but not limited to) from IT sensors, fed into a data fusion process and interpreted by the decision-maker. Despite CSA is related to the cyber context regarding cyber issues it cannot be treated in isolation, but as a part of information gathered from SA to obtain a more comprehensive view of the situation as an additional insight rather than a disjoint situation. On that account, CSA includes any suspicious/interesting activity occurring in cyberspace context (e.g., network activity). However, even with all these aspects of awareness, human intervention and decision are required as the ultimate component of the system gaining awareness. In practice and by definition, SA includes the hardware required for the software to work but also the human mental process for making advanced or final decisions. Lastly, awareness itself can be layered, since the human capacity of processing raw data is limited (or practically impossible due to the huge volumes of nowadays data), only archived by ML systems that converted data to readable information [118].

Back in the 2010, conclusions and observations about the state-of-the-art of the developing SA technologies included[118]:

- The gap between hardware and software. CSA relies upon the physical sensors signal processing or log files processors, antivirus alerts, and so on. There's this kind of compatibility layer between the *perception signals* and SA systems.
- SA is highly dependable of vulnerability analysis such as attack graphs, instruction detection systems and alert correlations. As the layer of abstraction is reduced the human decision factor also increases, making a labor-intensive, time-consuming and error-prone task to process the results of an underlying SA system. This, however, is progressively evolving with the work of mechanisms to avoid decision degradation and better decisions without the human factor as the handling of uncertainty is improving [120].
- In line with the last topic, the obvious gap between the human analysts' mental model and the capability of CSA frameworks.

A few points about the future of SA systems and its goals include[118]:

- The ultimate dream of developing such algorithm that provides a machine self-situation-awareness so that it could protect themselves
- The possibility of such algorithm to provide the recognition and evolving situations, generating reasoning, response plans and automatically taking action.

The collection of this goals and future perspectives are a projection of how things tend to evolve and how ideally a full comprehensive SA system should perform. Currently, the applicability and performance of SA must fit the capabilities currently available tools, many of them in development and under testing [121].

2.5.2 Applying Situation Awareness to Networks

Generalization of SA to the security context is beginning to emerge from more specific usage and applied to more contained contexts - smaller networks or particular scenarios, instead of a global view. Security analysts operators are one of the careers that observe online operations of corporate networks for external, internal, random or organized cyber-attacks. The usage of SA systems currently employ instance-based learning in a network for events recognition (e.g., an execution of a file) where the attributes events are compared with past instances. The models that power these SA systems can predict the sequence of events that may lead or correspond to attacks, providing SA to the analyst evaluation of accuracy and correctness for decision actions [122].

The definition of SA models exclusively for networks is also being improved. Latest proposed definition dates from 2014, ECSA focus in a holistic view of SA that includes three main phases[123]:

- Network awareness that includes analysis and enumerations of defense capabilities
- Threat or attack awareness that establishes the current situation picture of possible attack vectors
- Operational awareness that establishes the current SA status of the operation: how the degraded network operations affect the mission of a network.

This latest model, ECSA, is specially crafted for networks and more comprehensive than regular generalized CSA, giving better decision making and performance of the actions applied in a network [123].Without the regard of a model, maintaining CSA in dynamic and complex systems is difficult. Network complexity, inherent dynamic changes (hardware or software) and endless ways a user can behave are always factors for network uncertainty. Network monitoring tools generate a considerable amount of data, making its analysis more unpractical as the network dimension escalates. In this situations, there are proposals that instead of providing full network awareness, CSA should just prioritize alerts to experts [124].

In practical terms, network-oriented awareness has been tested in power grids, that due to its nature represent more contained and controlled view of how to combine ML for abnormality or misuse detection. The work evolved studies of the appropriate ML algorithm and the corresponding *feature engineering* selection: log files where used combined with network sensors. The result showed that awareness via ML produced exceptional results and high attack detection rates[125].

However, such concepts for network awareness are not linear and can't be generalized. For example, monitoring huge networks requires different approaches, as demonstrated by a study of a telescope of 16.5 million darknet IP address. Processing such large number of IPs and its correspondent raw data is not practical. In these cases, just the network artifacts is a practical way to still provide network SA in such scenario[126].

2.5.3 Network Security Situation Awareness

Finally, the combination of *Security*, *Networks* and *Situation Awareness* ultimately leads to NSSA. Models that provide a CSA are recently being used as IDS, preventing network systems from having confidentiality, integrity, and availability compromised. However, the development of IDS based on real network traffic generates a high number of alarms per day, being 99% of them false positives[127]. Once more, to archive NSSA, the combination with ML was unavoidable, because features collected from heterogeneous sources can have its noise reduced plus the possibility of fusing and reducing data with a Neural Network (NN), it is possible to develop a system that outputs lower false positives and can provide a threat level [127].

The research in this area has not matured yet, several prepositions and definitions emerged, being one of the most cited definitions of NSSA [128]:

Network security situation awareness is the cognitive process of the security status of the network system, including the original data measured from the system step by step to the fusion extraction process as well as the realization of the background condition and activity of the semantic system, so as to identify the existence of intention all kinds of network activities and the abnormal activities. Then it can obtain the understanding on the normal behavior of the network security situation trend as well as the influence on the normal operation of a network system."

A survey summarizes the funcition of a NSSA system split into three aspects [128]:

- Network Situation Awareness
- Network Threat Evaluation Deals with the destructive capacity of the malicious attack, and the estimation of the global network threat, based on network situation assessment (focus on effects of events)
- Network Situation Assessment Includes the extraction of situation elements, current situation analysis and situation prediction (focus on the emergence of events)

2.5.3.1 Structure of an NSSA model

There have been a variety of proposals for analytical models of NSSA. Among them, JDL. is a widely accepted model that is based on the concept of data fusion. This framework includes multi-source heterogeneous data collection, data preprocessing, event correlation, target recognition, situation assessment, threat assessment, response and early warning, visualization display of situation, process optimization control, and management. A macroscopic view of a combination of related research, as well the JDL model and SA proposed by Endsley, a structure model of NSSA can be seen as shown in Figure 2.13 as a concept model [128]:

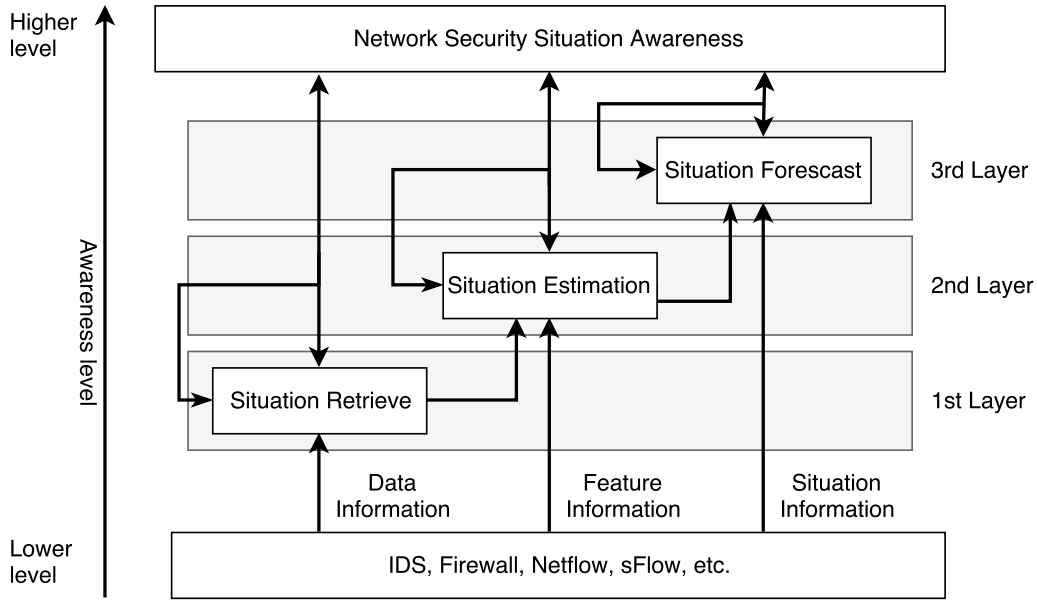

Figure 2.13: Conceptual model for Network Security Situation Awareness

The first layer mainly extracts the *situation index*, metrics that can model the current network behavior, by monitoring and analyze the network for abnormal or unusual network patterns (e.g., by monitoring network flow). The second layer analyses the network state, e.g., if it is under attack or if in a typical situation. This layer is responsible for the according to model to generate the corresponding status of network security. Lastly, the third layer and according to the information of the second layer, realizes and predicts the network security situation on the next stage, which helps the network management staff to act accordingly to this high-level information, providing the basis for a reasonable decision.

Execution of a NSSA model represents a continuous process, where the evaluation of the current situation is the core of the model. Having information what the current evaluation of the network security situation is providing intel to analyze and deal with the intrusion threats and details of the intrusion behavior, exposing a contrast of the layer what is known default behavior with the new malicious activity. For this reason, critical technologies in NSSA include network security feature extraction, security situation assessment and situation prediction, represented by the three layers of Figure 2.13.

- Extraction of Situation Features

  Due to the fact of network security situation is in constant change and constantly exposed to virus, bugs, and changes, using only one classifier to archive the extraction of the security situation may be just too complex and variable in a large-scale complex network environment, making challenging to guarantee or even to archive the detection effect. With this in mind, it has been proposed to extract security situation using a multi-classifier fusion, e.g., a combination of neural networks with support vector machines, fuzzy clustering and so on. The combined results for the same events results in better reasoning on the likelihood of an abnormal event.

- Assessment of Network Security Situation

  After extraction and preprocessing of network security events, the first step is the evaluation of the overall network security, which is a measure of the entire network system. This layer is in charge of evaluating the protection of the network to a threat. Therefore this assessment of network security evaluates if the network is safe or not and may include prediction of the value of the current network security state.
- Prediction of Network Security Situation

  For active protection in a large-scale network, the prediction is the key link of the completeness of the NSSA model, and the last step of the overall awareness. It can be seen as the continuation of the *Assessment of Network Security Situation*, Based on this, only the accurate evaluation of the current security situation can get a reliable prediction model that express the dynamic changes, which predicts the future network conditions, therefore helping decision-makers to perform decisions.

NSSA is an emerging technology that can extract the behavior of a threat, shifting to prediction and assessment the future trend of network security. All the different approaches to develop an IDS based on this concepts, that takes into account the past and current events to predict the future, can be seen as a precision measurement tool that helps decision-makers to carry out the actions needed to restore the security of a network. However, in a large-scale complex network environment, the biggest support of a hypothesis is visualization, both of current situation and future implications. Thus, work is being developed in order data visualization, giving the ultimate network comprehension system to analysts[129].

CHAPTER 3

Network entity classification

In this chapter is discussed the methodologies used to infer subtle network patterns of network entities. The whole process works as an assembly line, where the raw data of the network is the input, and the final result is the information about what entity corresponds each network observation and if it is acting on its usual behavior. The first section begins with details how raw data is gathered, followed by a description of the data structure used to store the activity model, how to overview and analyze the data set and finally the process of classifying and evaluate. Overall, this is the pipeline from data preparation until the usage of ML as a tool to detect patterns.

3.1 Network Data Collection

As discussed in Chapter 1 the main idea of inferring entity network patterns begins with generalized network activity observations. Watching the network can be performed at many layers and in way too many ways, therefore for the sake of simplicity and since it is outside the scope of this dissertation, the acquisition of entity network activity is considered a sniffer probe as close as possible to the entity with the following assumptions:

- Correlation between the IP address and the entity

  Each entity, human or not, is recognized by the IP addresses attributed to it. Doesn't necessarily mean that it is a single service or person, but defines the pattern to recognize.
- Probe can count network activity

  Deployed sniffer probes can read OSI level 3 (Network), 4 (Transport) and 5 (Session) to collect the values of each layer for every packet.
- Collected counters are time-based

  The read data from packets are pinpointed in time at database insertion or upon packet capture.

Concerning the requirements, the sniffer collects packet data from OSI levels 3 to 5, which depending on the architecture, can be stored in a document-oriented database or a capture

format as the PCAP format specification [130], [131]. Storing network activity supports queries and iterations over the captured data before finding an optimal network model, allowing extending or modification it in the future as different ways of viewing and processing the data emerges along the analysis of the first models. However, in a final version, the sniffer and its storage can be optimized by discarding unneeded packet data and keeping only the metadata of the chosen model. Figure 3.1 shows that information collected from the sniffer increases accordingly to the protocol of the captured packet. The suggested structure in Section 3.2.1 allows simplistic feature processing and storage, and thus fast prototyping, offering a compromise between oversimplification and storing the full of captured packet data.

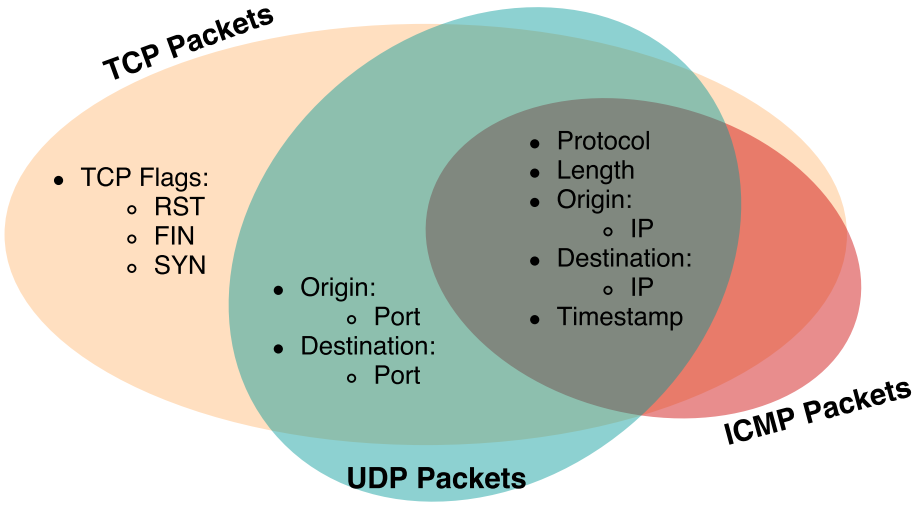

Figure 3.1: Representation of collected metadata from the network packets, and how they stack according to the protocol of the capture

Each field of the capture has the following information:

- Common to TCP, UDP and ICMP packets:
  - **Protocol**: TCP, UDP or ICMP
  - **Length**: size of the packet
  - Origin:
    - **IP**: origin IP
  - Destination:
    - **IP**: destination IP
  - **Timestamp**: UNIX epoch time of the packet
- Common to TCP and UDP:
  - Origin
    - **Port**: origin port

  - Destination:
    - **Port**: source port
- Used for TCP packets only:
  - TCP Flags
    - RST: If reset flag is present
    - FIN: If fin flag is present
    - SYN: If synchronize flag is present

As it is intended to test the hypothesis of inferring patterns from a network and not truly optimize or study the capture process, remained to be done the capture performance or due to heterogeneous nature of network implementations, strategies where to insert the probes. Section 2.2 details a generic process of the network capture and its objectives, which may influence the strategy of the probes implementation. The sniffer could be smarter (e.g., process the statistical data directly and insert later in the database the processed data, instead of centralizing the data treatment), and thus discarding the already ephemeral nature of network activity instead of storing it. In the conceptual implementation in this work, probes perform the minimum possible processing tasks before storing the captured data. The only non-network data collected and inserted by the probes deployed in the proof of concept of Chapter 4 is the current time and date of the packet. Extracting the time upon the packet collection allows avoiding the jitter communication with the database and keeps the database abstracted to a simple storage system which may be any of the suggested types. Not doing so would require the database/storing to have some way of keeping track of packet date. Avoiding jitter also allows fewer errors when inferring periodic network activity (as discussed in Section 3.2).

In light of this, at an enterprise network level, installation of a sniffer on every endpoint is much probably unacceptable and therefore adding processing capabilities to the sniffer probes is for certain unavoidable. The sniffer must recognize the entities without deploying it directly on the endpoint. Also, storing every network packet data even if just OSI layers 3 to 5 generates too much data volume, unreasonable in a real-world scenario. The process of obtaining the relevant information at the sniffer-level in an ideal scenario and as closely as possible to the probe. This way, the sniffed data collected by the probe can and should be discarded right after, removing the need for storing all of the packets, minimizing the overhead of data stream up to the decision points. Next section describes a possible structure to do so, as each probe computes the corresponding sliding window and sends only the result, minimizing the cost of sending statistics over the network itself.

Another thing to take into consideration is labeling the data according to its tasks and not just by a mere entity label. For example, labeling if the entity is a server, a desktop computer operated by a human or even if it is an IoT device.

However, and as in the case-study of this dissertation and its implementation, keeping all data allows to study and test different model and feature combinations by adding and removing *Counters* attributes, different time window lengths, and so forth. At this level,

packets are processed independently, and the only label applied is what packet belongs to each probe.

3.2 Feature engineering

3.2.1 Converting packets into features

The data collected by the probes do not represent any kind information and, at this point, it is only network packet metadata. It is necessary to iterate over the collection to have a model that describes a pattern or behavior. The dataset requires modeling that links features to a label, and in this case, defines the relationship between different network behaviors that ultimately distinguish the entities. Some models could be described with a few rules, but taking advantage of machine learning algorithms is exploiting the power of discovering the subtle patterns which make each label different from another. In other words: a long and detailed analysis of the dataset could perhaps determine the relationship of different behaviors which if translated into many condition statements could correctly classify the different entities. This way, traditional programming techniques could also possible solve this issue. On more complicated dataset discovering this relation becomes more difficult and maybe impossible, while good machine learning algorithms determine the relations for us. If it is fed enough representative examples into the right machine learning model and if these examples can represent such relationships data relations are automatically defined. However, these links are only discovered on data where a pattern exists. Determining what to feed into the algorithm is known as "feature engineering" and many times this judgment is a set of experience, knowing the problem and comprehending the data's source. In this case, trying to represent what determines different network expression and what can be extracted from the captured packets. There can be many different views on how to create the features over the captured data, and a few of them are more obvious than others, but ultimately will define the relationships on the network.

One prominent approach is to process the packet data to how it links with time. There are recurring events that happen across the entities at different rates, frequency, and scales. As in an illustration, a periodic "day" scale is a server running 24/7, which is different from an employee that works only during the day. A company file server will likely have more volume during the day rather than night, or backups that occur during the night. A minute-level scale is, for instance, reading news or blogs, as it which generate bursts of traffic for a few seconds every few minutes. The periodicity of packet rates reveals underlying services. However, due to the nature of this events and may vary and shift over time and in their frequency, they are more like pseudo-periodic and not strict. This variation should also be taking into account that its representation is a balance between detecting such events and tolerating their variations.

These examples are dependent on time, and therefore packets should be expressed/modeled that way, and much possible enrich the pattern recognition. With this in mind, Figure 3.2 shows a possible approach to abstract the nature of the network packets into a structure called *Timebar*. It represents the exchanged packets in a day, sliced into slots that aggregate the counting of the statistical data within a slice, that will be used to compute features. These features will afterward generate network observations.

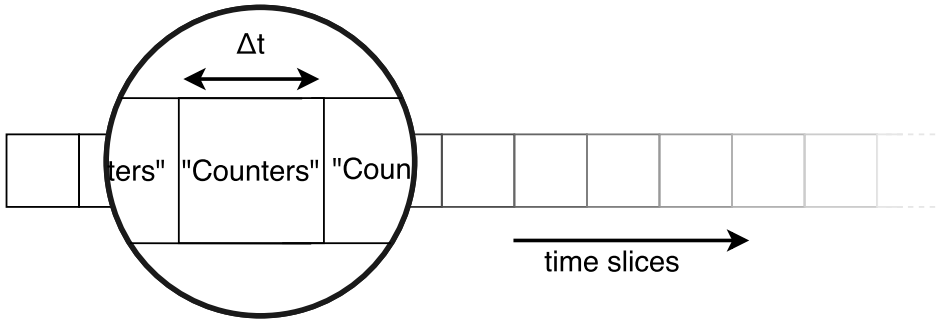

Figure 3.2: Representation of the *Timebar* slices that hold the sum of metadata collected from the probes. Each time slice aggregates the sum of packets metadata during the ΔT time frame

Each *Timebar* works as an array of variable length according to the number of slices. As an example, a *Timebar* with a single slot would aggregate and sum the data for a whole day, allowing to detect and infer patterns occurring per day basis (e.g., the pattern of weekdays, weekends, holidays, and so forth). A *Timebar* with 24 slots slices the day in hours, meaning that hourly patterns such as employee lunchtimes or typical work hours could be detected. The length *Timebar* variation allows different resolutions and therefore detecting the patterns that occur at each scale. The lower the ΔT less data is aggregated and high-frequency events are detected, such as machine-to-machine communications and protocols. Higher values of ΔT sums up more data and allows detecting more human-scale events.

Each slot contains attributes, delineated as *Counters* structure, which is the sum of the packet events occurred within a time slice, given by the captured by the probes. The *Counter* could be a model composition by following attributes:

- Statistical packet counts

  Counting of packets and their attributes decorates the model with the patterns of each entity. Different patterns are at last represented by its statistical expression of the exchanged packets.
- Data stream aggregations

  Data streams also represent entities communications nature, the volume, duration, peers and so forth represent different actions and purposes.
- Network protocol activity

  The number of ARP and UDP packets, the protocols used, etc represent its position on the network and its default pattern.
- Many others

  The list is not closed. The modeling depends on what is trying to be modeled. In the case of behavior, there is a trend to define the entity posture on the network, a "personality" that differs from each one to another.

The sum of packet values in each *Timebar* slice require computation, as it is the sum of the packets within a time frame and directly processed from the storage format. Figure3.3 shows a slice *Counter* low-level structure attributes implementation.

