UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Enhancing Cybersecurity Intelligence through Machine Learning: Cluster and Predictive Analysis of Honeypot Data

Author:
David ROSADO

Supervisors:
Santiago ROMEU
Dra. Laura IGUAL

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

February 13, 2025

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Master Thesis Title

by David ROSADO

In an era marked by an ever-increasing digital threat landscape, the need for robust proactive cybersecurity measures has become paramount. This master's thesis, conducted in collaboration with the Cybersecurity Agency of Catalunya, leverages a comprehensive dataset provided by the Global Cyber Alliance. We will be engaging with data consisting of cyberattacks records collected from honeypots distributed around the word. The primary objectives of this research are twofold: 1) Conduct a cluster analysis on this dataset to gain deeper insights into its structure and reveal common attack patterns and 2) Develop a predictive model capable of estimating the number of cyberattacks a honeypot is expected to receive.

The initial phase of this study involves a exploratory data analysis, which provides valuable insights into the characteristics of cyberattacks observed across diverse honeypot locations. Subsequently, advanced cluster analysis techniques are employed to categorize cyberattack data, facilitating the identification of common attack patterns. In the context of the predictive analysis, which represents a pivotal aspect of this research, we aim to forecast the expected number of cyberattacks a honeypot may face. This proves highly beneficial by providing cybersecurity experts with a proactive tool for resource allocation and strategy development.

Acknowledgements

I would like to express my sincere gratitude to Laura and Santi, my advisors, whose guidance and support have been invaluable throughout this journey. Special thanks to Santi for his patience and clear explanations of everything I needed, which made this process more enriching.

I extend my sincere thanks to the Agència de Ciberseguretat de Catalunya for providing me with the space and trust to develop this project in collaboration with them. The opportunity to work with this organization has been a tremendous privilege.

A special acknowledgment is due to the Global Cyber Alliance for generously sharing data and facilitating monthly meetings, significantly contributing to the shaping and validation of the research results.

Finally, my deepest appreciation goes to the individuals who support me every day. To all those who have offered a helping hand, a listening ear, or words of encouragement, I am truly thankful. Your support has made a significant impact on both my academic and personal life.

Contents

Chapter 1

Introduction

1.1 Cybersecurity significance

In today's digitally connected world, a large amount of data is being generated, processed, shared and stored on the internet every day. As the usage of the internet continues to grow and this data-driven environment keeps evolving, the crucial significance of cybersecurity becomes progressively clear. According to the Cybersecurity and Infrastructure Security Agency (CISA), cybersecurity is the art of protecting networks, devices, and data from unauthorized access or criminal use and the practice of ensuring confidentiality, integrity, and availability of information.

With the ever-expanding volume of data and the proliferation of internet-connected devices, the number of cyberattacks is on the rise. For instance, in 2015, the United States Office of Personnel Management (OPM) suffered a significant data breach. Approximately 21.5 million records of government employees, such as names, social security numbers, addresses and more, were stolen (Dasgupta, Akhtar, and Sen, 2022). Most recently, in March 2023, the Hospital Clinic of Barcelona fell victim to a huge cyberattack. The cyberattackers managed to stole 4.5 terabytes of data, leaving the hospital's computer system in shambles. Consequently, the hospital had no choice but to cancel approximately 150 non-urgent operations and between 2000 and 3000 checkups, as reported by the Generalitat of Catalunya.

Cybercriminals often steal valuable and compromising information, including financial records, personally identifiable information, or sensitive data. They exploit this stolen data by either making it public or selling it on the black market, resulting in significant economic and societal consequences. For example, in the case of the Barcelona hospital, the cyberattackers demanded the payment of 4.5 million dollars in order not to make public the data they had stolen. When examining the entire sector, statistical research indicates that, in recent years, cybercrimes have led to the theft of over 400 billion dollars (Rege and Mbah, 2018).

Given the ever-evolving complexity of cyberattacks, ongoing research is concentrated on leveraging the potential of machine learning and its tools to detect, stop, and respond to these sophisticated attacks. Several machine learning techniques currently in use for this topic include anomaly detection and network risk scoring. Anomaly detection aims to identify events that diverge from a predefined set of typical behaviors, essentially uncovering patterns in data that do not align with anticipated behaviors (Dua and Du, 2016). On the other hand, network risk scoring involves the use of quantitative measures to assign risk scores to various sections of a network, enabling security experts to efficiently allocate essential resources as needed (Rege and Mbah, 2018).