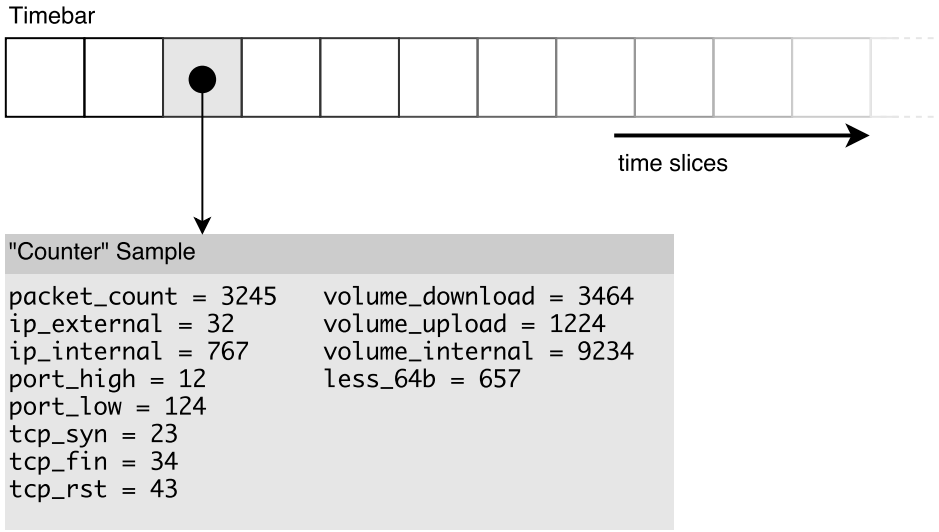

Figure 3.3: Low-level representation of a *Counter* structure in a *Timebar* slice

Attributes must also be flattened once processed; this means that each *Timebar* slice unrolls to an array with the *Counters*'s attributes length. As an example, a Timebar of 86400 slices (slicing every second) with 12 *Counter* attributes produces a bidimensional array of (86400, 12). A whole month of data can generate a tridimensional array of (days, Timebar slices, Counters), in this case: (31, 86400, 12) with a total of 32 million values representing a month of network activity for one entity, subsequently processed into entity observation features.

3.2.2 Generating network observations

The generation of *Counter*'s attributes is the first layer of computation performed over the network activity but doesn't necessarily model entities activity, it merely aggregates and sums of data flow and volume. Over the *Counter* it is necessary to transform data into something that represent activity in such way that pattern changes are perceived. Computing these values is the last preparation step of the sniffed data before using the ML pipeline to detect patterns and their differences. Therefore, the better is the modeling of the *Counter* attributes the easier will be for ML algorithms to output relevant results.

Two things must be taken into consideration before processing these data structures. The first place is how to iterate over over each entity *Timebar*'s *Counters* attributes. The iteration of the *Timebar* slices is directly related to the final system response time and must be balanced to not under or over split the slices. Secondly is what to extract from *Counters* that represent activity in such way, transforming it in insightful observations. As Table 3.1 shows, one of the initial approaches is to divide a day into blocks and compute features over those blocks.

Table 3.1: Relationship between *Timebar* slices granularity and the number of observations generated

Number of splits	Block time length	Timebar slots consumed
1	24 hours	86400
4	6 hours	21600
24	1 hour	3600
96	15 minutes	900
288	5 minutes	300
1440	1 minute	60

The splitting strategy is directly linked to response time and the number of observations produced. For example, if no splits are made over a *Timebar* day, means that the data is snapshotted as a whole day and will produce just one observation. It also means that the response time is only possible after a day completes, as it is only possible to compare if a day is normal or abnormal when the day ends. However, splitting too much dissolves the expression of different behaviors during the day: all data looks more and less the same when is oversplit. Finding the right balance of the number of observations must also take into account the ML algorithm being used, with some of them requiring the absolute minimum of 10k observations before being reliable [132]. Hard splitting into blocks also adds the problem of some entity shifted it is activity over time, e.g., if an employee skipped the work during the morning and performed the morning tasks in the afternoon. This activity change is too deeply connected with a day block and would be detected as an anomaly when shifts from one block to another. Splitting into different block sizes have different advantages according to the purpose of the system. For example, viewing the day as a whole may be a useful metric for other IDS systems that use this information to infer the Network Situation Awareness (NSA), while splitting day/nigh blocks allow the separation of automated tasks and employee activity. With this in mind, another strategy is instead of splitting a *Timebar* in blocks and compute over them, entity activity is better abstracted and expressed with a sliding window moving over the *Timebar Counters*.

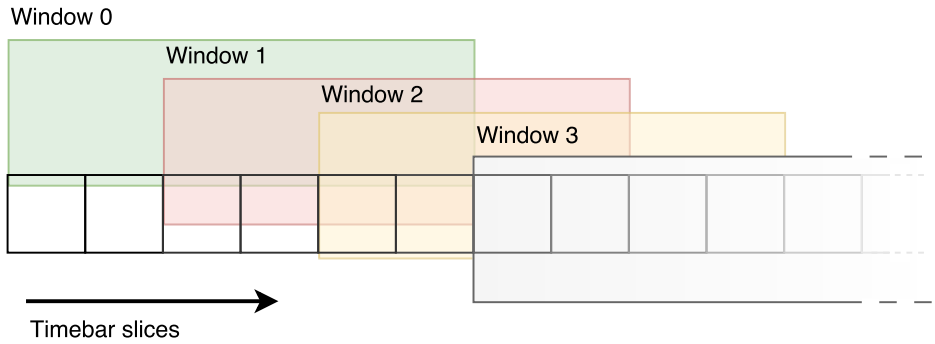

Figure 3.4: Representation of a sliding window overlaping and iterating over a *Timebar*. Each window reads a range of *Counter* attributes in range and generates network observations.

The sliding window strategy has a few advantages: the number of generated observations from a single *Timebar* increases, and by overlapping different portions of the same data

behaviors that shift along the day are captured without being considered an abnormality. It also allows having quicker response times since the overlap portion is the response time. This means that a window of one hour, with an overlap of 1/3 of its size yields a response time of twenty minutes.

As Table 3.2 shows, different window sizes generate different response times (equals to Step Size), with a trade-off between detecting events and patterns over a long period or a small period.

Table 3.2: Relationship between number of observations generated from a *Timebar* according to the window size and sliding window step size

Window size	Number of observations	Step size
24 hours	1	–
6 hours	10	2 hours
2 hours	34	40 minutes
1 hour	70	20 minutes
15 minutes	286	5 minutes
5 minutes	862	1 minute and 40 seconds
1 minute	4318	20 seconds

However, if the sliding window narrows too much, periodic events longer than the window range are not captured by the wavelets (capturing periodic events is discussed in Section 3.2.5). So it is all about finding and experimenting different window lengths in order to have balance what is suppose to be captured in the network activity patterns. Sliding windows at human scale trends to capture human behavior. For example, casual browsing generates periodic events in the scale of minutes that would be therefore undetected if the sliding window is shorter than a few minutes. On the other hand, capture machine-level protocol activity happens in a different scale: as for instance, when a webpage is loaded, HTTP exchanges small packets at the beginning, followed by a large download followed by a silent period before the next page: this is a periodic pattern detected even small time windows. As a final illustration, a window for the whole day would detect a periodic event that happens every hour, such as hourly data backups.

Whenever the size of the window, the features extracted need to model the general entity activity, machine or human to infer the occurrence of different kind of behaviors, meaning that *Counter* attributes are yet to be transformed in ML features and observations, which are explained in the following Sections.

3.2.3 Network and behavioral modeling

Taking into account the provided *Counters* attributes, there is a continuation of the process to generate values according to a view that expresses the characteristics of the network activity. One of the strategies to compute features is windows functions. Each feature computation happens inside of a sliding window and data is interpreted in different ways that try to extract the statistical value from it. As Figure 3.5 shows as an example the computation for each

individual *Counter* attribute, where may, for instance, be calculated its mean, skewness, kurtosis, and variance.

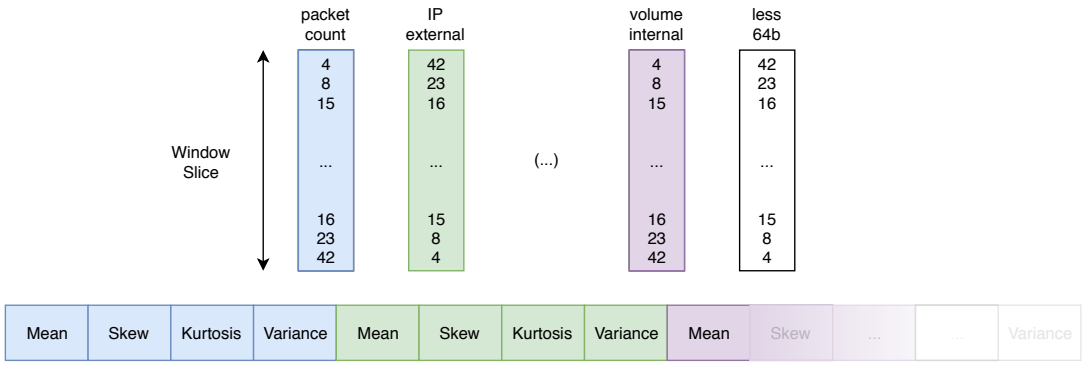

Figure 3.5: Representation of how observation features are computed from *Counter* attributes. Final features are related both with *Timebar* slice ΔT and the sliding window length.

Since ML is an excellent tool for detecting patterns, these values feed statistical data that that provides insight into the shape of the data. In the case of value outliers, it provides the information needed to indicate if the values inside of each window are in the within an acceptable range and therefore if it should be considered normal according to the past observations of an entity.

Complementary to parsing *Counter* attributes and to provide ML algorithms, another view is a hint of the relationship of activity, meaning that another possible feature is the number of zeros inside of each window. This information gives an insight of both of the entropy and completeness of the window. Figure 3.6 shows this data is not parsed by columns but as a whole.

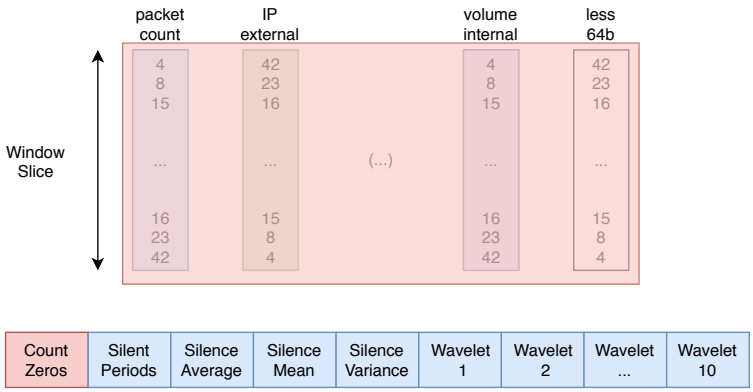

Figure 3.6: Representation how some features require using the entire window. Silent periods or Wavelets are highly dependent of the Window Slice length.

The described views are a summarized critical thinking that must be exercised in each modeling, depending on the context. Network patterns tend to fall into extracting statistical values, as the suggested above and is one of the most generalized and abstracted views despite

the fact of the risk of the Simpson's paradox[1].

3.2.4 Periodic events

Periodic events can happen in different layers, may be subject to filtering, parsing and depends on prior analysis before trying to detect such periodicity. For instance, detection of periodic network patterns could ignore network-wise protocols and try to catch only entity-generated periodic events: in the case of a download it would ignore the pattern of the protocol used to download a file but would detect the fact that a file is periodically downloaded. As discussed in 3.2.1, targeting what the periodicity which is attempted to be detected defines how to adapt the view on the data.

Even though the granularity and specificity could vary, it is undeniable that knowledge of existing periodic events has a significant role in distinguishing the inherent generator. In the context of work developed in this dissertation, and not despising the importance, events are as generalized as possible, being signal processing field by itself a whole study domain. With this in mind, packet count provides the necessary data to discover the silent periods counting, mean, variance and to calculate rudimentary periodic events with Wavelets[2]. The expected result is a discernment between machine-level and human-level pseudo-events: the information that something is somewhat periodic (e.g., a server that runs services or human scale habits) or just deducing from something that has no periodicity at all.

3.2.5 Features that describe human behavior

Some factors make hints for the human factor that aren't directly related to the captured data. Employees tend to have higher variability in their behavior than computer traffic, but their activity is always somehow related. For example, at night fewer people using computers is directly related to less traffic on servers. However, there may be such variability that could affect the detection of abnormality. The feature that relates the observation with the weekday, weekend or holiday was taken into account to infer abrupt pattern changes. The position on the day also influences the captured data. Employees usually have daytime working hours and scheduled lunch breaks, meaning that if the position of the captured data in the day were not taken into account, it might be possible add too high variance to the observations, attenuated with this association.

As a consequence, modeling the human factor should take into account human variations. For instance:

- Day variations

  Holidays, weekends, social events tend to change abruptly a human pattern that will be expressed in network pattern change. If this change is acknowledged then this shift in observations is taken into account.

[1] Also known as reversal paradox or amalgamation paradox, which is a phenomenon in which a trend appears in several different groups of data but disappears or reverses when these groups are combined

[2] Morlet's Wavelet is one of the possible examples

- Pseudo-periodic patterns and scales

  Human periodic patterns happen in a different scale that machine-to-machine communications (which tend to have lower variance) and are not quite periodic. For instance, even the pattern of reading web pages have a high variation in the network (e.g., a few bursts of data every few minutes or every few seconds) which are more difficult to model.
- Other features

  The list is not closed. A model could also take into account if an entity uses always the same gateway/switch or not. This trend to have a more chaotic pattern can also be modeled.

With the set of features complete, data must be still pre-processed to give good output results. In line with the work of this dissertation the following problems are taken into account:

- Inaccurate data or missing data

  In the case of entity activity, missing data has the meaning that no activity occurred. When the probe did not collect any packet for a given period all *Counters* are set to zero, and the observation is still taken into account. This means that, for example, an entity is usually powered off during a specified period of the day and in what period of the day.
- Noisy data

  Ideally erroneous and outliers would be filtered out from the training sample. However, a high variance of the data may be periodic over the days and not just at one training observation. For example, backups that occur every Wednesday that generate a high volume of traffic and which in turn contrast with the normal median and variation of an entity. This spike that contrasts with the data set should not be filtered despite the fact of causing a high variance on the training set. On the other hand, low data entropy does contribute to the harder differentiation of patterns. Machines or entity idling trend to generate such low volume that it looks too much system-wide identical. A threshold could be set that the Counters would be taking into account only if their entropy was higher than a certain value.

For most ML projects, data cleaning, removing or filling would be a part of the pipeline before pattern recognition. In a particular point of view, the absence of packets directly translates into data that is still part of the training set. Even defining a threshold of minimum data entropy would generalize too much what is below the considered minimum being some of the periodic low-volume events pass be unnoticed.

3.2.6 Features dimensiality and completeness

Although more features could be added, it does not necessarily mean increasing the performance of the classifier. No rule defines the ideal number of features in a classification problem and depends on the amount of training data. The smaller the number of training samples fewer features should be used. The number of training samples should grow exponentially with the number of the number of the dimensions used.

In fact, increasing to much the dimensionality by adding too many features would cause *The Curse of dimensionality* [3]. This happens because ideally would be an infinite number of training samples. Since this is impossible, for n number of features should be such that *feature dimensionality* $\ll$ *training samples*, and increasing the number of features without increasing the number of samples results in a decrease of classifier performance [134].

In practice and since there is a limited number of samples possible a compromise must be made. Selecting the optimal subset of *training observations* for n dimensions could be aided recurring to two ways: The first is the use of feature selection algorithms that employ heuristics to locate the optimal number and combination of features. For instance, removing features with high variance (and testing them with cross-validation as discussed in 2.4.4). Another technique is to replace N features by a set of M features, each of which is a combination of the original feature values. There are algorithms that try to find the optimal linear and non-linear combination of original features to reduce the dimensionality. This process is known as *Feature Extraction* and a well-known one is Principal Component Analysis (PCA), as shown in Section 3.3.4.

Another problem of excessive dimensionality is the computational complexity. As for example, computational complexity for SVM is approximately between $\mathcal{O}(n_{features} * n^2_{samples})$ and $\mathcal{O}(n_{features} * n^3_{samples})$ [135]. Adding too many features results in large computation times that for some algorithms would take unpractical amount of time to converge.

[3] Also known as *Hughes phenomenon* or *peaking phenomena*[133]

3.3 High level data overview

3.3.1 Dataset composition

Once all the *Counters* values are flattened into a model that express entity activity patterns, ML is mostly ready to infer the inherent patterns that distinguish each entity of another and if operating on its usual behavior. When loaded into the pipeline, and since this is a Supervised ML project, every observation of the training set includes a label, which is the entity that originated the observation.

As described in Section 3.2, modeling activity implies having samples that all features are zero, meaning that the entity did not generate any data on that period. For example, a machine that as an employee associated is much probably turn off during the night, and therefore it does not generate traffic. This information is needed because if it starts to have night activity it may be a malicious one. The number of observations is directly related to the length of time of the captured data and window slices sizes. In the case of the suggested network modeling the number of the feature columns is fixed at 64, including the label as explained in Section 3.2.1, and all the attributes are numerical.

Overviewing the data is the ideal starting point to have a feel of its nature. For instance, looking at the count of observations per entity, the mean, min, maximum, standard derivation gives a quick way to understand the shape of the data. Standard deviation and percentiles measures how dispersed the values are against each other, the count of observations per entity provides a quick way to know if the samples are too unbalanced, biased, tail-heavy, and so on.

Figure 3.7 shows observations from a server. This is another useful way to understand data nature is plotting a histogram that shows the number of instances (on the vertical axis) that have a given value range (on the horizontal axis).

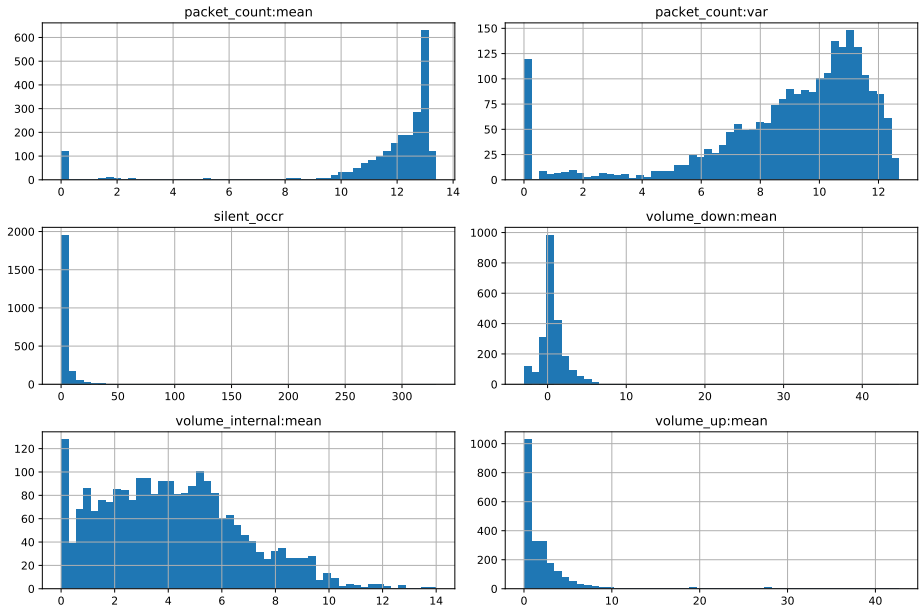

Figure 3.7: Histogram of some features from Lab-D observations. X-coordinates are the observed values and Y-coordinates the count of occurrences.

Histogram for this entity shows that silence occurrences (number of periods with no data) are infrequent, meaning that this entity exchanges packets almost every second, matching a typical active server.

Since features are collected and generated with the network model in mind, there is no particular data filling or removing needed. The preprocessing and the nature of capture process leverage the data post-treatment.

3.3.2 Dataset splitting

The data set is afterward split into training, testing and validating set. The data could be shuffled and then divided according to the needed portions[4], but to avoid any biasing, Stratified sampling has a few advantages over random sampling. Especially in the case of datasets not large enough, when there is a higher risk of introducing significant sampling bias. In the case of network observations, where the number of collected network observations for each entity probably varies due to machines being added or removed, random shuffling is even riskier. Stratified sampling tries to ensure that the split divides train and test set into homogeneous subgroups and that the test set is representative of the overall data. In the case of a randomized split, that would possibly cause a change of sampling with a skewed test set for each of the entities.

Equation 3.1 represents a split benchmark, in which the mean of the training set divided by the testing set is approximately one, and the same happens if the mean is calculated for any of the subsets, as:

$$\frac{mean(Training\ set(Lab-A))}{mean(Testing\ set(Lab-A))} \approx 1 \tag{3.1}$$

Although more analysis could be done, the split sampling using this method is unbiased across all subgroups and as a whole, meaning that the testing data set is representative of the overall training set.

3.3.3 Dataset correlations

One way to calculate the relationship between the data is to compute standard correlation coeficient[5], between every pair of features. For a linear correlation between the most correlated features, a scatter matrix between the most correlation ones represents such relationship. Figure 3.8 shows correlation paired with different types of features. The main diagonal (from top left to bottom right) would be full of straight lines (each feature directly correlates with itself), so instead it is plotted the representation of a histogram of each diagonal feature.

[4] Usually following Pareto Principle of 80% for training and 20% for testing

[5] Also known as Pearson's correlation

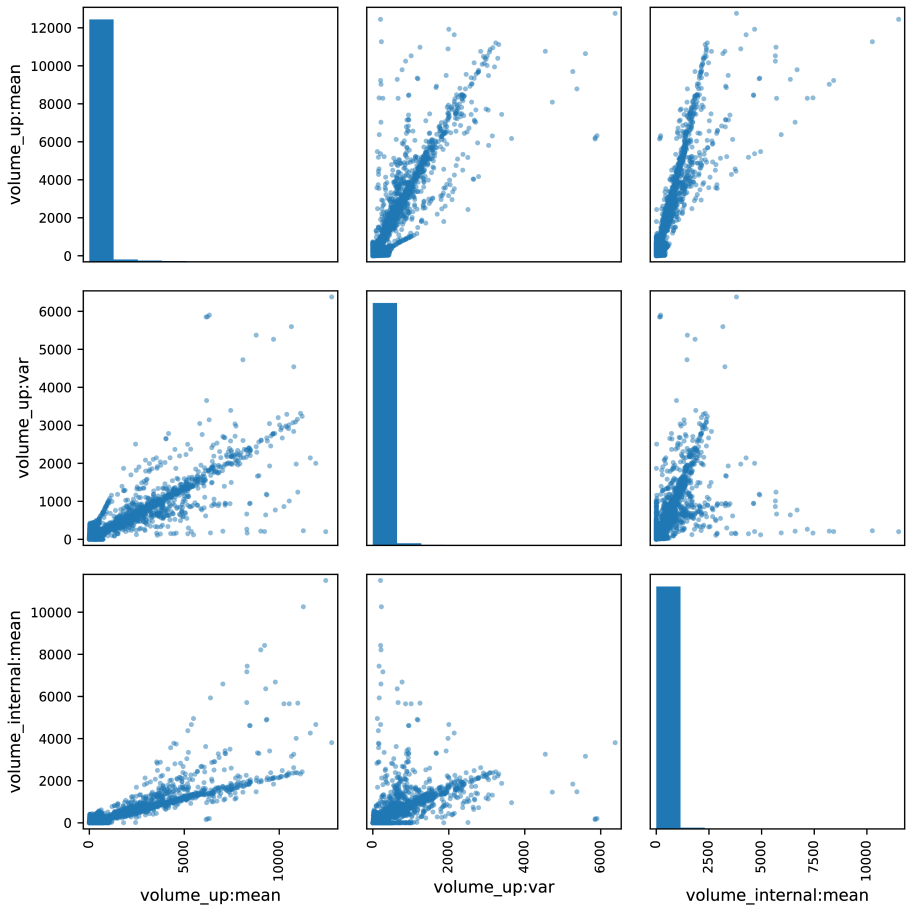

Figure 3.8: Matrix of the data most correlated features of the data set. Diagonals are feature histograms. Graphs show the linear relationship each pair of features.