1.2 Problem statement

The Cybersecurity Agency of Catalonia is in charge of implementing public cybersecurity policy and developing the Government of Catalonia's cybersecurity strategy. Thanks to their partnership with the Global Cyber Alliance (GCA), a non-profit organization committed to enhancing online safety by mitigating cyber risks, we have access to an extensive dataset. This data includes records of cyberattacks collected from honeypots distributed around the world. Some of the key information you can access includes the origin and target locations of the attack, the executed commands, the IP addresses of both the attacker and target, and the username/password combination used to log into the system. The data, as well as the honeypot meaning, are described in detail in section 2.

This thesis focuses on exploring and exploiting the data provided by the GCA using Machine Learning (ML) techniques. The first step in our analysis begins with an Exploratory Data Analysis (EDA). After retrieving the data from the GCA Application Programming Interface (API), our goal is to conduct an initial examination of the data through visualization. Data visualization plays a crucial role in Data Science because, as we will soon discover, it allows us to obtain valuable insights without relying on any ML algorithms. Following this step, we aim to address the following questions:

1. Which country experiences the highest impact from cyberattacks?
2. Which country has conducted the highest number of cyberattacks?
3. Which is the average duration of cyberattacks?
4. What are the most frequently employed attack commands by cyberattackers?

Once we address these questions and gain better understanding of the data's composition, our objective is to identify attack patterns. We aim to categorize the data through cluster analysis, organizing it into groups where data within the same group shares similarities in terms of attack type. This stage holds immense significance for cybersecurity professionals. It goes beyond just recognizing patterns and groupings; it empowers them to detect malware in their devices (Martínez Torres, Iglesias Comesaña, and García-Nieto, 2019). Some of the questions we ask ourselves at this step are:

1. What are the most commons cyberattacks patterns?
2. Is the data distinctly separated, such that the clusters are significantly different from each other?

The last phase of our project involves conducting a predictive analysis. The goal is to predict the number of cyberattacks a honeypot is expected to receive. This predictive capability is invaluable in enabling security experts to allocate the necessary resources and develop effective strategies for countering these cyberattacks. In this step, we aim to address the following questions:

1. Is it possible to predict the expected number of cyberattacks that a honeypot will receive?

1.3 Literature review

In this section, we provide a detailed examination of research carried out in the field, focusing on the primary publications related to cluster analysis and predictive analysis using honeypot data.

In (Kheirkhah et al., 2013) the authors analyzed data collected from honeypots. They categorized this data into three distinct groups based on the emulated commands used by intruders. The first category comprised intruders who quickly logged out after gaining access. Their departure could be due to various reasons, such as planning to return later or realizing that the system was indeed a honeypot. The second category consisted of intruders who executed basic fingerprinting commands on the server to gather essential information about it. The third category encompassed intruders who not only performed fingerprinting but also proceeded to download and execute malware on the compromised system.

Three years later, a study was conducted to analyze honeypot data using statistical methods (Melese and Avadhani, 2016). In this study, the authors identified a common sequence of steps that cyberattackers typically follow when infiltrating a system. First, attackers often attempt to gain access by guessing credentials using automated dictionary-based attacks. Among the frequently encountered credentials were default passwords like *admin, root, 123456,* and others. Once they successfully log in to the SSH server, most attackers initiate their intrusion by downloading tools from remote servers. The study also aimed to identify the most commonly used shell commands executed by attackers. Additionally, the authors tracked the geographic locations of the attackers based on their IP addresses and documented the SSH client used during each session. Notably, a significant number of attacks originated from China and the United States, and the SSH client *SSH-2.0-libssh2* was prevalent among attackers.

A novel clustering approach was introduced in the work by (Fraunholz et al., 2017), which incorporated ML for the purpose of classifying attacks. In the initial attempt to categorize honeypot data, attacks were classified based on the skill levels of the attackers. This involved assigning points to attackers for each command identified in a predefined list created by the authors. Consequently, attackers with a higher total number of points were considered more skilled. Subsequently, clustering algorithms were applied as part of a ML-based approach, utilizing two distinct feature sets. On one hand, the authors proposed clustering based on the credentials used at the start of each session, while on the other hand, clustering was suggested based on the commands issued by the attackers. The results revealed that the ML-based methods excelled in creating distinct clusters, whereas the skill-based approach failed due to issues with the honeypot configuration.