In this example, the correlation plot reveals a few things. Firstly that correlation between the volume upload mean is strongly related to volume upload variance. However, this could be related to traffic bursts on servers. High-speed bursts that generate high volume in a small time window, justifying the relationship with the high variance. Interestingly, all correlations look tail-heavy, meaning that are high potentials to transformation (e.g., computing the features into a logarithm scale) when experimenting with different ML algorithms. Lastly, all these relationships are features with high potential of being combined into fewer ones with PCA, as described in the following section.

3.3.4 Feature analysis

Training an ML algorithm with an unoptimized number of features does not only makes it slower but can also affect system's ability to find a good solution[6]. Adjusting feature dimensionality does lose some information but can also filter some noise and unnecessary details, resulting in higher performance (discussion about performance metrics in Section 3.5.2.1). Reducing the number of dimensions to two or three helps to project the data and

[6]Also known as *The Curse of Dimensionality*

therefore making visualization possible (as seen in the Section above), thus providing some insights by visually seeing the patterns (e.g., clusters).

Features that describe the entities network (as discussed in Section 3.2.3 and 3.2.5) patterns are divided into subsets of features:

- **Network statistics** (Group 1): packet count, IP count, Port count, TCP FIN count, upload and download volume, and so on.
- **Activity statistics** (Group 2): Silent periods, relative day position
- **Behavior statistics** (Group 3): Wavelets

Training with a limited set of features and testing it against a classifier may reveal the quality of the model and its relevance in overall distinction factor between each of the entities. Table 3.3 shows the compromise of the combination of different each group of features that try to model different behaviors.

Table 3.3: Relationship of the features groups and its impact in classifier accuracy. The influence of each group hints on the models quality discrepancy of the patterns of the entities.

Group kept	Features
2+3	16
1+3	59
1+2	54
3	11
2	6
1	49

As feature reduction and combination, PCA is the most popular dimensionality reduction algorithm. The algorithm identifies the hyperplane that lies closest to the data and then projects the data on it while trying to preserve the maximum amount of variance - as it will much likely lose less information than a projection with lower variance. Another metric used is the mean squared distance between the original dataset and its projection in one of the axis. The PCA algorithm reduction executed is a SVD decomposition [136]–[138]. Figure 4.3 shows PCA dimensionality reduction, which works as a factorization mechanism. Figure 3.9 shows an example of a PCA reduction.

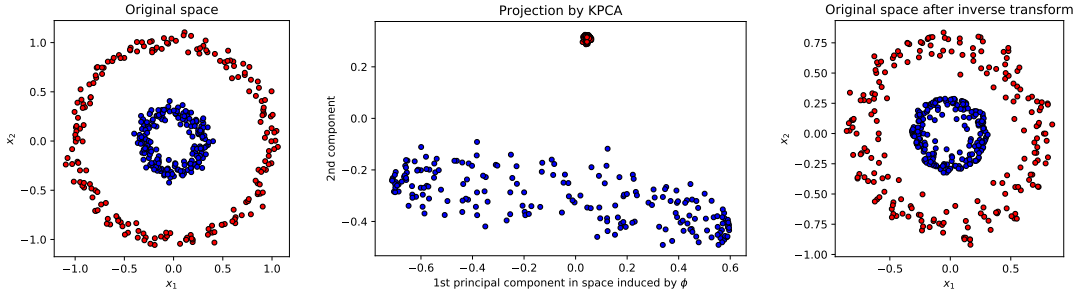

Figure 3.9: Example of a dataset, the resulting applied PCA reduction and, as for example, the inverse transformation that shows that the reduction kept almost all information

From left to right, the first image is the original data, the second is a projection using PCA and the last the inverse transformation of the second image. The reduction was able to find a projection of the data that makes data linearly separable.

3.4 Knowledge extraction

3.4.1 Data pipeline

Data, as it is described in the sections before, is practically ready for consumption by a ML algorithm. To automate the process, as Figure 3.10 shows, data flows through a pipeline that allows experimentation and a combination of different ML strategies with different data views/processing.

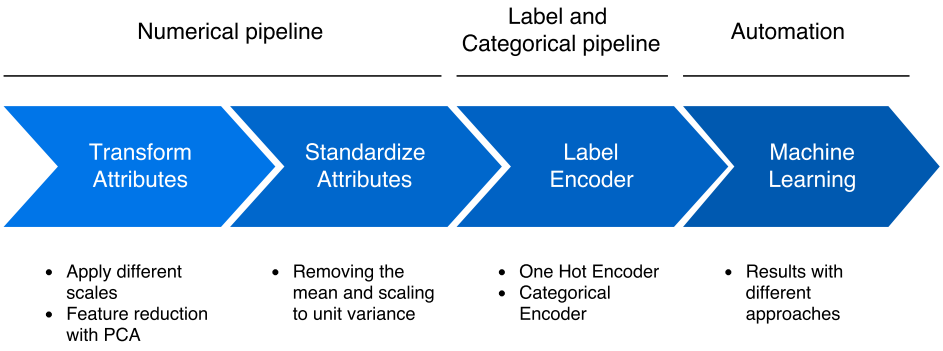

Figure 3.10: Representation of the observations data set pipeline. Data is transformed and adapted to be suitable for Machine-Learning algorithms.

Each step can be described as:

- Transform Attributes

  Feature transformation can, for example, scaling of tail-heavy features, well as the combination of multiple sets of features to reduce noise and increase the result accuracy.
- Standardize Attributes

  In the particular case of NN that use sigmoid activation functions (e.g., Tanh[α] or cosh[α], commonly used in gradients) the standardization of features helps ML algorithms to improve their accuracy and speed [139]. Independently for each feature, the mean is removed and scaled to unit variance. The mean and variation are stored and used on newer data.

  Other algorithms such as SVM benefit from Scaling, since they assume all features are centered around zero and their variance is in the same magnitude [140].
- Label Encoder

  There are multiple strategies when encoding the labels, especially for supervised clustering projects. For example in the case of the proof of concept of Chapter 4, labels must be encoded from text to numbers. Solutions to this range from classifying each label sequentially, or to create a matrix using one-of-K scheme (one column per label).
- Machine Learning

  The last step is the usage and experimentation with various ML algorithms since there is no prior indication which will work better beforehand (as discussed in Section2.4.4)

Once created the pipeline and all the attributes prepared, selecting and training the model a straightforward process. Performance evaluation and precision metrics are done using cross-validation strategy (as discussed in section 2.4.4). This method performs *K-fold cross-validation*, meaning that it randomly splits the training set into, for example, ten distinct subsets called *folds*. Then it trains and tests each ML method ten times. For each attempt, are picked different nine folds for training and one for evaluation. The final result is an array of ten evaluation scores.

3.4.2 Entity Classification

Identifying overall network behavior depends on the underlying performance success of identifying what is generating activity, and therefore identifying its entities/clients. If a new network observation leads to a successful entity classification, it means that network traffic has know source matching a known pattern. Straightforward classification methodology covers a board set of problems from the engineering point of view: it matches an entity with a score and generalizes the current NSA at the network level. Figure 3.11 shows the how classification directly leads to decision.

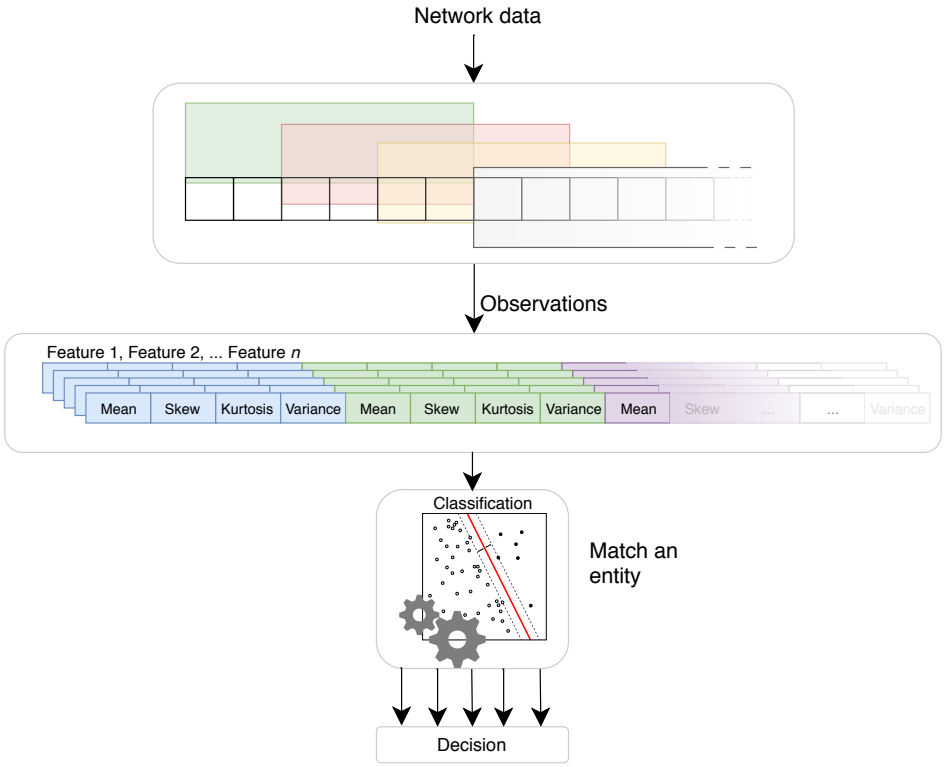

Figure 3.11: Representation how network data translatess into a decision. When performing classification, the classificator decides which ether the entity of the observation corresponds to.

However, this decision block comes with a few disadvantages. When network entities are added or removed the whole block must be retrained in order of the possible different inputs match the outputs. This is because classification algorithms are roughly divided into two

groups: nonprobabilistic and probabilistic-based. Nonprobabilistic converge into a solution by using *Reduction Funcions* and *Minimization functions* that fits weights to optimize the output. On the other hand, probabilistic systems take into account the pure statistical expression of the data and make a decision taking into account the different proximity of the new samples to the known ones [141]. Either one needs training, which means adjusting the output according to known data and may not be suitable for a dynamic network where entities are constantly being added or removed. Advantages of using such are mainly on network systems where the number of entities rarely change and the combination of such output with other IDS systems by providing a high-level network overview. However, direct classification information alone isn't enough for a more complete NSA. Recent approaches to anomaly detection using machine learning suggest using a regression model that maps abnormal data against a training model and the usage of outlier detection rather than classification. A deeper evaluation comes with anomaly detection in the following section.

3.4.3 Anomaly Detection

Classification allows the clearance of the entities operating in a network. However, it does not identify at an entity-level if the behavior is according to the expected. Another, more fine level of awareness is to know if the entity is behaving according to its patterns. Figure 3.12 shows the difference to the classification architecture of Figure 3.11. Each entity has its pattern being recognized, meaning that they are controlled individually. The advantage of entity-wise anomaly detection is a better adaptation to more volatile networks: when an entity is added or removed only the entity model must be trained and not the whole system. However, it requires a more comprehensive interpretation of the results. Multiple classification entities may state an abnormal state - which requires reviewing, or in the worst case scenario: the pattern not matching to any of the entities.

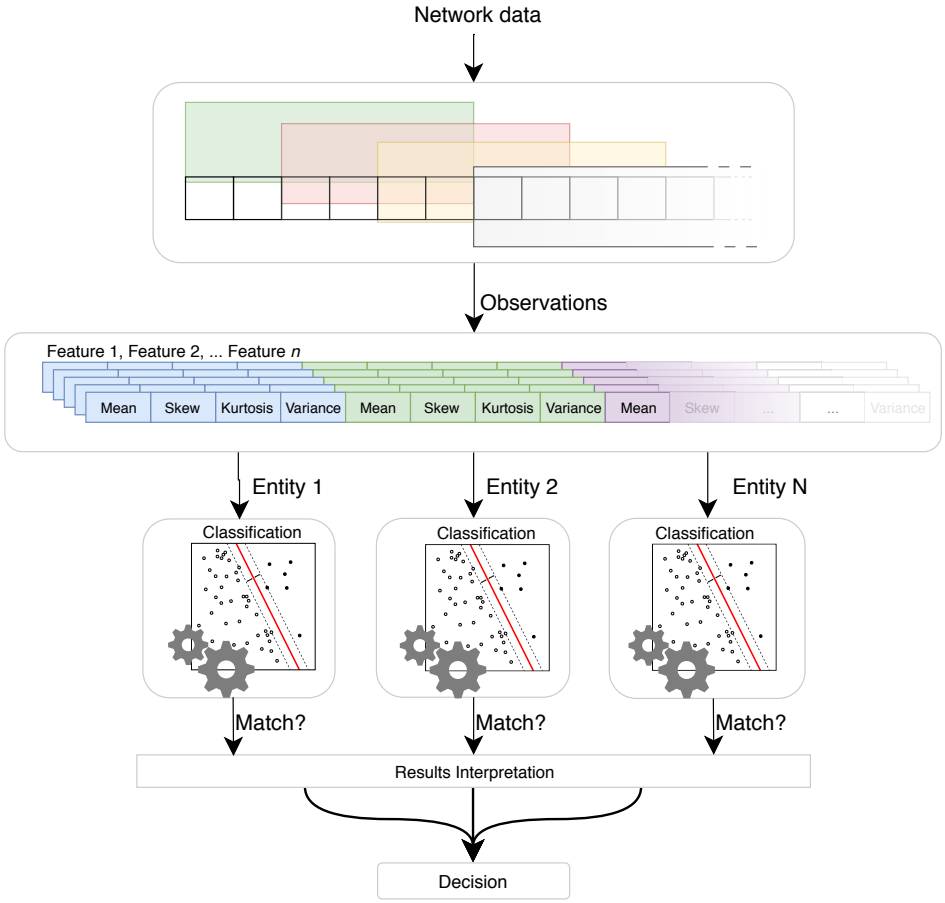

Figure 3.12: Representation of the flow from the Network Data, its conversion into features and observation, classification until a decision of the current Network Security Situation Awareness

The network data is gathered and converted into features (as explained in section 3.1), but instead of classification, each new observation is compared against N ML trained instances that recognize if the sample matches to any of entities behavior. The output is afterward processed in a new layer, human, automatic or even heuristic. For example, if the sample does not match any of the entities, it is probably an abnormal behavior in the system and can afterward be compared to the closest entity behavior. If it matches only one of the entities, then the network is in its normal state. All in all, it allows asserting the entities act according to the past and recognizing behavior changes. Results are discussed in section 3.4.3.

3.5 Knowledge process evaluation

3.5.1 Network entities classification

3.5.1.1 Statistical classification

As discussed in Section 3.2, feature engineering tries to find the relationship in data that links the network activity to the entities generating network traffic. Depending on the quality of such features, network activity inherently expresses patterns, which in turn has statistical and probabilistic expression. For example, a fileserver will have a higher median for volume download and packet size, that contrasts with an employee that works in human resources and has many video calls - low median packet size and high packet count.

Working under these assumptions, benchmarking the model quality (Section 3.2.3 and 3.2.5) is also evaluating if classification is possible by looking purely to the statistical expression of features. Each feature has a median/skewness/standard derivation, which computed from multiple observations will have a centroid that works as an entity self-reference. Figure 3.13 illustrates the concept.

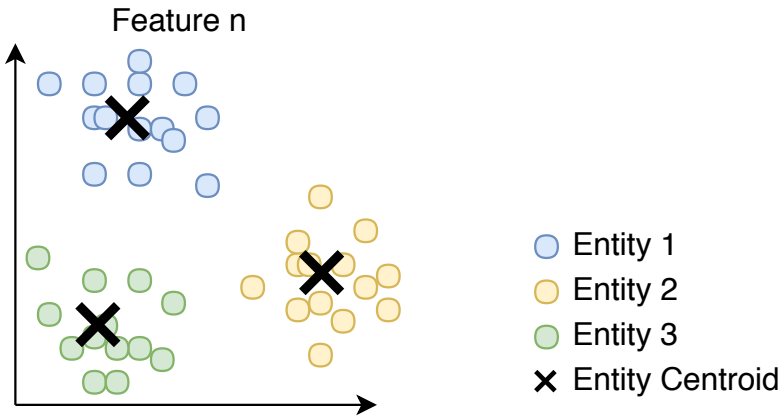

Figure 3.13: Entity feature centroid is calculated from the mean of each feature

When a new observation needs classification, and in the case of centroids calculated from feature mean, matching is performed without any additional computation since the new observation attributes are matched to known medians. As Figure 3.14 shows, the classification for new observations is the based on the Euclidean distance (or other distances metrics) to each known centroid. The less the distance, the higher relationship of known past patterns.

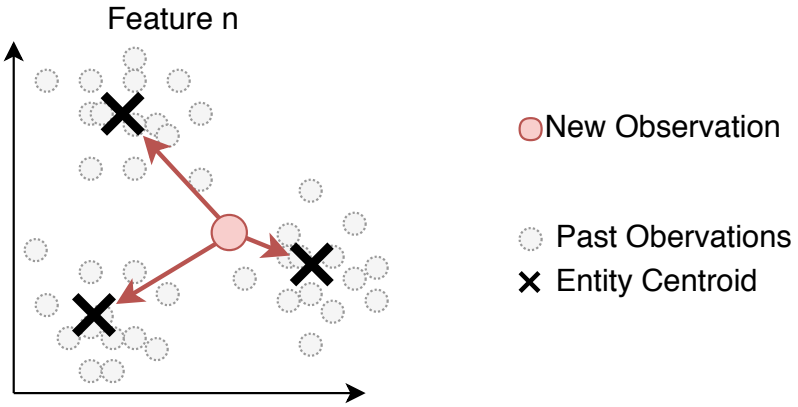

Figure 3.14: New observation is compared to the existing feature entity centroids, being the closest the most probably related with know observations

Therefore, classification is based on distances to known centroids. Due to its simple statistical mechanism, this method may be a good indicator if the chosen network model does, in fact, expresses different entity behaviors and patterns, and if those are distinct from each other.

Statistical analysis and benchmarking can be extended (and ultimately, ML are fine composed statistical models) to other analysis. Dataset histograms, such as seen in Figure 3.7, by definition immediately express its own Probability Density Function (PDF). Since there are multiple features, the PDF can be extended to multivariate Gaussian distribution (and other distributions), in which combination of the features defines a distribution parameter.

A new observation classification is therefore based upon the proximity to a known distribution mean and covariance, of the current past classes observations.

3.5.1.2 Support Vector Machines

ML combines many kinds of statistics, approximations, and data separation techniques, many with the same concepts as the Section before. Numerous algorithms can be tuned by adjusting parameters to react better to the dataset and improve the decision efficiency. The high-level concept of SVM - a ML algorithm - is to find the boundaries between known distinct samples. These boundaries define how the algorithm decides the classification of the observations, hence the decision of choosing on each label the observation corresponds to. SVM is sensitive to feature scales, meaning that the pipeline must (Figure 3.10) include attribute standardization. There are several SVM algorithms, offering diverse options for parameter tuning with different mathematical formulations. *LibLinear* is a linear SVM algorithm implementation, winner of *ICML 2008 large-scale learning challenge* and one of the most renowned ones. It is often recommended as the ideal multi-purpose ML algorithm [89], [132], [142].

Many of the ML algorithms require labeled data, and some of them (like SVM) are binary classifiers capable of distinguishing just to labels. Since this is a multi-label classification problem, the system must include a procedure to deal with this limitation. One way of doing it is the One-vs-Rest (OvR) strategy, in which are trained the same number of models as the

classes to classify. In the case of the Proof of Concept (PoC) of this dissertation an since there are six probes, then six binary classifiers are trained. Another strategy is One-vs-One (OvO), in which trains a binary classifier for every pair of probes. In this case, with the same six probes, means that the system trains $Nx(N-1)/2$ classifiers, in a total of fifteen classifiers.

In addition, training an **SVC!** (**SVC!**) algorithm generally takes into account the following:

- **C parameter** Penalty parameter C tells the SVM algorithm how much to avoid misclassify each training sample. For example, large values the optimization will choose softer margins, and all training trend to be classified correctly. Small values will make the hyperplane to have a larger margin and misclassify more points.
- **Loss parameter**:

  For the loss function, 'hinge' as the used as SVM maximum-margin classification [89], [143].
- **Multi-class Strategy**: OvR

  As the multiclass strategy, since the training set is composed of more than two classes, OvR strategy.

Calibrating the parameters above, especially the balance of the C parameter, requires studying the training set, the nature of the data being separated by the hyperplanes, and the ratio of the outliers in the samples. Figure 3.15 represents how the C parameter tunes the bias-variance[144].

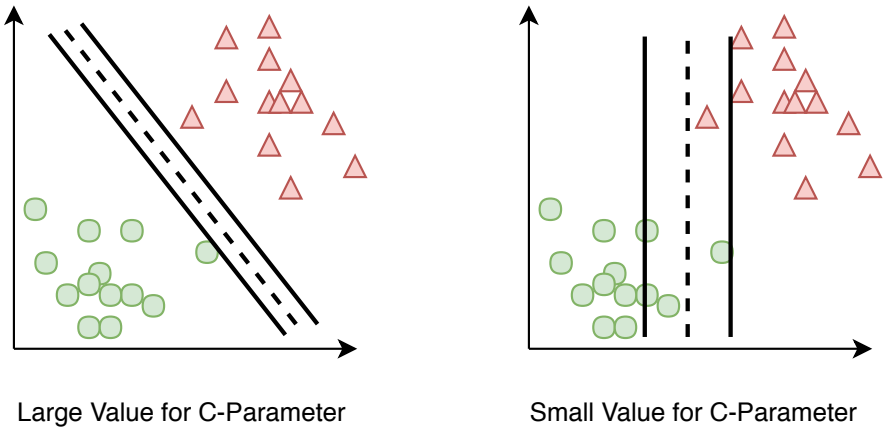

Figure 3.15: Visualization of the impact of SVM's C parameter separation margin and the tolerance of the outliers in such spearation

A larger C parameter means a higher penalty for misclassification and small values trend to generalize outliers better, thus avoiding overfitting the data. The algorithm accuracy calculation is done according to Equation 3.2, a metric for classification problems such as this.

$$Accuracy = \frac{Correctly\ classified\ samples}{Total\ numberof\ samples} \tag{3.2}$$

Accuracy is the most straightforward metric to evaluate model performance, and many others must be taken into account before deciding if a model fits as a solution for a given problem.

3.5.1.3 SVM Classification Error analysis

A more delicate evaluation also means interpreting what accuracy means in the context of a classification problem and, depending on the purpose of the system how the classification is performed along the classes. *Confusion Matrix* helps to evaluate a classification by analyzing how many times a class misclassifies a class. Figure 4.5 shows the computed confusion matrix, using K-fold cross-validation [7], guaranteeing that evaluation is made using the different combinations of training and classification folds. Prediction made by a model is made over data never seen during the training phase.

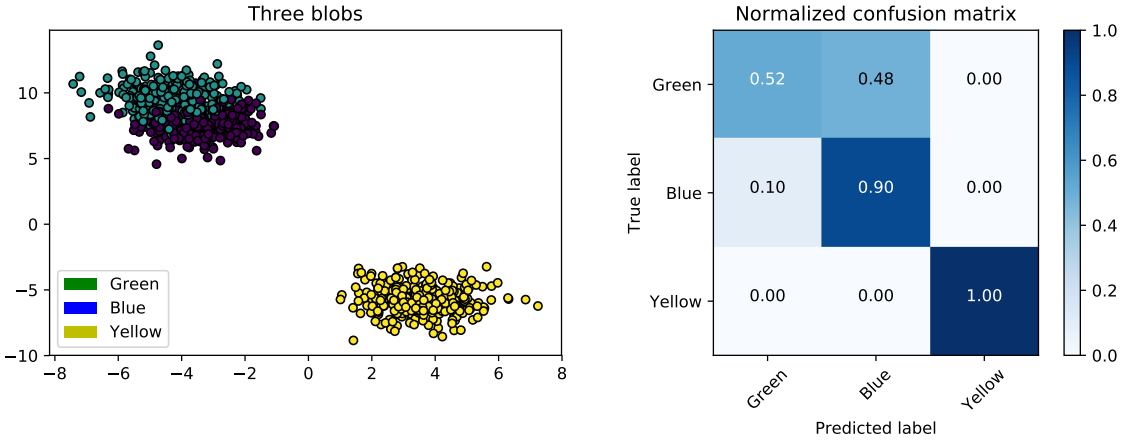

Figure 3.16: Example confusion matrix for the dataset on the left. Confusion matrix helps algorithm interpretation by knowing True Positives, False Negatives and their combination according to the labels.

Rows of Matrix of Figure 4.5 are the actual classes and columns are the predicted entities. Darker color means correct classification. In the case of this dataset example, there are incorrect classifications between the Blue and Green labels. There is also fewer matches of the Green when the label is Green, meaning that this feature requires a better model that helps it to differentiate from the Blue label.

3.5.1.4 Neural Networks

As shown in Section 4.3.1, even though SVM performs excellently when classifying network patterns and behaviors of the entities of Chapter 4, fails at scalability. While some solutions and adaptations exist, SVM is not an online algorithm and does not support incremental learning (as discussed in 2.4.2.2). Deep Learning, more know as Neural Networks[8] and its derivatives are gaining popularity as the availability of enormous amounts of data are more common, including the increasing need of having real-time results adapted to each instant. Examples include classifying millions of images (e.g., Google Photos and Google Images), speech recognition services (e.g., Apple's Siri or Amazon Alexa), recommending millions of

[7] K-fold is explained in Section 3.4.1

[8] And many other names depending on the implementation, such as *Multi-layer Perceptron*, *Artificial Neural Networks*, *Deep Neural Networks*, and so on.

suggestions to users every day based on their history (e.g., Youtube and Spotify), and so forth. NN seem to have entered a virtuous cycle of funding and progress due to the fact of products based in this kind of solution often make headline news, pulling more attention and funding towards its development, resulting in more and more progress and even more amazing products.

Despite the fact that NN is not a new concept[145], the tremendous increase of computing power due to the Moore's Law and the gaming industry that produces millions of GPUs, combined with a few small algorithm tweaks that had a substantial positive impact, made it possible the train of large NN in a reasonable amount of time, thus outperforming other ML algorithms for massive datasets. It is now expected that this wave of investment in NN will impact even further the research and development, with significant impact in our lives, making NN future-proof for the incoming years.NN is already adapted to intrusion detection and, as shown in this chapter, can also solve the problem of network pattern recognition [146], [147]. While NN offer some advantages over SVM, it also has a few disadvantages, such as: variability of initial initialization values; requiring calibration of hyperparameters like the number of hidden neurons and iterations; and sensitive to feature scaling (as discussed in 3.5.1.2).

The NN structure can have different implementations. The simplest one is one *perceptron* with multiple inputs weighted into one output. Multi-layer Perceptron (MLP) implements multiple layers combining multiple perceptrons and internal organizations, such as having or not backpropagation or bias neurons. NN PoC of Section POC:NN classification training is implemented thought *Stochastic Gradient Descent*[148], being *Adam* one of the most used algorithms [149] to run it. Since *Adam* algorithm trends to perform better with huge datasets, an alternative for relatively smaller training sets is the *L-BFGS* algorithm [150], [151]. It performs faster due to the fact of *L-BFGS* not being an online algorithm, making it useful as a heuristic to find out if NN approach fits a specific problem, such as the one studied in this dissertation. Both algorithms are implemented by the SciKit-Learn framework and are the ones tested in the following sections.

There are a few recommendations when initializing a NN solver, such as:

- **Feature Scaling**: although it depends on the activation function, each attribute is scaled to [-1,1] with a mean of zero. Already included in the pipeline discussed in 3.4.1
- **Calibrate α parameter**: finding a reasonable regularization parameter. Usually within the range of 10^{-10} to 10^{-1}
- **Test L-BFGS algorithm**: as said before, converging faster and performing better for relatively small training sets, such as the one used in this dissertation.

3.5.1.5 Ensemble methods

As discussed in Section 2.4.2.1 and the section before, ML algorithms perform generalizations still fail to adapt to mixed data sets as Figure 3.18 shows. Meta-estimators such as *AdaBoost*[9],

[9]Yoav Freund and Robert Schapire won 2003 Gödel Prize for AdaBoost meta-algorithm work

refers to an Ensemble method that combines several weak learners into one strong learner. The idea is to train and combine a sequence of predictors in which each one tries to correct its predecessor. Once again, there are several implementations of boosting being *AdaBoost*[152] and *Gradient Boosting*[153] the most popular ones.

In the case of AdaBoost, SAMME algorithm[154] extends the original Boost algorithm with hyperparameters for the learning rate, meaning that for each predictor of the pipeline the system senses and then weights each predictor (called weak learners) according to its accuracy. The system is tweaked in favor of the misclassified samples of the previous classifiers, meaning that is prone to overfitting and sensitive to noise data, although that can depend on the nature of the training data.

Figure 3.17 demonstrates the process of iterating until the desired number of predictors is reached or when the system finds the perfect predictor sequence - one strong predictor that represents the final output of the boosted classifier.

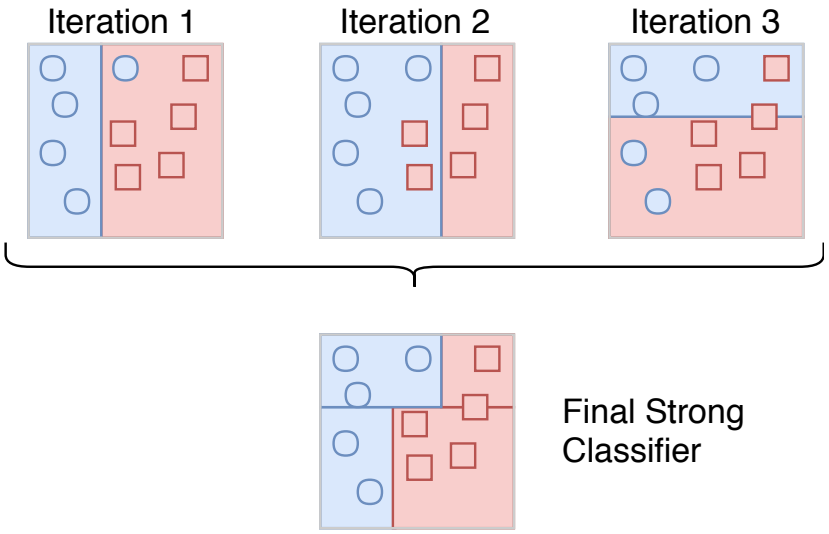

Figure 3.17: Representation how ensemble methods iterate over the each own results and try to iteratively stack more classifiers to self adjust to edge cases

Like any other ML algorithm, results depend on the problem type and the amount of work of combining all the possible hyperparameters before archiving optimal performance. However, even though SVM and NN were discussed, the nature of the traffic model of having subgroups of some network entities, AdaBoost could not be ignored due to its features and due to the fact of being often referred as one of the best out-of-the-box classifier [155], [156]. The loop of correcting the weights of each preditor on each iteration, fitting new predictors to residual errors is, in fact, working as a sub-class distinction. AdaBoost is particularity interesting because its features fit in favor of this dissertation use case: the existence of subgroups in data, like the baseline classes of Section sec:Network Data Collection2.

Figure 4.11 shows the results of applying AdaBoost-SAMME combined with LinearSVC, as used in Section 3.5.1.2. The train parameters are for how long should the boosting algorithm continue to combine and add more estimators.

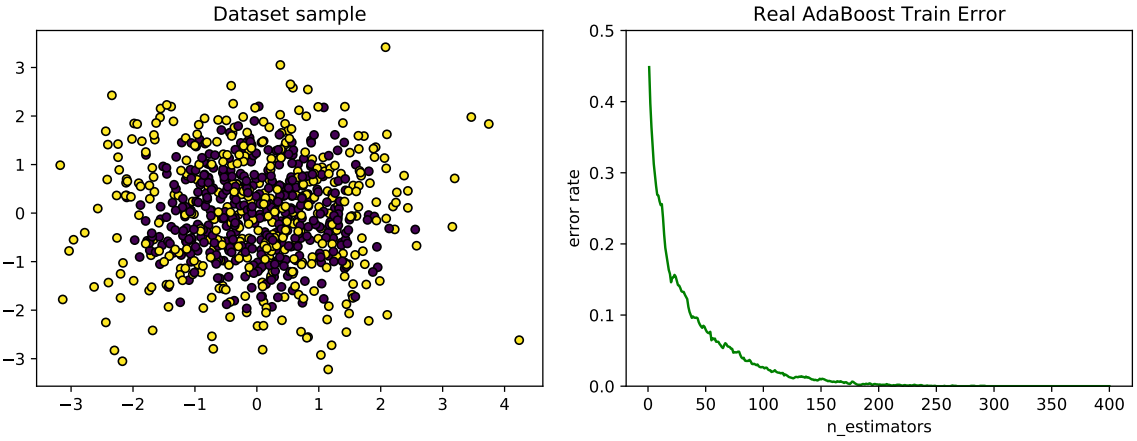

Figure 3.18: Error rate over the required number of estimators (classifiers) in order to archive perfect classification rate

The number of used estimators reduces the error rate until a hitting a rate threshold or until reaching a perfect classification. In such scenario, as the data set on the left in Figure 3.18, no other algorithm can archive such classification, that is especially tricky tor linear classifiers. As expected, AdaBoost is hugely prone to overfitting, so special care is necessary when finding a balance between the error rate and the number of estimators.

3.5.2 Anomaly detection

Despite the fact of the lack of surveys do not review or compare different ML algorithms that perform anomaly or outlier detection, SVM trend to perform better against any other ML algorithm (such as NN, K-Nearest neighborhoods, Decision Trees and so forth) both on processing speed[10], and therefore better response time, with low false positives rates [157], [158].

Consequently, suggestions of combining abnormality detection in computer networks to enhance IDS are surfacing and their implementation trending to the usage of SVM, NN and Decicion Tree Classifier (DTC) algorithms [159], [160]. Even though the system can detect deviance in behaviors, it is still needed a way to link the specific nuances of ML systems with IDS.

In light of this, precision by itself is, by far, not the ideal performance evaluator. Along with precision, the *Recall* metric takes into account the ratio of positive classifications and false negatives, meaning that combines the results when the system says there is no abnormality when there is one and vice-versa. Therefore *Precision* and *Recall* help to understand the classifier performance for a given task, depending on the purpose of the system. The combination of the two is known as *F1*, a scoring metric that combines both and favors when the system as same precision and recall. In the case of abnormality detection, it is necessary to find a good compromise between precision and recall. High precision means that the system detects all the abnormal behaviors but will have many false positives (low recall). In contrast,

[10]Benchmark ran against DARPA dataset

a high recall will have almost no false positives but will not detect all the abnormalities[11]. The calibration between the precision and recall tradeoff depends on the scale and purpose of the system. In the case of an IDS system, combined with other metrics it is probably more beneficial to have high precision combined with a low recall, meaning that some of the abnormalities will be undetected, but all the detected ones are *true positive* classifications.

3.5.2.1 Performance evaluation

Evaluating the performance of a classification problem is not lessened to measuring accuracy. Cross-validation is a method to avoid initial biasing, miss interpretation or misuse of the training data, and when combined with the *Confusion Matrix* provides a general idea of the internals of the ML algorithms, in this case, the *Linear SVM*. Extending the concept to binary classification (that is, how the classification operates internally using the OvR strategy) the *Confusion Matrix* in Table 3.4 provides an explanation.

Table 3.4: Confusion matrix for binary classification

Classified as $\rightarrow$	$\bar{X}$	X
$\bar{X}$	True Negative	False Positive
X	False Negative	True Positive

The first row is the negative class, which considers $\bar{X}$ entities. True Negative is when the samples are $\bar{X}$ and classified as $\bar{X}$, which is correct. False Positive is when $\bar{X}$ is classified as X. False Negative is the X classified as $\bar{X}$ and True Positive is X correctly classified as X. These classification statistics are the foundation to many metrics, including the Receiver Operating Characteristic (ROC) curve and provide a comparison between models, algorithms, and insight how to improve some features. The accuracy of the positive predictions is the classifier *precision*, as 3.3 shows.

$$precision = \frac{True\ Positives}{True\ Positives\ +\ FalsePositives} \tag{3.3}$$

However, evaluating the performance just by using accuracy is not representative of the performance of the classifier and can be misleading. Another metric is often is the *Recall*[12], that as Equation 3.4 shows, is the ratio of the positive instances correctly detected by the classified - the true positive rate.

$$recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.4}$$

There is a direct compromise between precision and recall, and the calibration depending on the objective of the system, that in this case is to detect an abnormality. Due to the nature of variability of patterns of a server, a system with low recall (some abnormalities are not detected) with high precision (when the system detects an abnormality then almost for

[11] Also know as the *precision/recall* tradeoff
[12] Recall is also known as Sensitivity

sure something unusual happening) is preferred [161]. This *precision/recall tradeoff* that is adjusted according to the desired purpose. Figure 3.19 exemplifies this tradeoff for the known IRIS dataset [162] using SVM classifier.

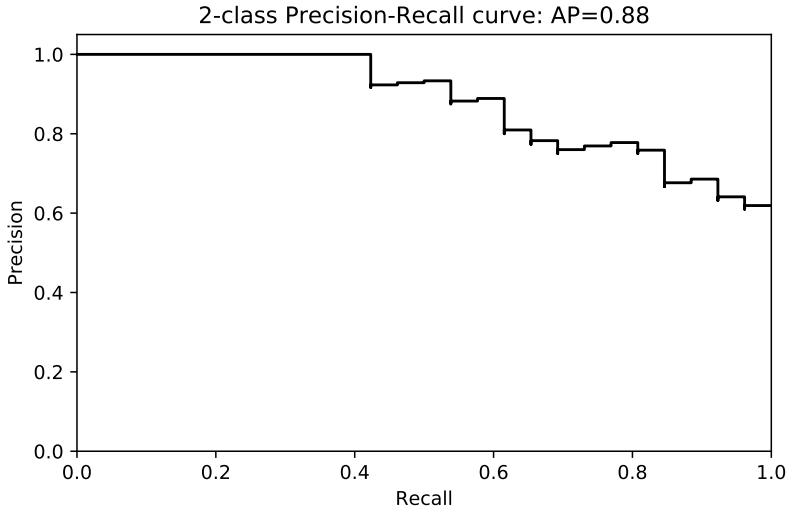

Figure 3.19: Precision-Recall Curve for binary SVM classification of the IRIS dataset, showing a decrease of Recall

The recall is the number of correctly predicted classes out of the total classes to predict. This means that to have full precision (a correct classification without any false positives) is only possible by matching only around 40% of the positive classes. To match all the positive classes, it is necessary to have only 60% precision, being all 40% the classifications of false positives. Precision is directly related to recall, and there is always compromise between these two metrics.

Combination of the precision and recall metrics is known as the F1 metric, that is the harmonic mean of precision and recall (Equation 3.5). Being harmonic mean, instead of treating equally all values, gives more weight to low values. However, increasing precision reduces recall and vice versa[13].

$$f1 = \frac{TruePositives}{TruePositives + \frac{FalseNegatives+FalsePositives}{2}} \tag{3.5}$$

ROC Curve comes to hand when evaluating binary classifiers such as SVM, that plots system performance by plotting *true positive rate* against *false positive rate*. The ROC curve has the advantage over other metrics of viewing the tradeoff between sensitivity and specificity for all the possible thresholds and if needed to compare against other classifiers. Provides a way to assess a model independently of the choice of a threshold.

Comparing various ROC curves against different ML algorithms eases and promote a straightforward comparison of the performance of abnormality detection. Inline with discussion of NN and Adaboost of Section 3.5.1.5.

[13]Know as Precision/Recall tradeoff

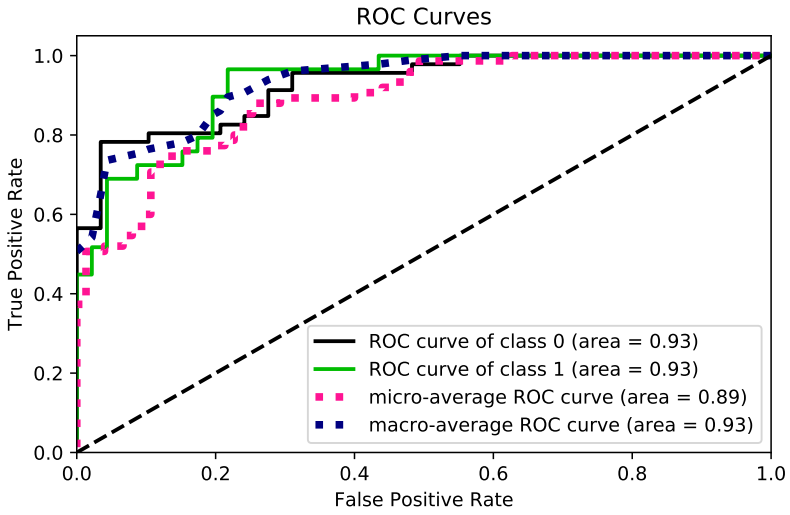

Figure 3.20: Precision-Recall Curve for binary SVM classification of the IRIS dataset, showing a decrease of Recall

Once again there is a tradeoff: the higher the recall (TPR), the more false positives (FPR) the classifier produces. The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner). One way to compare classifiers is to measure the area under the curve (AUC). A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5.

CHAPTER 4

Proof of Concept and Evaluation

In this chapter are briefly tested the methods proposed in this dissertation that classify network entities according to each network observation. The proof of concept runs against the proposed ML algorithms and using the suggested evaluation metrics, suitable for the network domain and its link to other systems, such as IDS. Both of the pipelines, classification and anomaly detection, run against a dataset collected using real-world entities.

4.1 Network Data Collection

Concerning the requirements of Section 3.1, the probe collects packet data from OSI levels 3 to 5, compatible with a document-oriented database (such as MongoDB). The implemented sniffer probe requires read access to network adapter, correct (updated) system time and date, and allows simple configuration to identify the node collection point. Due to its threaded nature, it did not add any noticeable impact on the network and system performance at consumer-level network activity, relieving the concern of the probing entity/user activity would lead to somewhat sacrifice the performance of customarily performed tasks.

Selection of entities to collect from are divided into two categories: human and no human (machine). At machine level, two of them (LAB-1 and LAB-2) work as a baseline signature of the operating system, as Table 4.1 describe:

Table 4.1: Description of machines purpose where the Sniffer Probes were installed

Sniffer Probe	**Description**
Lab-1	Provides a baseline for Ubuntu OS fingerprint
Lab-2	Provides a baseline for Windows 10 fingerprint
Lab-A	DHT node, videostream client. SSH, and file server.
Lab-B	Desktop used light web browsing, music streamming.
Lab-C	Computer used for heavy browsing and streamming services.
Lab-D	TOR node and I2P router

Sniffer probes insert each collected packet information in the database that is, in this case, just a storage abstraction that the only elemental function is storing and querying of its contents.

As Figure 4.1 shows, for a collection of a month, probes collected a total of 184 million packets.

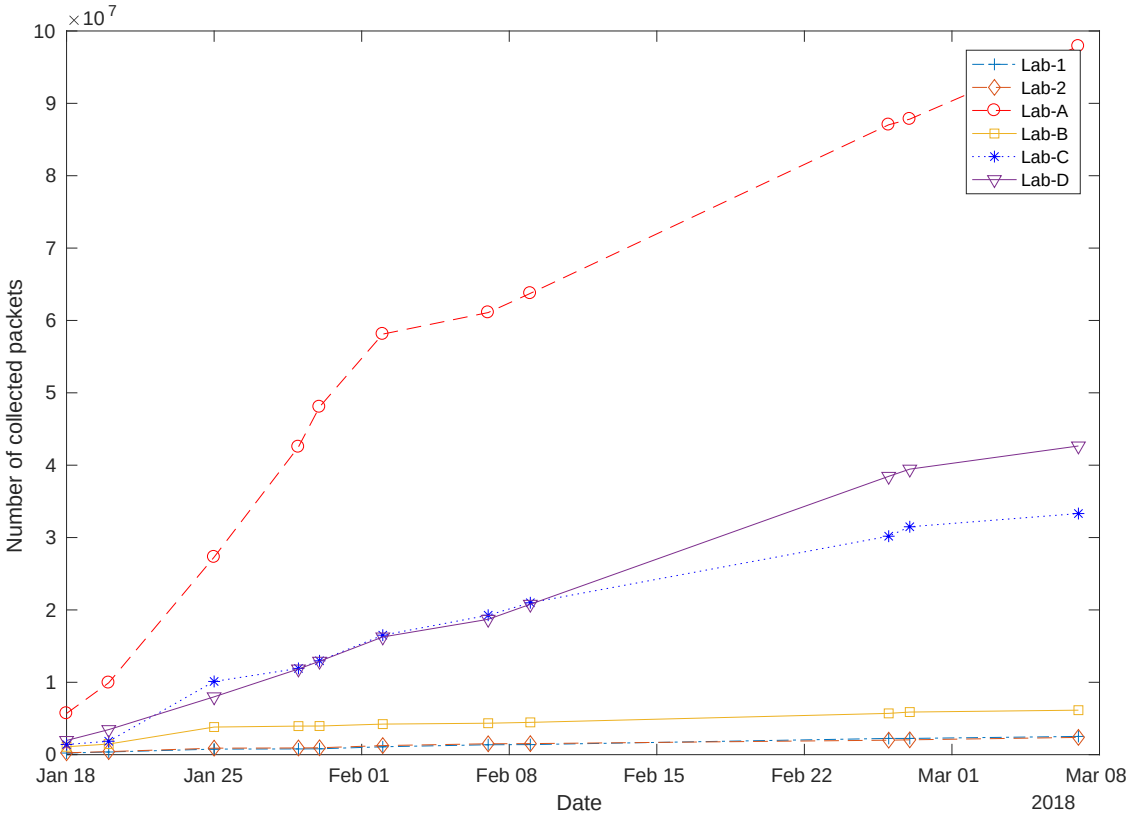

Figure 4.1: Total number of collected packets of each Sniffer Probe

It is also visible that different probes collect packets at different rates, that *per se* is a distinguishing pattern factor. Only *Lab-1* and *Lab-2* have a similar packet rate. This kind of different behavior is afterward converted into valuable features that represent a pattern, as described in Section 3.2.1.

4.2 Dataset composition

For window slices of a size of 15 minutes, generated data set has 55k observations, which is relatively acceptable for a ML classification project[1].

As Section 3.2 introduced, a parser processed the packets to model the inherent patterns. Table 4.2 shows the extracted features over the original captured data.

[1]The absolute minimum is 10k samples, according to some recommendations

Table 4.2: Low level view and description of the metadata within each *Counters* block

Counter	Description (sum in the slice)
packet_count	Number of collected packets
ip_external	Number of external IPs contacted
ip_internal	Number of internal IPs contacted
port_high	Number of contacted ports higher than 1024
port_low	Number of contacted ports lower or equal to 1024
tcp_syn	Number of packets with TCP SYN flag
tcp_fin	Number of packets with TCP FIN flag
tcp_rst	Number of packets with TCP RST flag
volume_download	Volume in bytes of downloaded data
volume_upload	Volume in bytes of uploaded data
volume_internal	Volume in bytes in the internal network
less_64 bytes	Number of packets with less than 64 bytes

For every packet is calculated the packet direction and if its nature is internal or external, filling some of the attributes. The requirement is knowing the public IPs that belong to the public servers and, at this level, the limitation of not inferring traffic direction between two internal entities. Doing so would require tracking at the entity level instead of at probe-level which process a network event. The reason of each is the following:

- Packet count

  The number of the packets exchanged in each *Timebar* slice allows the differentiation between periodic automated communications, the randomness of network traffic, and the detection of silence periods.
- IP external

  The number of external IPs contacted allows, among others, distinguishing an entity that acts as a public server from another entity that is intended to work only for internal network clients.
- IP internal

  Same as IP external, but in this case allows to detect if an entity is supposed to be internal and upon an abnormal activity if it starts contacting external peers, detecting the entity exposure to the outside network.
- Port High and Port Low

  Servers usually run using low port numbers (<=1024) and generate data from this ports. On the other hand, clients typically do not upload much data using high port numbers that are often used by malicious activity due to the fact of not requiring administrator privileges to listen on these ports.
- TCP SYN, FIN, and RST flags

  The ratio between TCP flags may indicate unusual network traffic or network disruption. The unbalanced counting of flags is an indicator of a line disruption or some entity generating incorrect traffic.
- Download, upload, and internal volume

Servers generate a considerable volume of upload data. Endpoints generate more download than upload and also generate a pattern of internal:external volume ratios. For example, a recently infected machine with ransomware generates an unusual volume of internal data if starts to encrypt all documents of a company NAS, which in turn is expressed in a change of the pattern of the internal volume.

- Count of packets with less than 64 bytes

  Usually, packets with less than 64 bytes are TCP control ACKs, which combined with the packet count and volume can indicate network entropy and helps to infer network disruption.

This calculation alone generates 48 features. Another features taken into account and not directly linked to the network statistical expression are the following:

- Is Weekday

  If the training sample is a weekday or weekend since working patterns usually change during these days.
- Is holiday

  A holiday can cause a weekday high variance, and if this information is not fed into pattern recognition, it will look like as an abnormality.
- Relative day position

  The relative position during the day helps to infer and distinguish working hours, breaks and nighttime of overall activity. It is directly related to the window position since it moves incrementally over the *Timebar Counters*.
- Silent periods variance and median

  The number of silent periods median and variance inside a window serves to differentiate entities that tend to have a higher variance. Namely, in the case of a machine to machine communication or a server network activity, silent periods tend to have lower variance (e.g., service timmers). In the case of human activity, opening a webpage generates an occurrence followed by an extended period of silence, being its variance much higher.
- Wavelet data

  Wavelets are used to detect periodic events of different periodicity. Inferring the periodicity of events works both for machine-to-machine communication (finding different periodic events) or just the information of no periodic events found, meaning a more random nature of occurrences plausibly caused by human behavior. Human behavior also trends to have periodic events in the range of minutes while machine in the order of seconds.

The complete set of features accounts for 63 dimensions that work as an input for the ML algorithms. Table 4.3 snapshot shows a view of the dataset with a few observations.

Table 4.3: View of a random sample from the final data set, resulted from sliding windows. Each row is an observation and columns are observation features.

	entity	packet__count:mean	packet__count:skew	...	tcp__fin:var	...	wavelet__9
1274	Lab-5	0.000 000	0.000 000	...	0.000 000	...	0.000 000
1311	Lab-A	0.000 000	0.000 000	...	0.000 000	...	0.000 000
1725	Lab-2	2.449 444	0.545 833	...	0.107 222	...	0.000 548
2163	Lab-3	0.000 000	0.000 000	...	0.000 000	...	0.000 000
1122	Lab-4	0.097 222	0.018 889	...	0.050 278	...	0.000 721
1271	Lab-4	0.000 000	0.000 000	...	0.000 000	...	0.000 000
518	Lab-2	1.270 000	0.283 056	...	0.014 722	...	0.000 726
396	Lab-1	6.860 278	0.372 778	...	3.638 889	...	0.001 262
1580	Lab-2	11.471 111	5.508 056	...	3.653 611	...	0.000 785
2	Lab-B	0.000 000	0.000 000	...	0.000 000	...	0.000 000

Just looking at the data without any criterion does not provide any useful information but gives a perception of data shape. In the case of the suggested network modeling the number of the feature columns is fixed at 64, including the label as explained in Section 3.2.1, and all the attributes are numerical, except for the label that must be later encoded to numerical. For a more informative view, Table 4.4 gives a better glimpse of the dataset.

Table 4.4: Analysis of mean, standard derivation, percentiles, minimum and maximum of the observations features.

	packet__count:mean	packet__count:skew	...	ip__external:mean	...	wavelet__9
count	13 580.000 000	13 580.000 000	...	13 580.000 000	...	13 580.000 000
mean	8.653 799	1.592 699	...	3.556 813	...	0.000 670
std	14.158 156	2.526 326	...	8.084 591	...	0.000 518
min	0.000 000	0.000 000	...	0.000 000	...	0.000 000
25%	0.000 000	0.000 000	...	0.000 000	...	0.000 000
50%	1.863 611	0.413 333	...	0.060 833	...	0.000 750
75%	12.596 181	1.699 514	...	2.590 625	...	0.000 971
max	122.005 000	9.651 111	...	70.400 833	...	0.002 907

The count, mean, min, and max rows are self-explanatory, and since the missing data is translated into "entity without activity" there are no null values. As this values are for the whole data, huge variations are expected. The 25%, 50%, and 75% rows show the corresponding percentiles that indicate the value below which a given percentage of observations of a group falls. For example, Table 4.5 shows an example that is more accurate,which are the values of machine *Lab-E* :

Table 4.5: Comparison of data statistics of Lab-E and and how it differs from the global data set

	packet__count:mean	ip__external:mean	...	wavelet__8	wavelet__9
count	2310.000 000	2310.000 000	...	2310.000 000	2310.000 000
mean	25.575 931	14.542 244	...	0.000 843	0.000 866
std	18.994 624	12.301 817	...	0.000 338	0.000 409
min	0.000 000	0.000 000	...	0.000 000	0.000 000
25%	10.750 417	4.910 069	...	0.000 764	0.000 698
50%	19.669 722	9.883 194	...	0.000 898	0.000 877
75%	44.103 472	25.162 222	...	0.001 033	0.001 066
max	58.275 000	45.457 500	...	0.001 731	0.002 326

The simple statistical description gives a fast way to of the type of data, for example, 25% of *packet count mean* are lower than 10 while 50% are lower than 20. These are values for a 1-hour window slice, which explains why the standard deviation is almost as the mean: for example, a small download within a window could increase the packet count and therefore generate a high variance and standard deviation.

Figure 3.7 of the previous Chapter has shown the histogram for a server machine. In comparison, histograms of Figure 4.2 from *Lab-1* show a different outlook. This entity is idling most of the time, and it reflects in many of its features. Histogram show such a difference that its pattern is visible both on the scale and the number of the occurrences in its scale. These two entities are significantly different from each other, being this is a good indicator that the entities are linearly separable.

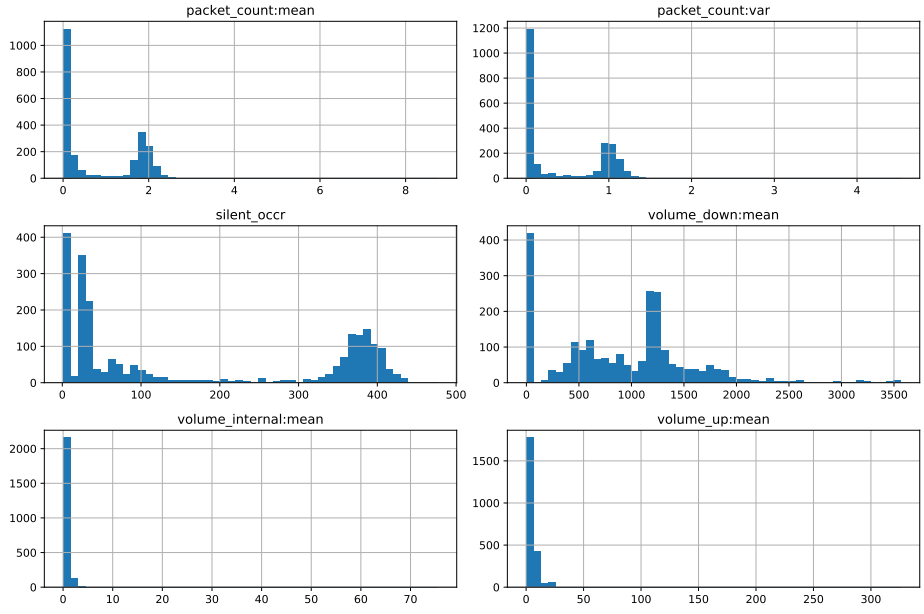

Figure 4.2: Histogram of some of the features from Lab-1 observations.

It is also possible to observe for this entity the silence occurrences are much more dispersed and frequent, meaning that has long and periods of inactivity, which matches its idling nature. These two histogram comparisons are distinguishable just at a naked eye and should represent

an explicit pattern recognition for ML classification algorithms.

An additional, more concrete perspective of the overall data set is the PCA analysis, as discussed in Section 3.3.4. In the case of this dataset, Figure 4.3 plots the variance ratio versus the number of features.

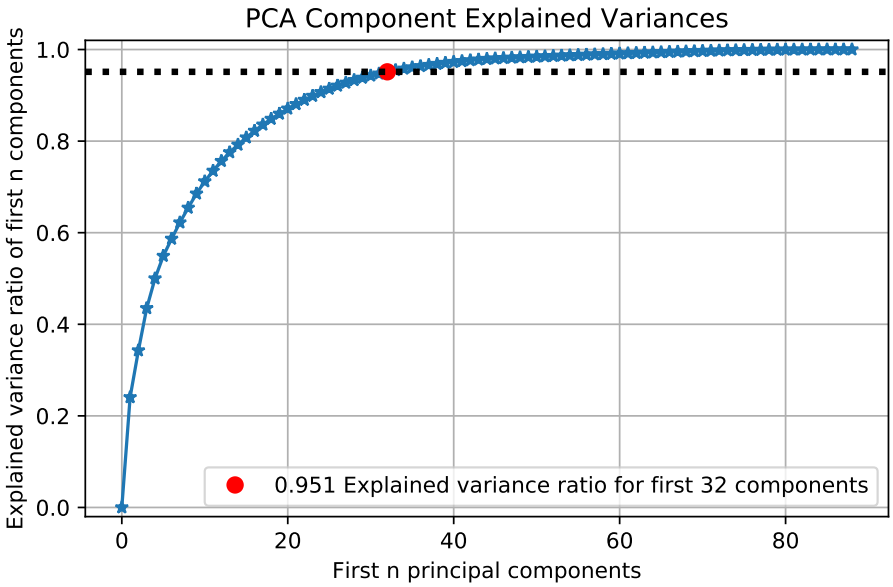

Figure 4.3: Total kept variance ratio of features over PCA reduction interactions. The less of the number of the features that the data set is reduced to, the less variance ratio and less expression of the original model.

The ideal number of features for *explained variance ratio* of 95% is 32 features. This means that 95% of the data set variance lies within a reduction into one-third of the features, and probably that the chosen model has many linear correlations for the entities of this dataset, as seen in Figure 3.8.

PCA reveals that the model suggested in Section 3.2.3 and 3.2.5 has room for improvement for more specific use cases and not so generic classification.

4.3 Classification Evaluation

In Section 3.5.1.1 one of the preliminary benchmarks on the model is to perform simple classification based on medians and PDF. If the entities are distinguishable enough, then it reflects on the accuracy. Taking averages into account alone archives 65% correct classifications[2].

A more delicate procedure is to take into account the distribution of feature values. Predictably, even though if this is a relatively simple computation over data, it improves the correct classification rate from feature mean centroids from 65% to 70% when using multi-variable distributions.

[2]Implementation available at the link in the Introduction Chapter

4.3.1 SVM classification

In ML field, Section 3.5.1.2 suggests as the first method using SVM classification. The measurement to determine the effectiveness of the system is accuracy, as explained by formula 3.2 in the same Section. As multi-label classification, Scikit-Learn's *LinearSVC* implementation for multi-class classification employs OvR strategy, meaning that trains the same number of classes as the labels [163]. Regarding parameter tuning, Table 4.6 shows the results of tests ran again different C parameters, in steps of power of 10s with the respective cross-validation folds[89].

Table 4.6: Impact o C parameter in the overall SVM classification performance

Rank	C	Accuracy
1	1	0.994 ± 0.010
2	0.1	0.992 ± 0.001
3	100000	0.991 ± 0.005
4	10000	0.990 ± 0.005
5	1000	0.990 ± 0.006

The classification works so well that the impact of tuning parameter C is practically indistinguishable. To understand why, Figure 4.4 shows different samples of C values for *Lab-1* entity:

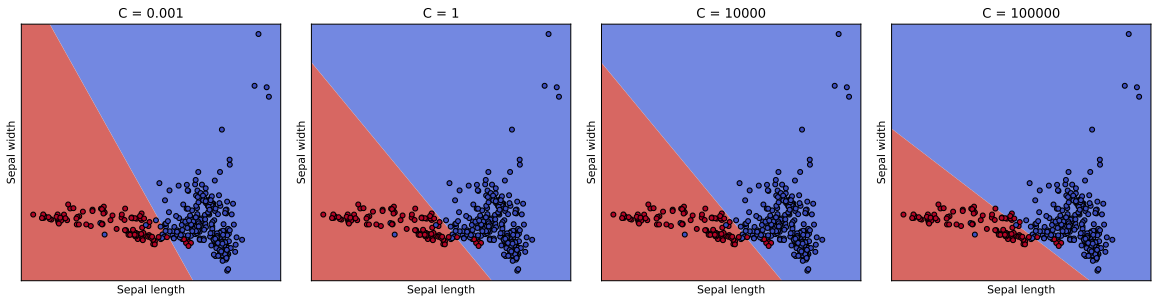

Figure 4.4: Visualization of the impact of SVM's C parameter separation margin and the tolerance of the outliers in such spearation

The plot justifies why the accuracy is not affected by the margins. The model makes this entity linearly separable and adjusting it barely affects the frontier of separation. Plotting the data is only possible due to the reduction of features to 2 dimensions using PCA, which means that despite the fact some of the labels do not look separable at two dimensions, they become more and more distinguishable as the number of dimensions.

Understanding the variations of tuning SVM ML algorithm is about examining the decisions across different labels. Figure 4.5 shows the confusion matrix (discussed in Section 3.5.1.2) for the proof of concept data set.

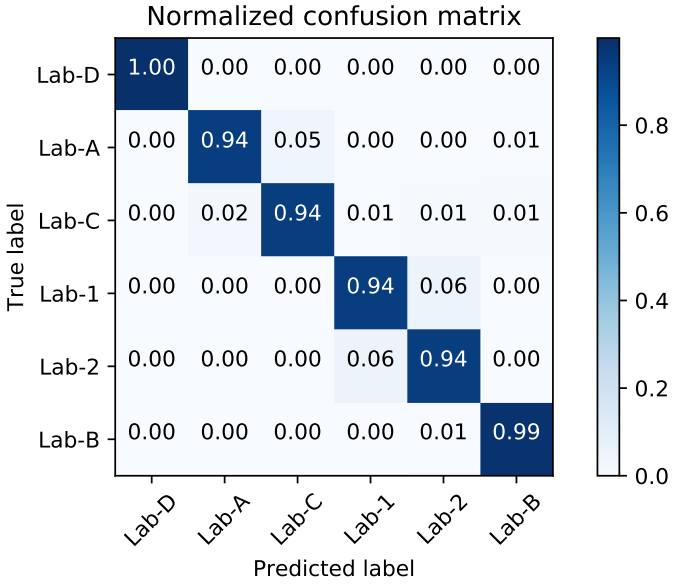

Figure 4.5: Normalized confusion matrix shows the miss classifications of the True label to the Predicted label for the SVM algorithm.

Variations of the diagonal are caused by variation in the number of samples for each of the probes. There is also a higher number of cases that Lab-1 is predicted as Lab-2 and vice-versa. Analyzing the error is easier by comparing the error rates instead. To do so, each value is divided by the number of entities in each class, then normalizing by dividing each cell by the corresponding sum, resulting in the Figure 4.6:

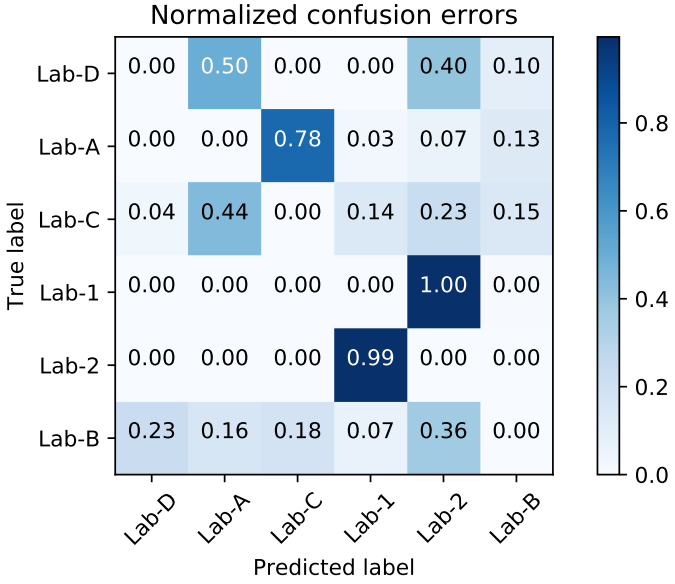

Figure 4.6: Normalized errors of confusion matrix of the SVM classification. The number of times a label as miss classified as another label.

Darker color means a higher misinterpretation of the proper probe. In the case of Lab-1

and Lab-2, this is understandable. These entities generate low traffic that is almost identical to each other. Also, the column and Lab-2 is also darker than any other, meaning that the system wrongly classifies any of the other probes with these two. Reasons for this are the silent periods (periods of time which the probes are not collecting data) confused with low network activity pattern of the fingerprint probes. Plus, low traffic from all use cases may be confused with the profile of an always-low traffic pattern. SVM decision boundaries can also help to visualize how the data looks. For such representation, features must are reduced via PCA to two for representation in a 2D plot, with the limitation of such reduction leading to classification underperform and losing the distinction between of probes, but good enough for representation. Figure 4.7 shows decision for worst case Lab-1 vs Lab-2.

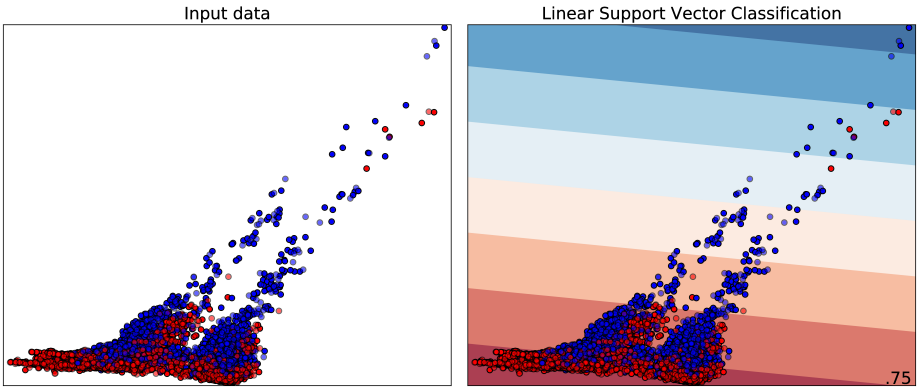

Figure 4.7: SVM decision boundaries for the worst classification case. The model generates feature observations that too identical and hard to distinguish at two dimensions

The data is barely linearly separable, and the algorithm tries to optimize the classification as the boundaries show. This means that for Lab-1 and Lab-2 the features are not enough to differentiate the entities and that their behavior is too identical. Figure 4.8 shows the same case but at three dimensions, suggesting that despite these two entities having identical patterns the separation relies on the dimensions rather than on the quality of the model expressing their differences.

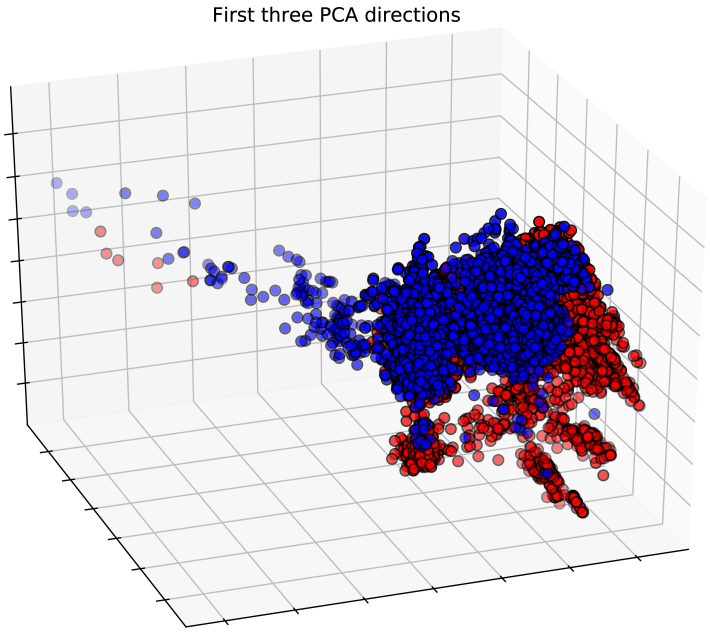

Figure 4.8: SVM decision boundaries for the worst classification case. The model generates feature observations that too identical and hard to distinguish at two dimensions

The accuracy results suggest that system has enough features to linearly classify the entities without tunning and that the collected data is sufficiently distinct enough to perform this separation. This means that it is possible to perform PCA reduction and therefore reduce the noise. Table 4.7 shows the results of C calibration combined with PCA reduction.

Table 4.7: Impact of PCA feature reduction combined with C parameter in the overall SVM classification performance

Rank	C	PCA reduction	Accuracy
1	1	83	0.994 ± 0.001
2	100000	79	0.993 ± 0.001
3	1	79	0.993 ± 0.001
4	1000	79	0.992 ± 0.001
5	10000	79	0.992 ± 0.001
...	...	...	...
9	1	59	0.991 ± 0.001

The SVM algorithm continues to perform exceptionally, independent to parameter calibration and PCA reduction. However, it deteriorates directly related to PCA. This may lead to a few conclusions: firstly that despite some feature redundancy, all groups (discussed in Section 3.3) of features at some point are significantly weighted into a decision that distinguishes the entities. Secondly, the model has room for improvement when it comes having factors that describe the entity behavior against other activities. Moreover, lastly, combination of per group PCA reduction may lead to a more profound conclusion about the relevance of each

feature group.

Training with a limited set of features and testing its accuracy gives a quick vision of what is relevant for network pattern classification, as Table 4.8 shows.

Table 4.8: Relationship of the features groups and its impact in classifier accuracy. The influence of each group hints on the models quality discrepancy of the patterns of the entities.

Group kept	Features	Accuracy
2+3	16	0.67 ± 0.02
1+3	59	0.96 ± 0.21
1+2	54	0.96 ± 0.31
3	11	0.39 ± 0.03
2	6	0.51 ± 0.09
1	49	0.96 ± 0.17

In the case of Group 3 indicates that this feature group probably needs a second layer of processing before its usage as a class feature. Network statistics seems to feed almost all the needed information to classify and distinguish each entity from all the others and suggests that there is room for improvement of dimensionality reduction and therefore noise removal and increased performance. The high accuracy of the results when only using Group 1 suggests that the classification is service-wise, meaning, that it is not distinguishing the human behavior but the services that the user is using. Volume, packet count and so forth describe the services running and the entities being classified use such different services that it is enough to successful classification without the human modeling element features.

4.3.2 NN classification

In comparison with SVM, the same methods of training and testing were performed using NN. As a result, the algorithm accuracy is so high (99.3% ± 0.01) that varying the parameters only introduces a small noise of ± 0.1% of the accuracy. Consequently, the moreover than studying the calibration of parameters of the NN algorithm (as suggested in Section 3.5.1.4) it is more interesting analyze the particular cases and to see how the system behaves in such conditions.

As discussed in Section 3.5.1.3, Figure 4.9 shows the normalized confusion matrix for errors, for the NN classifier:

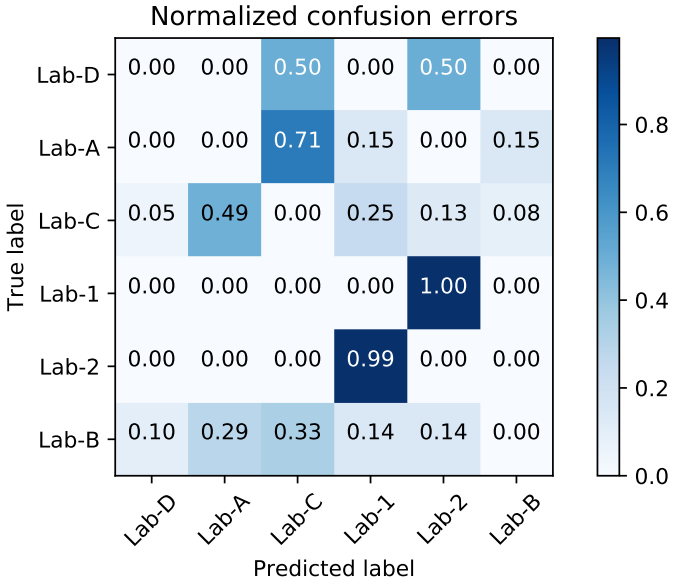

Figure 4.9: Normalized confusion matrix of the Neural Network classification errors. The number of times a label as miss classified as another label

Not surprisingly, likewise to SVM of Section before, continues is still the pair *Lab-1* and *Lab-2* and the errors tend to spawn along the same entities. Analysing the α hyperparameter, as Figure 4.10 shows, gives some insights why. The first row shows the worse case scenario and the second row shows the best case.

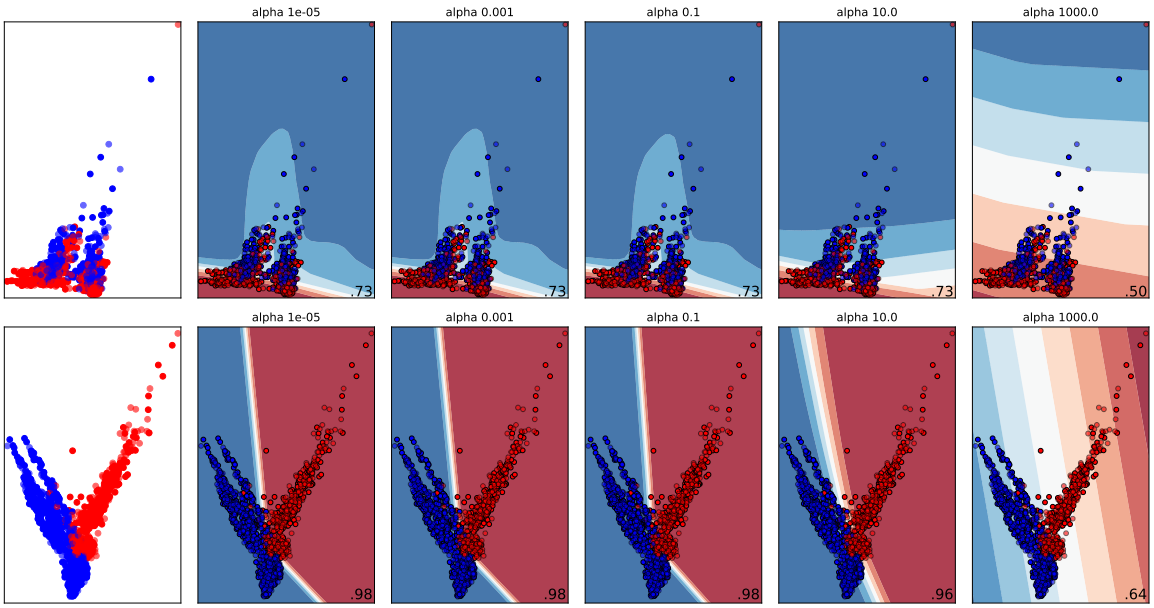

Figure 4.10: Impact of the α Neural Network parameter in the decision boundaries for the worst case (first row) and the worst case (second row)

For higher α values the algorithm is practically identical to SVM. Works as regularization and helps to avoid overfitting, by penalizing weights with large magnitudes[164]. As discussed,

NN fits the need of large-scale data sets, improving the classification response as the system will learn the new patterns incrementally without retraining using the whole training data.

4.3.3 Ensemble classification

In the Sections before, although both of the ML algorithms had a solid overall accuracy, most of the errors of the classification are for both on subgroups of the entities *Lab-1* and *Lab-2*. When comparing these two entities with the other four, the traffic pattern is too peculiar. These mixed data type, just like it is a subgroup within the whole data set group makes the Ensemble method a likely solution when mixing network patterns that are too different between each other. Where other ML algorithms are prone to errors, being the solution adjusting the features and algorithm parameters, AdaBoost self-corrects and recognizes this subtle differences. Figure 4.11 shows the number of iterations needed to detect these subgroups.

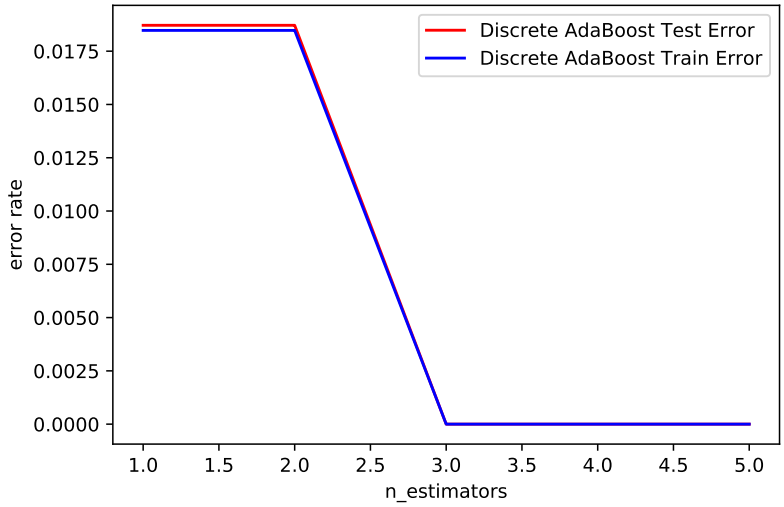

Figure 4.11: Error rate over the required number of estimators (classifiers) in order to archive perfect classification rate

In just four chain SVM estimators the system perfectly fits the learning data (which is tested with cross-validation), even to the particular subcases/groups as Lab-1 and Lab-2 that contrast with the rest of the set. This means that despite AdaBoost meta-classificator being prone to overfitting, it adapts perfectly to varied types of data such as this one. Identifying the entities are running on a network is one metric of knowing if the current status is according to its normal state. The next, as discussed in the next section, is to evaluate if the entity is behaving according to its conventional pattern.

4.4 Evaluating Anomaly detection

As discussed in 3.4.3, the pipeline of anomaly detection works a little differently of the classification pipeline tested in the Section above. In this test, the pipeline tries to classify

each entity individually and classifying its pattern independently of all other entities, if it is acting according to know history and patterns.

Table 4.9 detection rate for each entity, namely, each entity has its own ML algorithm running and comparing new data against old data and outputting if the new data resembles to known patterns.

Table 4.9: Anomaly detection accuracy rate for independent entity SVM classification

Entity	Accuracy
Lab-1	0.98 ± 0.06
Lab-2	0.77 ± 0.05
Lab-A	0.93 ± 0.05
Lab-B	0.84 ± 0.37
Lab-C	0.98 ± 0.04
Lab-D	0.86 ± 0.28

In the case of binary classification if an entity is acting or not anomaly, the ROC curve for SVM shows that the true positive rate elbow is around 0.3 false positive rate - ROC curve of class one and matches the approximate abnormality detection of 0.84 ± 0.0552 of the results shown in Table 4.9.

In the other hand, Figure 4.12 shows ROC curve when using NN, with the elbow for the optimal true positive rate with lower than 0.2 false positive rate.

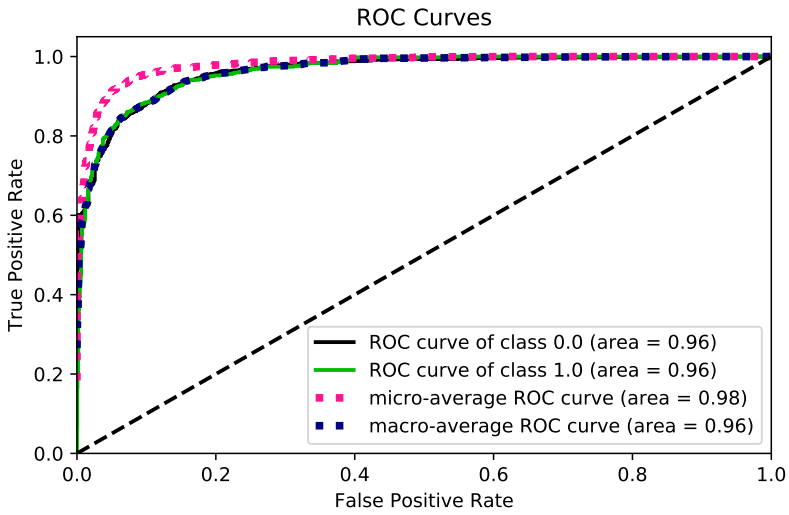

Figure 4.12: Precision-Recall Curve for binary Neural Network classification, showing a smoother compromise between True Positive Rate and the False Positive Rate.

This means that NN offer more flexibility between the compromise of detecting abnormalities and keeping a few false positives, or the other way around: no false positives high an more and less detection rate of 50%. The elbow is also smoother, meaning that NN classification has finer edges in its classification, probably because of SVM linearly separates the observations and NN uses a more flexible decision function.

In the case of binary classification if an entity is acting anomaly or not, the ROC curve for SVM shows that the true positive rate elbow is around 0.3 false positive rate - ROC curve of class one and matches the approximate abnormality detection of 0.84 ± 0.05 of the results shown in Table 4.9.

Table 4.10: Confusion matrix for SVM binary classification

Classified as →	$\bar{X}$	X
$\bar{X}$	24698 (True Negative)	5871 (False Positive)
X	494 (False Negative)	7238 (True Positive)

The first row is the negative class, which considers non-Lab-2 entities. *24698* of the samples were correctly classified as not belonging to Lab-2 traffic pattern - true negatives, while the remaining *5781* were wrongly classified as the Lab-2 pattern - a false positive. In the second row, the positive class, *494* samples were wrongly classified as non-Lab2 traffic pattern - false negative, while the remaining *7232* traffic samples were correctly classified as the Lab-2 pattern - true positive. The accuracy of the positive predictions is the classifier *precision*, as 3.2 shows.

Figure 4.13 shows this tradeoff for the worst case scenario of *Lab-2*.

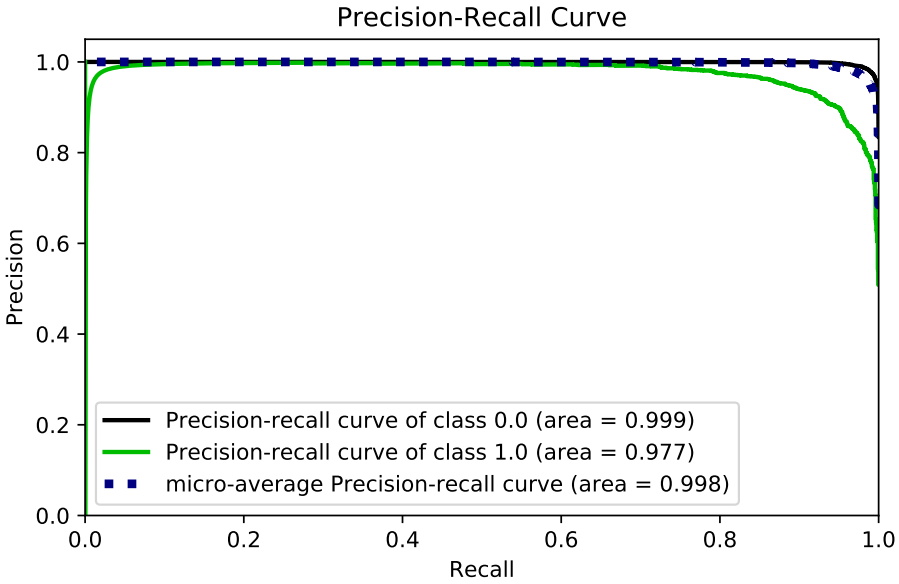

Figure 4.13: Precision-Recall Curve for binary SVM classification with a sharp elbow arround the 95% of precision

The curve shows that for a precision of 99% the recall is 60%. This means that for a system with high precision the lowest recall possible is around 60%.

CHAPTER 5

Conclusions and future work

In regards to the work developed in this dissertation of internet security and its relationship with the network, with the increase of awareness for privacy topics, it is safe to expect a strengthening of this association. As discussed in Chapter 2, being privacy a new hot topic and fueled by new regulations, it is increasingly gaining the interest and research investment. New tools that improve misuse of systems and detect data leakage are emerging, being network activity one of the primary heuristics for such detection. Inferring network security state is knowing who or what is operating within the network is merely the first layer of awareness. Perceiving the state of each entity boarders the security to the entity level increases precision and distinction of what is not running according to the expected. Identifying the network threat and its origin not only improves the audit process but also grants a whole new level of potential actions to mitigate an attack, such as automatic response or entity isolation.

Considering network data collection of Section 3.1, despite not focusing neither on the efficiency of the placement of network sniffers, the work developed analyzed a possible approach of retrieving the required peer interaction communications'information. Being generic allows future adaptation to each use case: the position, processing, and level of probe infer capabilities can be tweaked and adjusted per the use case. Collecting all packets information and storing them serve the only purpose of studying the methods of inferring network patterns. Iterating over such data set may, in some cases, become excessively inefficient, especially in the case of entities generating packets at a high frequency. However and because only six were classified, using document-orient storage allowed prompt experimentation without the hassle of leading with more scalable architecture like complex specific hardware or software solutions. Due to the limitations of resources during this dissertation, there was not possible to study more network-generating entities patterns, even if was highly desirable. Ideally, the entities and more sniffers would collect groups of similar patterns with closer patterns. It would be highly propitious to have a group of servers (or if possible servers with a specific purpose like HTTP or file server), a group of computers of employees that perform identical computer work, and a group of IoT devices, such as SmartTVs, weather stations, home surveillance cameras, and

so on. These use cases would be much beneficial and inline what discussed in Section 2.1.1. In fact (and unfortunately) the sniffed entities Lab-A to D (Table 4.1) had probably an overly distinct pattern, which may justify the remarkably good performance of ML classification algorithms, which contrasted with too identical and scarce network data from entities Lab-1 and Lab-2. Despite this and once again, Section 3.5 shows that algorithm tunning has vast room for improvement to face the challenges of having more machines with thinner pattern distinctions.

On Feature Engineering level discussed in Section 3.2, the collection of pure statistical data (as shown in Table 4.2) from the network proved to be more than enough to distinguish these entities. However, some literature suggested direct inference at the feature generation level. A straightforward improvement and purely related to the network is adding protocol-related information :

- Generate feature attributes from network streams (e.g., TCP sessions), by analyzing the ratios of TCP flags within a session
- Flow durations
- Ratios of TCP sessions re-connections, misplaced TCP timestamps and windows sizes, and so on
- Frequency of ICMP packet and ratios with its volumes and codes
- Correlate volume to ports, to IPs, and so on

Other suggestions include adding metrics for detecting traffic bursts and their duration and many other metrics in the counting, means, symmetry, defining thresholds, and so on [165], [166]. Any of each requires a balance of overhead of computing features and the real-time classification that in some cases may be too resource intensive or requiring more collection time. In the case of distinguishing humans and machines, and distinguishing between different humans and different machines, remained to be tested within this premises. As discussed in Sections 3.2.3 and 3.2.5 in which modeling of human and machine behavior falls back on frequency analysis, the next level would be the automatic machine and human modeling classification. This means that is possible to classify events frequency recurring to wavelets within a time window (as discussed in Section 3.2.2) and then use its frequency to infer human and machine behavior: humans tend to have frequency at a much larger and infrequent scale (from seconds to a few minutes) when machines tend to have a more timely more frequently pattern. This classification alone can be a whole new system that feeds the current classification system - a whole new field of study of multiple ML systems feeding information in each other and working combined.

From this perspective, combinations of all the types of features and their generation is by itself a whole domain, in which is, in fact, supported by the analysis in Section 3.3.4, that shown that network statistics proven by them self to provide enough distinct factors to classify the entities of this dissertation. This may also suggest that despite the fact that Lab-B and Lab-C are operated by humans, the human factor barely weights on the pattern distinction. Is at the machine-level used by the employees that create the pattern, when using services (like YouTube, Spotify, Skype, etc.) or the activity being mainly created by programs and

applications instead of being a result from the direct human action. Plus, the relatively high error rate between Lab-1 and Lab-2 asserts the idea of that feature engineering needs more research to combine metrics that distinguish these similar patterns.

The two types of ML pipelining architecture in section 3.4, wherein all from network data up to ML classification results into two possible outcomes, showing that same data is seen in various views with different commitments. Entity classification requires training on every change but identifies the network players are operating and works as a first level of NSA where a decision is performed by the system automatically. A more in-depth level, NSSA (as discussed in Sections 2.5.2 and 2.5.3) is therefore provided to independently entity view of anomaly detection of Section 3.4.3, that inline of knowing what entities are operating sums the information of which one is performing on its standard behavior pattern or not.

Evaluation of the process conducted in Section 3.5 indicates, at best, the compromise of the granularity of algorithm control. Statistical analysis demonstrated in Section 3.5.1.1 is way more transparent process. The advantage of manual classification implementation gives total control over the internals of the decision steps, how the process is made, and how the pure statistical expression of dataset provides pattern recognition. SVM and NN are harder to understand their decision process, that in turn are counterbalanced with visualization and performance tools evaluations of Sections 3.5.1.3, and 4.3.2. Ensemble hybrid ML algorithms like AdaBoost of Section 3.5.1.5 insinuate the ability of layering decisions to solve the case of disparity of group entities such as the groups of Lab-1+Lab-2 and Lab-A to D. Using the same metrics to evaluate the classification process (like the confusion matrix) across different algorithms, supports the idea that NN algorithms offer the scalability over SVM. For abnormality detection, the ROC analysis allows a higher utility of such system. It can work as merely an indicator of many false positives or with significant weight in the current security situation decision, by signaling an abnormality only when something dubious is (almost) for sure happening.

All in all, Chapter 3 shown that it is possible to correlate network activity to an entity, and therefore enhancing the awareness of the systems operating in a network. Raw data is converted into information, which in turn converted into knowledge and awareness. In light of this, data set testing lacks better evaluation with real-world tests (e.g., ransomware operating over the network), even though of ML being remarkably good at inferring and learning from known data and extrapolating learned observations to future ones. At an enterprise level, the scalability of NN algorithms permits recognition without compromising the classification rate making it possible to escalate to large networks, keeping relatively fast response times.

The examination conducted is, *de facto*, in line with the suggestions of last years trends of suggesting the research of the usage of ML based anomaly detection [147] and cover some flaws of latest studies in this domain. Criticism of academic work includes the overuse of DARPA dataset, the omission of outlier detection, too high error rates and unadaptability of ML to network context [167]. All those subjects covered in this dissertation. Despite the small set of lab machines, the developed pipeline in the approached studied, they altogether suggest the possibility of filling these gaps.

However, there are some obvious challenges that may render this kind of system unusable. The high variability of network traffic due to too many reasons extends the network activity model beyond to network statistics. For example, weekdays, viral events or coordinated actions, they all cause a high variance that may render the system ineffective unless it can model all these variations. Having said that, the domain of ML for abnormality, and despite the current good heuristics, is too extended and remote for archiving full complete NSSA without human interpretation. Yet.

References

[1] S. S. IBM Global Technology Services, "IBM X-Force Threat Intelligence Report 2016", *IBM Security Managing Security Services*, no. February, pp. 1–24, 2016. [Online]. Available: http://www.foerderland.de/fileadmin/pdf/IBM%7B%5C_%7DXForce%7B%5C_%7DReport%7B%5C_%7D2016.pdf.

[2] *What is the fourth industrial revolution? | World Economic Forum.* [Online]. Available: https://www.weforum.org/agenda/2016/01/what-is-the-fourth-industrial-revolution/ (visited on 11/07/2017).

[3] *What are the Panama Papers? A guide to history's biggest data leak | News | The Guardian.* [Online]. Available: https://www.theguardian.com/news/2016/apr/03/what-you-need-to-know-about-the-panama-papers (visited on 11/03/2017).

[4] *Panama Papers: What happened next? - BBC News.* [Online]. Available: http://www.bbc.com/news/world-38319026 (visited on 11/03/2017).

[5] *Clashes as Pakistani anti-government protesters bear down on capital.* [Online]. Available: http://in.reuters.com/article/pakistan-politics-protests/clashes-as-pakistani-anti-government-protesters-bear-down-on-capital-idINKBN12V0OE (visited on 11/03/2017).

[6] *Panama Papers: Protesters demand Cameron's resignation | News | Al Jazeera.* [Online]. Available: http://www.aljazeera.com/news/2016/04/panama-papers-protesters-demand-cameron-resignation-160409184433734.html (visited on 11/03/2017).

[7] *Panama Papers Have Had Historic Global Effects — and the Impacts Keep Coming · ICIJ.* [Online]. Available: https://panamapapers.icij.org/20161201-global-impact.html (visited on 11/03/2017).

[8] B. D. Lea, "Equifax could face largest class-action suit 'in US history'", *FOXBusiness*, Sep. 2017. [Online]. Available: http://www.foxbusiness.com/markets/2017/09/08/equifax-could-face-largest-class-action-suit-in-us-history.html.

[9] *The biggest data breaches ever - Sep. 7, 2017.* [Online]. Available: http://money.cnn.com/2017/09/07/technology/business/biggest-breaches-ever/index.html (visited on 10/03/2017).

[10] *How Equifax compares to biggest hacks of all time: CHART - Business Insider*, 2017. [Online]. Available: http://www.businessinsider.com/how-equifax-compares-to-biggest-hacks-of-all-time-chart-2017-9 (visited on 10/03/2017).

[11] *How the Equifax data breach happened: What we know now - Sep. 16, 2017.* [Online]. Available: http://money.cnn.com/2017/09/16/technology/equifax-breach-security-hole/index.html (visited on 10/21/2017).

[12] B. McKenna, "Symantec's Thompson pronounces old style IT security dead", *Network Security*, vol. 2005, no. 2, pp. 1–3, 2005, ISSN: 13534858. DOI: 10.1016/S1353-4858(05)00194-7. arXiv: z0024. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1353485805001947.

[13] A. Glitz and E. Meyersson, "Industrial Espionage and Productivity", pp. 25–27, 2017, ISSN: 18154654.

[14] Level3, *Level3 outage? Current problems and outages | Down Detector*, 2016. [Online]. Available: http://downdetector.com/status/level3/map/ (visited on 10/23/2016).

[15] *Mirai DDoS botnet powers up, infects Sierra Wireless gateways | ZDNet.* [Online]. Available: http://www.zdnet.com/article/mirai-ddos-botnet-powers-up-infects-sierra-wireless-gateways/ (visited on 10/03/2017).

[16] *Source code for Mirai IoT Malware was recently released.* [Online]. Available: https://www.webroot.com/blog/2016/10/10/source-code-mirai-iot-malware-released/ (visited on 10/03/2017).

[17] *Leaked Mirai Malware Boosts IoT Insecurity Threat Level.* [Online]. Available: https://securityintelligence.com/news/leaked-mirai-malware-boosts-iot-insecurity-threat-level/ (visited on 10/03/2017).

[18] *Breaking Down Mirai: An IoT DDoS Botnet Analysis.* [Online]. Available: https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html (visited on 10/03/2017).

[19] *Large DDoS attacks cause outages at Twitter, Spotify, and other sites | TechCrunch.* [Online]. Available: https://techcrunch.com/2016/10/21/many-sites-including-twitter-and-spotify-suffering-outage/ (visited on 10/21/2017).

[20] *Internet Outage: Homeland Security Looking Into All Causes | Time.com.* [Online]. Available: http://time.com/4540921/internet-dyn-outage-homeland-security/ (visited on 10/21/2017).

[21] H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, "Evading Machine Learning Malware Detection", *Black Hat*, 2017. [Online]. Available: https://www.blackhat.com/docs/us-17/thursday/us-17-Anderson-Bot-Vs-Bot-Evading-Machine-Learning-Malware-Detection-wp.pdf.

[22] *Government-backed attackers may be trying to steal your password - Gmail Help.* [Online]. Available: https://support.google.com/mail/answer/2591015?hl=en (visited on 11/22/2017).

[23] *Yahoo says half a billion accounts breached by nation-sponsored hackers | Ars Technica.* [Online]. Available: https://arstechnica.com/information-technology/2016/09/yahoo-says-half-a-billion-accounts-breached-by-nation-sponsored-hackers/ (visited on 11/22/2017).

[24] L. A. Kramer, R. J. Heuer Jr., and K. S. Crawford, "Technological, Social, and Economic Trends That Are Increasing U.S. Vulnerability to Insider Espionage", no. May, 2005. DOI: 10.1037/e454452006-001. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord%7B%5C&%7DmetadataPrefix=html%7B%5C&%7Didentifier=ADA433793.

[25] *Apple, Google, others settle antipoaching lawsuit for $415 million - CNET.* [Online]. Available: https://www.cnet.com/news/apple-google-others-settle-anti-poaching-lawsuit-for-415-million/ (visited on 11/07/2017).

[26] I. Winkler, "Case study of industrial espionage through social engineering", *Proceedings of the 19 th Information Systems Security Conference*, pp. 1–7, 1996. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.115%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[27] A. Crane, "In the company of spies: When competitive intelligence gathering becomes industrial espionage", *Business Horizons*, vol. 48, no. 3, pp. 233–240, 2005, ISSN: 00076813. DOI: 10.1016/j.bushor.2004.11.005.

[28] M. Chan, "Corporate Espionage and Workplace Trust/Distrust", *Journal of Business Ethics*, vol. 42, no. 1, pp. 45–58, 2003, ISSN: 01674544. DOI: 10.1023/A:1021611601240.

[29] I. W. Monitor, "Tracking GhostNet: Investigating a Cyber Espionage Network", *Think Tank White Paper*, no. March 29, pp. 1–53, 2009. [Online]. Available: www.infowar-monitor.net/ghostnet%7B%5C%%7D5Cnpapers://fcc2d7b3-74ea-47b3-9ccd-d3deae240bcc/Paper/p2431.

[30] D. Version, "Aalborg Universitet Of Social Engineers & Corporate Espionage Agents Yeboah-Boateng, Ezer Osei", 2013.

[31] S. Sheng, M. Holbrook, P. Kumaraguru, L. F. Cranor, and J. Downs, "Who falls for phish?", *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, p. 373, 2010, ISSN: 1605589292. DOI: 10.1145/1753326.1753383. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1753326.1753383.

[32] K. Jackson, *Building a secure computer system*, 8. 1989, vol. 11, pp. 18–19, ISBN: 0442230222. DOI: 10.1016/0142-0496(86)90071-8.

[33] B. Bashir and A. Khalique, "A Review on Security versus Ethics", *International Journal of Computer Applications*, vol. 151, no. 11, pp. 975–8887, 2016.

[34] T. Speed and J. Ellis, *Internet Security*. Wiley, 2003, pp. 169–197, ISBN: 9781555582982. DOI: 10.1016/B978-155558298-2/50009-9. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781555582982500099.

[35] H. Wahid, S. Ahmad, M. A. M. Nor, and M. A. Rashid, "Prestasi kecekapan pengurusan kewangan dan agihan zakat: perbandingan antara majlis agama islam negeri di Malaysia", Tech. Rep. 2, 2017, pp. 39–54. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3. [Online]. Available: http://register.consilium.europa.eu/doc/srv?l=EN%7B%5C&%7Df=ST%206225%202013%20INIT.

[36] European Commission, "EU Cybersecurity Initiatives", *Factsheet*, p. 8, 2017. [Online]. Available: http://ec.europa.eu/information%7B%5C_%7Dsociety/newsroom/image/document/2017-3/factsheet%7B%5C_%7Dcybersecurity%7B%5C_%7Dupdate%7B%5C_%7Djanuary%7B%5C_%7D2017%7B%5C_%7D41543.pdf.

[37] H. Wahid, S. Ahmad, M. A. M. Nor, and M. A. Rashid, *Prestasi kecekapan pengurusan kewangan dan agihan zakat: perbandingan antara majlis agama islam negeri di Malaysia*, 2017. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016L1148%7B%5C&%7Dfrom=EN.

[38] F. V.-p. Frans and R. Unit, "Security Union: Commission accelerates measures to prevent radicalisation and the cyber threat", no. June, pp. 3–4, 2017. [Online]. Available: http://europa.eu/rapid/press-release%7B%5C_%7DIP-17-1789%7B%5C_%7Den.htm.

[39] *Hackers post stolen HBO 'Game of Thrones' scripts online, demand bitcoin ransom - The Washington Post*, 2017. [Online]. Available: https://www.washingtonpost.com/news/morning-mix/wp/2017/08/08/hackers-post-stolen-hbo-game-of-thrones-scripts-online-demand-bitcoin-ransom/ (visited on 10/03/2017).

[40] *'GAME OF THRONES' FINALE SETS RATINGS RECORD - THE NEW YORK TIMES*. [Online]. Available: https://www.nytimes.com/2017/08/28/arts/television/game-of-thrones-finale-sets-ratings-record.html (visited on 10/03/2017).

[41] D. Kim, *Fundamentals of Information Systems Security*, 3rd Editio. Jones & Bartlett Learning, 2010, p. 526, ISBN: 978-0-7637-9025·7.

[42] *44 U.S. Code § 3542 - Definitions | US Law | LII / Legal Information Institute*. [Online]. Available: https://www.law.cornell.edu/uscode/text/44/3542%20https://www.law.cornell.edu/uscode/text/44/3542%7B%5C%%7D5Cnhttp://www.gpo.gov/fdsys/pkg/USCODE-2013-title44/pdf/USCODE-2013-title44-chap35-subchapIII-sec3542.pdf (visited on 10/03/2017).

[43] C. Serrano-Cinca, Y. Fuertes-Callén, and C. Mar-Molinero, "Measuring DEA efficiency in Internet companies", *Decision Support Systems*, vol. 38, no. 4, pp. 557–573, 2005, ISSN: 01679236. DOI: 10.1016/j.dss.2003.08.004. arXiv: 1011.1669v3.

[44] C. Zhang, X. Shen, X. Pei, and Y. Yao, "Applying Big Data Analytics into Network Security: Challenges, Techniques and Outlooks", in *Proceedings - 2016 IEEE International Conference on Smart Cloud, SmartCloud 2016*, 2016, pp. 325–329, ISBN: 9781509052622. DOI: 10.1109/SmartCloud.2016.62.

[45] *Publicly Available Standards*. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/ (visited on 10/10/2017).

[46] *ISO/IEC 27001:2013 - Information technology – Security techniques – Information security management systems – Requirements*. [Online]. Available: https://www.iso.org/standard/54534.html (visited on 10/10/2017).

[47] *ISO/IEC 27002 code of practice*. [Online]. Available: http://www.iso27001security.com/html/27002.html (visited on 10/10/2017).

[48] ISO, *Introduction To ISO 27002 (ISO27002)*, 2013. [Online]. Available: http://www.27000.org/iso-27002.htm (visited on 10/10/2017).

[49] T. UcedaVelez and M. M. Morana, *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. Wiley, 2015. [Online]. Available: https://www.amazon.com/Risk-Centric-Threat-Modeling-Simulation-ebook/dp/B00XN46KTU?SubscriptionId=0JYN1NVW651KCA56C102%

7B%5C&%7Dtag=techkie-20%7B%5C&%7DlinkCode=xm2%7B%5C&%7Dcamp=2025%7B%5C&%7Dcreative=165953%7B%5C&%7DcreativeASIN=B00XN46KTU.

[50] E. Attack and R. Software, "Process for Attack Simulation and Threat Analysis", no. October, 2014.

[51] S. Sahibudin, M. Sharifi, and M. Ayat, "Combining ITIL, COBIT and ISO/IEC 27002 in order to design a comprehensive IT framework in organizations", *Proceedings - 2nd Asia International Conference on Modelling and Simulation, AMS 2008*, pp. 749–753, 2008, ISSN: 2237-4558. DOI: 10.1109/AMS.2008.145.

[52] RackSpace, "A new world, a new security approach", p. 2, 2017. [Online]. Available: https://www.rackspace.com/resources/white-paper-our-rms-security-philosophy.

[53] *Active vs. Passive network monitoring: an infographic – Iris Network Systems.* [Online]. Available: https://www.irisns.com/active-vs-passive-network-monitoring-an-infographic/ (visited on 12/05/2017).

[54] H. T. Work, O. S. Types, S. Ratios, O. Speeds, P. Budgets, L. Loss, and B. Technology, "Understanding Network TAPs – The First Step to Visibility", *Gigamon*, pp. 1–9, 2017.

[55] W. You, "To TAP or SPAN?", *Gigamon*, pp. 1–4, 2017.

[56] S. C. Gordy, H. D. Poelstra, R. W. Otis, and T. Gallatin, *Network security tap for use with intrusion detection system*, 2005. [Online]. Available: https://www.google.com/patents/US6898632.

[57] *Sniffing Tutorial part 1 - Intercepting Network Traffic - NETRESEC Blog.* [Online]. Available: http://www.netresec.com/?page=Blog%7B%5C&%7Dmonth=2011-03%7B%5C&%7Dpost=Sniffing-Tutorial-part-1---Intercepting-Network-Traffic (visited on 11/15/2017).

[58] • *SDN/NFV DATA CENTER TRAFFIC WORLDWIDE 2015-2020 | STATISTIC.* [Online]. Available: https://www.statista.com/statistics/637966/worldwide-sdn-nfv-data-center-traffic/ (visited on 12/05/2017).

[59] C. Wang, O. Spatscheck, V. Gopalakrishnan, Y. Xu, and D. Applegate, "Toward High-Performance and Scalable Network Functions Virtualization", *IEEE Internet Computing*, vol. 20, no. 6, pp. 10–20, 2016, ISSN: 10897801. DOI: 10.1109/MIC.2016.111.

[60] *DPDK: Data Plane Development Kit – What it is.* [Online]. Available: http://dpdk.org/ (visited on 11/20/2017).

[61] *Intel DPDK, switch and server ref designs push SDN ecosystem forward.* [Online]. Available: http://searchsdn.techtarget.com/news/2240182264/Intel-DPDK-switch-and-server-ref-designs-push-SDN-ecosystem-forward (visited on 11/20/2017).

[62] J.-P. Navarro, B. Nickless, and L. Winkler, "Combining cisco netflow exports with relational database technology for usage statistics, intrusion detection, and network forensics", in *Proceedings of the 14th Large Installation Systems Administration Conference (LISA 2000)*, 2000, pp. 285–290.

[63] T. Margoni and M. Perry, "Legal Consequences of Packet Inspection", *SSRN Electronic Journal*, 2011, ISSN: 1556-5068. DOI: 10.2139/ssrn.2028981. [Online]. Available: http://www.ssrn.com/abstract=2028981.

[64] *NSA Prism program taps in to user data of Apple, Google and others | US news | The Guardian.* [Online]. Available: https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data (visited on 11/14/2017).

[65] *Encrypted Web Traffic More Than Doubles After NSA Revelations | WIRED.* [Online]. Available: https://www.wired.com/2014/05/sandvine-report/ (visited on 11/14/2017).

[66] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The Cost of the "S" in HTTPS", *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies - CoNEXT '14*, pp. 133–140, 2014. DOI: 10.1145/2674005.2674991. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2674005.2674991.

[67] *Telemetry/FAQ - MozillaWiki*. [Online]. Available: https://wiki.mozilla.org/Telemetry/FAQ%7B%5C#%7DTelemetry%7B%5C_%7Dand%7B%5C_%7DUser%7B%5C_%7DControl:%7B%5C_%7DFAQ (visited on 11/14/2017).

[68] *Google's SSL Update - What Does It Mean for Your Website?* [Online]. Available: https://www.theedesign.com/blog/2017/googles-ssl-update (visited on 11/14/2017).

[69] B. Miller, L. Huang, A. D. Joseph, and J. D. Tygar, "I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis", in *Privacy Enhancing Technologies: 14th International Symposium, PETS 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings*, E. De Cristofaro and S. J. Murdoch, Eds. Cham: Springer International Publishing, 2014, pp. 143–163, ISBN: 978-3-319-08506-7. DOI: 10.1007/978-3-319-08506-7_8. [Online]. Available: https://doi.org/10.1007/978-3-319-08506-7%7B%5C_%7D8.

[70] D. Schatzmann and W. Mühlbauer, "Digging into HTTPS : Flow-Based Classification of Webmail Traffic", *Proceedings of the 10th ...*, pp. 322–327, 2010. DOI: http://doi.acm.org/10.1145/1879141.1879184. [Online]. Available: http://dl.acm.org/citation.cfm?id=1879184.

[71] *Data Mining - MIT Technology Review*. [Online]. Available: http://www2.technologyreview.com/news/401766/data-mining/ (visited on 12/11/2017).

[72] M. Sharma, "Data Mining : A Literature Survey", *International Journal of Emerging Research in Management &Technology*, vol. 9359, no. 2, pp. 1–4, 2014.

[73] G. A. Marcoulides, *Discovering Knowledge in Data: an Introduction to Data Mining*, 472. 2005, vol. 100, pp. 1465–1465, ISBN: 6610275297. DOI: 10.1198/jasa.2005.s61. arXiv: arXiv:1011.1669v3. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1198/jasa.2005.s61.

[74] SQLPower, *What is Business Intelligence?*, 2017. [Online]. Available: https://business.tutsplus.com/tutorials/what-is-business-intelligence--cms-23412%20http://www.sqlpower.ca/xbrlpower/page/biprimer (visited on 12/12/2017).

[75] H. Wahid, S. Ahmad, M. A. M. Nor, and M. A. Rashid, "Prestasi kecekapan pengurusan kewangan dan agihan zakat: perbandingan antara majlis agama islam negeri di Malaysia", *Jurnal Ekonomi Malaysia*, vol. 51, no. 2, pp. 39–54, 2017, ISSN: 01261962. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[76] *Statistics vs Data Science vs BI (Revolutions)*. [Online]. Available: http://blog.revolutionanalytics.com/2013/05/statistics-vs-data-science-vs-bi.html (visited on 12/12/2017).

[77] U. von Luxburg, "Clustering Stability: An Overview", no. December, 2010, ISSN: 1935-8237. DOI: 10.1561/2200000008. arXiv: 1007.1075. [Online]. Available: http://arxiv.org/abs/1007.1075%7B%5C%%7D0Ahttp://dx.doi.org/10.1561/2200000008.

[78] M. Sabourin and A. Mitiche, "Optical character recognition by a neural network", *Neural Networks*, vol. 5, no. 5, pp. 843–852, 1992, ISSN: 08936080. DOI: 10.1016/S0893-6080(05)80144-3.

[79] E. Abdullah, A. Idris, and A. Saparon, "Papr reduction using scs-slm technique in stfbc mimo-ofdm", *ARPN Journal of Engineering and Applied Sciences*, vol. 12, no. 10, pp. 3218–3221, 2017, ISSN: 18196608. DOI: 10.1111/ijlh.12426. arXiv: 0608246v3 [arXiv:physics].

[80] Forbes, *Future Trends In The Gartner Hype Cycle For Emerging Technologies, 2017*, 2017. [Online]. Available: http://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/%20https://www.forbes.com/sites/gartnergroup/2017/08/18/future-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/%7B%5C#%7D43030aaf4b97 (visited on 09/14/2017).

[81] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006, ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. arXiv: 1111.6189v1. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.2006.18.7.1527.

[82] Jeff Walsh, *Machine Learning: The Evolution of AI and Design*, 2016. [Online]. Available: https://redshift.autodesk.com/machine-learning/.

[83] T. M. Mitchell, *Machine Learning*, 1. 1997, p. 2, ISBN: 0070428077. DOI: 10.1145/242224.242229. arXiv: 0-387-31073-8. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20%7B%5C&%7Dpath=ASIN/0070428077.

[84] Michael Copeland(Nvidia), *The Difference Between AI, Machine Learning, and Deep Learning? | NVIDIA Blog*, 2016. [Online]. Available: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/ (visited on 09/21/2017).

[85] *What Is The Difference Between Artificial Intelligence And Machine Learning?* [Online]. Available: https://www.forbes.com/sites/bernardmarr/2016/12/06/what-is-the-difference-between-artificial-intelligence-and-machine-learning/%7B%5C#%7D26dab29d2742 (visited on 09/21/2017).

[86] *The Difference Between AI and Machine Learning*, 2017. [Online]. Available: https://www.intel.com/content/www/us/en/analytics/ai-luminary-reza-zadeh-video.html (visited on 09/21/2017).

[87] M. Kantardzic, *Data Mining*, 1. 2009, vol. 18 Suppl, IT35–6, ISBN: 9780470544341. DOI: 10.1109/9780470544341. [Online]. Available: http://ieeexplore.ieee.org/xpl/bkabstractplus.jsp?bkn=5265979.

[88] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to Spam filtering", *Expert Systems with Applications*, vol. 36, no. 7, pp. 10 206–10 222, 2009, ISSN: 09574174. DOI: 10.1016/j.eswa.2009.02.037.

[89] A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais, A. Muller, J. Kossaifi, A. Gramfort, B. Thirion, and G. Varoquaux, *Machine Learning for Neuroimaging with Scikit-Learn*. O'Reilly Media, 2014, p. 568, ISBN: 1662-5196 (Electronic)\r1662-5196 (Linking). DOI: 10.3389/fninf.2014.00014. arXiv: 1412.3919. [Online]. Available: http://arxiv.org/abs/1412.3919.

[90] D. Lowd and C. Meek, "Good Word Attacks on Statistical Spam Filters", *Conference on email and anti-spam*, 2005. [Online]. Available: http://www.utdallas.edu/%7E~%7Dmuratk/courses/dmsec%7B%5C_%7Dfiles/125.pdf.

[91] G. Hinton, L. Deng, D. Yu, G. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012, ISSN: 10535888. DOI: 10.1109/MSP.2012.2205597. arXiv: 1207.0580.

[92] A. Lazaridou, A. Peysakhovich, and M. Baroni, "Multi-Agent Cooperation and the Emergence of (Natural) Language", pp. 1–11, 2016. arXiv: 1612.07182. [Online]. Available: http://arxiv.org/abs/1612.07182.

[93] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning", *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 2970–2979, 2017, ISSN: 15505499. DOI: 10.1109/ICCV.2017.321. arXiv: 1703.06585.

[94] I. Mordatch and P. Abbeel, "Emergence of Grounded Compositional Language in Multi-Agent Populations", 2017. arXiv: 1703.04908. [Online]. Available: http://arxiv.org/abs/1703.04908.

[95] M. R. Wick and W. B. Thompson, "Reconstructive expert system explanation", *Artificial Intelligence*, vol. 54, no. 1-2, pp. 33–70, 1992, ISSN: 00043702. DOI: 10.1016/0004-3702(92)90087-E. arXiv: 1802.08129. [Online]. Available: http://arxiv.org/abs/1802.08129.

[96] P. Santos, "World ' s largest Science , Technology & Medicine Open Access book publisher :", 2017.

[97] R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, C. D. Manning, and A. Y. Ng, "Zero-Shot Learning Through Cross-Modal Transfer", 2013, ISSN: 10495258. arXiv: 1301.3666. [Online]. Available: http://arxiv.org/abs/1301.3666.

[98] X. Zhu, "Semi-Supervised Learning Literature Survey", *Sciences-New York*, pp. 1–59, 2007, ISSN: 1520-5126. DOI: 10.1.1.146.2352. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.146.2352%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[99] M. A. Yasin, W. A. Al-Ashwal, A. M. Shire, S. A. Hamzah, and K. N. Ramli, "Tri-band planar inverted F-antenna (PIFA) for GSM bands and bluetooth applications", *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 19, pp. 8740–8744, 2015, ISSN: 18196608. DOI: 10.1613/jair.301. arXiv: 9605103 [cs].

[100] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, "Emergence of Locomotion Behaviours in Rich Environments", 2017. arXiv: 1707.02286. [Online]. Available: http://arxiv.org/abs/1707.02286.

[101] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour", *Journal of Machine Learning Research*, vol. 10, pp. 2899–2934, 2017, ISSN: 2167-3888. DOI: 10.1561/2400000003. arXiv: 1706.02677. [Online]. Available: http://arxiv.org/abs/1706.02677.

[102] E. D. Munz, "Psychotherapie in der Psychiatrie", *Nervenheilkunde*, vol. 36, no. 10, pp. 800–805, 2017, ISSN: 07221541. DOI: 10.1007/s13398-014-0173-7.2. arXiv: arXiv:1011.1669v3. [Online]. Available: http://arxiv.org/abs/1201.0490.

[103] L. Vinet and A. Zhedanov, "A "missing" family of classical orthogonal polynomials", *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL '01*, pp. 26–33, Nov. 2010, ISSN: 00043702. DOI: 10.1088/1751-8113/44/8/085201. arXiv: 1011.1669. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1073012.1073017%20http://arxiv.org/abs/1011.1669%20http://dx.doi.org/10.1088/1751-8113/44/8/085201.

[104] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era", *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 843–852, 2017, ISSN: 15505499. DOI: 10.1109/ICCV.2017.97. arXiv: 1707.02968. [Online]. Available: http://arxiv.org/abs/1707.02968.

[105] R. Coulangeon and G. Nebe, "The unreasonable effectiveness of the tensor product", *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, Jan. 2012, ISSN: 1541-1672. DOI: 10.1109/MIS.2009.36. arXiv: 1201.1832. [Online]. Available: http://ieeexplore.ieee.org/document/4804817/%20http://arxiv.org/abs/1201.1832.

[106] R. M. Kaplan, D. A. Chambers, and R. E. Glasgow, "Big data and large sample size: A cautionary note on the potential for bias", *Clinical and Translational Science*, vol. 7, no. 4, pp. 342–346, 2014, ISSN: 17528062. DOI: 10.1111/cts.12178.

[107] T. Dietterich, "Overfitting and undercomputing in machine learning", *ACM Computing Surveys*, vol. 27, no. 3, pp. 326–327, 1995, ISSN: 03600300. DOI: 10.1145/212094.212114. [Online]. Available: http://portal.acm.org/citation.cfm?doid=212094.212114.

[108] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data", *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, p. 20, 2004, ISSN: 19310145. DOI: 10.1145/1007730.1007735. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1007730.1007735.

[109] D. H. Wolpert, "The Existence of A Priori Distinctions Between Learning Algorithms", *Neural Computation*, vol. 8, no. 7, pp. 1391–1420, 1996, ISSN: 0899-7667. DOI: 10.1162/neco.1996.8.7.1391. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.1996.8.7.1391.

[110] A. Ranganathan and R. H. Campbell, "An infrastructure for context-awareness based on first order logic", *Personal and Ubiquitous Computing*, vol. 7, no. 6, pp. 353–364, 2003, ISSN: 16174909. DOI: 10.1007/s00779-003-0251-x.

[111] M. R. Endsley, *Situation global assessment technique (SAGAT)*, 1988.

[112] B. N. Schilit and M. M. Theimer, "Disseminating Active Map Information to Mobile Hosts", *Netwrk. Mag. of Global Internetwkg.*, vol. 8, no. 5, pp. 22–32, 1994, ISSN: 0890-8044. DOI: 10.1109/65.313011. [Online]. Available: http://dx.doi.org/10.1109/65.313011.

[113] Y. H. Feng, T. H. Teng, and A. H. Tan, "Modelling situation awareness for Context-aware Decision Support", *Expert Systems with Applications*, vol. 36, no. 1, pp. 455–463, 2009, ISSN: 09574174. DOI: 10.1016/j.eswa.2007.09.061. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2007.09.061.

[114] N. G. Brannon, J. E. Seiffertt, T. J. Draelos, and D. C. Wunsch II, "Coordinated machine learning and decision support for situation awareness", *Neural Networks*, vol. 22, no. 3, pp. 316–325, 2009, ISSN: 0893-6080. DOI: http://dx.doi.org/10.1016/j.neunet.2009.03.013.

[115] U. Franke and J. Brynielsson, "Cyber situational awareness - A systematic review of the literature", *Computers and Security*, vol. 46, pp. 18–31, 2014, ISSN: 01674048. DOI: 10.1016/j.cose.2014.06.008. arXiv: 72. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167404814001011.

[116] J. Brynielsson, *A Gaming Perspective on Command and Control*. 2006, ISBN: 9171783652. [Online]. Available: http://www.nada.kth.se/%7B~%7Djoel/PhD.pdf.

[117] M. R. Endsley, "Toward a Theory of Situation Awareness in Dynamic Systems", *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 37, no. 1, pp. 32–64, 1995, ISSN: 0018-7208. DOI: 10.1518/001872095779049543. [Online]. Available: http://journals.sagepub.com/doi/10.1518/001872095779049543.

[118] S. Jajodia, P. Liu, V. Swarup, and C. Wang, *Cyber situational awareness: Issues and research*. 2010, vol. 46, pp. 155–176, ISBN: 9781441901392. DOI: 10.1007/978-1-4419-0140-8. [Online]. Available: http://dblp.uni-trier.de/db/series/ais/ais46.html%7B%5C#%7DLiuJZXJBL10.

[119] J. Llinas, C. Bowman, G. Rogova, and A. Steinberg, "Revisiting the JDL data fusion model II", *Space and Naval Warfare Systems Command*, vol. 1, no. 7, pp. 1–14, 2004. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord%7B%5C&%7DmetadataPrefix=html%7B%5C&%7Didentifier=ADA525721.

[120] D. Galar, A. Thaduri, M. Catelani, and L. Ciani, "Context awareness for maintenance decision making: A diagnosis and prognosis approach", *Measurement: Journal of the International Measurement Confederation*, vol. 67, pp. 137–150, 2015, ISSN: 02632241. DOI: 10.1016/j.measurement.2015.01.015. [Online]. Available: http://dx.doi.org/10.1016/j.measurement.2015.01.015.

[121] K. Wrona, T. Moye, P. Lagadec, M. Street, P. Lenk, and F. Jordan, "Cybersecurity Innovation in NATO: Lessons Learned and Recommendations", *Information & Security: An International Journal*, vol. 36, p. 11 610, 2017.

[122] V. Dutt, Y. S. Ahn, and C. Gonzalez, "Cyber situation awareness: Modeling the security analyst in a cyber-attack scenario through instance-based learning", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6818 LNCS, 2011, pp. 280–292, ISBN: 9783642223471. DOI: 10.1007/978-3-642-22348-8_24.

[123] T. Pahi, M. Leitner, and F. Skopik, "Analysis and Assessment of Situational Awareness Models for National Cyber Security Centers", *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, pp. 334–345, 2017. DOI: 10.5220/0006149703340345. [Online]. Available: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006149703340345.

[124] E. Allison Newcomb, R. J. Hammell, and S. Hutchinson, "Effective prioritization of network intrusion alerts to enhance situational awareness", *IEEE International Conference on Intelligence and Security Informatics: Cybersecurity and Big Data, ISI 2016*, pp. 73–78, 2016. DOI: 10.1109/ISI.2016.7745446.

[125] L. P. Prieto, M. J. Rodríguez-Triana, M. Kusmin, and M. Laanpere, "Smart school multimodal dataset and challenges", *CEUR Workshop Proceedings*, vol. 1828, pp. 53–59, 2017, ISSN: 16130073. DOI: 10.1145/1235. arXiv: arXiv:1603.07016v1.

[126] E. Bou-Harb, M. Husak, M. Debbabi, and C. Assi, "Big Data Sanitization and Cyber Situational Awareness: A Network Telescope Perspective", *IEEE Transactions on Big Data*, pp. 1–1, 2017, ISSN: 2332-7790. DOI: 10.1109/TBDATA.2017.2723398. [Online]. Available: http://ieeexplore.ieee.org/document/7968317/.

[127] H. Wang, X. Liu, J. Lai, and Y. Liang, "Network security situation awareness based on heterogeneous multi-sensor data fusion and neural network", in *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, 2007, pp. 352–359, ISBN: 0-7695-3039-7. DOI: 10.1109/IMSCCS.2007.15. [Online]. Available: http://ieeexplore.ieee.org/document/4392625/.

[128] Z. Power and S. Company, "Survey of Network Security Situation Awareness Changlin He", *Atlantis Press*, vol. 81, no. ICCSE, pp. 136–141, 2017. DOI: 10.2991.

[129] R. Xi, S. Jin, X. Yun, and Y. Zhang, "CNSSA: A comprehensive network security situation awareness system", in *Proc. 10th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, TrustCom 2011, 8th IEEE Int. Conf. on Embedded Software and Systems, ICESS 2011, 6th Int. Conf. on FCST 2011*, 2011, pp. 482–487, ISBN: 9780769546001. DOI: 10.1109/TrustCom.2011.62.

[130] *The pcap Next Generation Capture File Format*. [Online]. Available: https://github.com/pcapng/pcapng (visited on 05/01/2018).

[131] *Development/LibpcapFileFormat - The Wireshark Wiki*. [Online]. Available: https://wiki.wireshark.org/Development/LibpcapFileFormat (visited on 05/01/2018).

[132] *Choosing the right estimator — scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/tutorial/machine%7B%5C_%7Dlearning%7B%5C_%7Dmap/index.html (visited on 03/16/2018).

[133] L. Holmström and P. Koistinen, *Pattern recognition*, 4. Academic Press, 2010, vol. 2, pp. 404–413, ISBN: 1939-0068. DOI: 10.1002/wics.99. [Online]. Available: https://www.amazon.com/Pattern-Recognition-Sergios-Theodoridis-ebook/dp/B003FQM374?SubscriptionId=0JYN1NVW651KCA56C102%7B%5C&%7Dtag=techkie-20%7B%5C&%7DlinkCode=xm2%7B%5C&%7Dcamp=2025%7B%5C&%7Dcreative=165953%7B%5C&%7DcreativeASIN=B003FQM374.

[134] G. V. Trunk, "A Problem of Dimensionality: A Simple Example", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 3, pp. 306–307, 1979, ISSN: 01628828. DOI: 10.1109/TPAMI.1979.4766926.

[135] O. Chapelle, "Training a Support Vector Machine in the Primal", *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007, ISSN: 0899-7667. DOI: 10.1162/neco.2007.19.5.1155. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.2007.19.5.1155.

[136] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space", *Philosophical Magazine Series 6*, vol. 2, no. 11, pp. 559–572, 1901, ISSN: 1941-5982. DOI: 10.1080/14786440109462720. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/14786440109462720.

[137] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions", pp. 1–74, 2009, ISSN: 0036-1445. DOI: 10.1137/090771806. arXiv: 0909.4061. [Online]. Available: http://arxiv.org/abs/0909.4061.

[138] M. E. Tipping and C. M. Bishop, "Mixtures of Probabilistic Principal Component Analyzers", *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999, ISSN: 0899-7667. DOI: 10.1162/089976699300016728. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/089976699300016728.

[139] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, *Learning Activation Functions to Improve Deep Neural Networks*. 2014, vol. 7246, pp. 122–133, ISBN: 9783642352881. DOI: 10.1007/3-540-49430-8. arXiv: 1412.6830. [Online]. Available: http://arxiv.org/abs/1412.6830.

[140] *sklearn.preprocessing.StandardScaler — scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (visited on 03/28/2018).

[141] Y. C. Ho and A. K. Agrawala, "on Pattern Classification Algorithms Introduction and Survey", *Proceedings of the Institute of Electrical and Electronics Engineers*, vol. 56, no. 12, p. 2101, 1968.

[142] H. W. Brown and J. Roberts, "Exploring the factors contributing to sibling correlations in BMI: A study using the panel study of income dynamics", *Obesity*, vol. 20, no. 5, pp. 978–984, 2012, ISSN: 19307381. DOI: 10.1038/oby.2011.351.

[143] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are Loss Functions All the Same?", *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004, ISSN: 0899-7667. DOI: 10.1162/089976604773135104. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/089976604773135104.

[144] E. D. Munz, *Psychotherapie in der Psychiatrie*, 10. 2017, vol. 36, pp. 800–805, ISBN: 9780874216561. DOI: 10.1007/s13398-014-0173-7.2. arXiv: arXiv:1011.1669v3.

[145] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943, ISSN: 00074985. DOI: 10.1007/BF02478259. arXiv: arXiv:1011.1669v3.

[146] Xue-Wen Chen and Xiaotong Lin, "Big Data Deep Learning: Challenges and Perspectives", *IEEE Access*, vol. 2, pp. 514–525, 2014, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2325029. arXiv: MSP.2010.939038 [10.1109]. [Online]. Available: http://ieeexplore.ieee.org/document/6817512/.

[147] C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin, *Intrusion detection by machine learning: A review*, 2009. DOI: 10.1016/j.eswa.2009.05.029. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2009.05.029.

[148] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[149] D. Lee and K. Myung, "Read my lips, login to the virtual world", *2017 IEEE International Conference on Consumer Electronics, ICCE 2017*, pp. 434–435, 2017, ISSN: 09252312. DOI: 10.1109/ICCE.2017.7889386. arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980.

[150] G. Andrew and J. Gao, "Scalable training of L ¹ -regularized log-linear models", *Proceedings of the 24th international conference on Machine learning - ICML '07*, pp. 33–40, 2007. DOI: 10.1145/1273496.1273501. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1273496.1273501.

[151] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A Limited Memory Algorithm for Bound Constrained Optimization", *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995, ISSN: 1064-8275. DOI: 10.1137/0916069. arXiv: arXiv:1011.1669v3. [Online]. Available: http://epubs.siam.org/doi/10.1137/0916069.

[152] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting", in, vol. 139, 1995, pp. 23–37, ISBN: 3540591192. DOI: 10.1007/3-540-59119-2_166. arXiv: /statistics.berkeley.edu/sites/default/files/tech-reports/367.pdf [http:]. [Online]. Available: http://link.springer.com/10.1007/3-540-59119-2%7B%5C_%7D166.

[153] L. Breiman, "Arcing the edge", *Statistics*, vol. 4, pp. 1–14, 1997. DOI: 10.1.1.367.9480.

[154] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class AdaBoost", *Statistics and Its Interface*, vol. 2, no. 3, pp. 349–360, 2009, ISSN: 19387989. DOI: 10.4310/SII.2009.v2.n3.a8. arXiv: arXiv:1011.1669v3. [Online]. Available: http://www.intlpress.com/site/pub/pages/journals/items/sii/content/vols/0002/0003/a008/.

[155] B. Kégl, "The return of AdaBoost.MH: multi-class Hamming trees", 2013. arXiv: 1312.6086. [Online]. Available: http://arxiv.org/abs/1312.6086.

[156] *adaboost | Sachin Joglekar's blog*. [Online]. Available: https://codesachin.wordpress.com/tag/adaboost/ (visited on 04/12/2018).

[157] W. T. Wong and S. H. Hsu, "Application of SVM and ANN for image retrieval", *European Journal of Operational Research*, vol. 173, no. 3, pp. 938–950, 2006, ISSN: 03772217. DOI: 10.1016/j.ejor.2005.08.002.

[158] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines", *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, pp. 1702–1707, 2002, ISSN: 1098-7576. DOI: 10.1109/IJCNN.2002.1007774. [Online]. Available: http://ieeexplore.ieee.org/document/1007774/.

[159] R. Kumar and D. Verma, "Classification Algorithms for Data Mining: A Survey", *International Journal of Innovations in Engineering ...*, vol. 1, no. 2, pp. 7–14, 2012. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.305.2167%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[160] N. Mukherjee and S. Mukherjee, *Predicting Signal Peptides with Support Vector Machines*. 2002, vol. 2388, pp. 1–7, ISBN: 978-3-540-44016-1. DOI: 10.1007/3-540-45665-1_1. [Online]. Available: http://link.springer.com/10.1007/3-540-45665-1%7B%5C_%7D1.

[161] C. Serrano-Cinca, Y. Fuertes-Callén, and C. Mar-Molinero, "Measuring DEA efficiency in Internet companies", *Decision Support Systems*, vol. 38, no. 4, pp. 557–573, 2005, ISSN: 01679236. DOI: 10.1016/j.dss.2003.08.004. arXiv: 1011.1669v3.

[162] R. A. FISHER, "the Use of Multiple Measurements in Taxonomic Problems", *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936, ISSN: 20501420. DOI: 10.1111/j.1469-1809.1936.tb02137.x. arXiv: arXiv:1011.1669v3. [Online]. Available: http://doi.wiley.com/10.1111/j.1469-1809.1936.tb02137.x.

[163] 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 {P}ython", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[164] Scikit-learn, *1.17. Neural network models (supervised)*. [Online]. Available: http://scikit-learn.org/stable/modules/neural%7B%5C_%7Dnetworks%7B%5C_%7Dsupervised.html (visited on 04/12/2018).

[165] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning", *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008, ISSN: 1553-877X. DOI: 10.1109/SURV.2008.080406. [Online]. Available: http://ieeexplore.ieee.org/document/4738466/.

[166] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection", in *SIAM International Conference on Data Mining*, 2003, pp. 25–36.

[167] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection", *Proceedings - IEEE Symposium on Security and Privacy*, pp. 305–316, 2010, ISSN: 10816011. DOI: 10.1109/SP.2010.25.