Following the same research line, the study (Valero et al., 2020) presents a method for detecting and categorizing cybersecurity threats using SSH-based honeypot systems. Although their initial attempt to create broad clusters of honeypot data using ML techniques did not succeed, they did manage to establish three distinct threat levels based on specific features. The first level represents the lowest threat, including commands focused on gathering information about the target system. The second level, categorized as an intermediate threat, encompasses commands related to installing software packages, compiling code, and granting new permissions to

unknown files. The third level comprises sessions attempting to download and execute suspicious software, remove critical system files, and suspend or shut down vital system processes, making it the most critical threat level. Furthermore, the researchers employed various supervised ML algorithms for classification, using the aforementioned threat level labels as the target. The resulting model successfully identified and categorized cyberthreats, achieving A F1-score[1] of 0.9714, thus validating an optimal solution.

To wrap up the cluster analysis literature review, in 2021, the study (Tabari, Ou, and Singhal, 2021) introduced an innovative method for constructing a honeypot ecosystem. In this study, researchers collected data and conducted a cluster analysis by creating a similarity matrix for each unique command pair. They employed a Gaussian Mixture Model to generate 50 distinct clusters, helping them identify the objectives of attackers within each group.

Shifting to predictive analysis, the research presented in (Zuzčák and Bujok, 2021) explored several traditional approaches to forecast cyberattacks, treating them as a time series forecasting model. The study employed various techniques, including ARIMA, SARIMA, GARCH, and others. To evaluate the models, visual assessments and the MAPE metric were employed. The findings indicate that, despite employing multiple methods and fine-tuning them, achieving satisfactory accuracy across all observed aspects remains challenging. Additionally, the study highlighted that predictive methods incorporating a seasonal component in the time series become more efficient as the number of available seasons increases.

In (Fang et al., 2019), the authors introduced a Deep Learning framework known as BRNN-LSTM, which utilizes bi-directional recurrent neural networks with long short-term memory. This framework demonstrates the ability to forecast cyberattacks. The authors conducted experiments with five real-world datasets, revealing that the proposed framework outperforms traditional statistical approaches like ARIMA models.

In the research discussed in (Husák et al., 2021), three distinct approaches to predictive methods in cyber defense are explored. The first method involves utilizing data mining to identify frequent attack scenarios, projecting ongoing cyberattacks based on this information. The second approach employs a dynamic network entity reputation score to anticipate malicious actors. The third approach utilizes classical time series analysis to forecast attack rates in the network (we were interested in this approach). The researchers conducted a comprehensive evaluation of these three methods within a common environment—a intrusion detection alert sharing platform. The experimental results demonstrated that all three methods reached a sufficient technology readiness level, making them suitable for experimental deployment in operational settings. Importantly, they exhibited promising accuracy and usability, affirming their potential in practical applications.

[1] F1 score is a machine learning evaluation metric that combines precision and recall scores, making it particularly useful for assessing imbalanced data.

Chapter 2

Data Ovierview and Sources

2.1 Honeypots

Before examining the data, it is essential to understand the concepts of honeypots, honeynets, and honeyfarms. These concepts are all cybersecurity mechanisms used for deception and threat detection, but they differ in terms of scale, complexity, and purpose.

A **honeypot** is an isolated security system whose value lies in being probed, attacked, or compromised. The key to these systems is that honeypots look like a real computer system, fooling cybercriminals into thinking it is a legitimate target. These systems simulate networks and application services such as Secure Shell (SSH) and Telnet, amongst others.

There are several reasons for incorporating a honeypot into your system, including distracting attackers from your machine, receiving early alerts about emerging threats, and collecting valuable data about attackers for analysis both during and after an attack (Mokube and Adams, 2007). Honeypots can be categorized according to the level of interaction permitted between intruders and the system:

1. *Low-interaction honeypots*. These honeypots emulate a limited set of services and behaviors. They do not involve real operating systems or applications, which reduces the risk but restricts their functionality. They are easy to deploy and maintain, and they can be useful to analyze spammers.

2. *Medium-interaction honeypots*. These honeypots, as the previous ones, do not have an operating system installed, but they can run actual services and applications. They are more realistic than low-interaction honeypots, since there is more for the attacker to interact with.

3. *High-interaction honeypots*. These honeypots use real operating systems and applications, providing a fully realistic environment. They have the potential to collect a significant amount of information about the attack because the attacker can interact with the entire operating system. However, they are the most complex to maintain and involve considerable risk.

A **honeynet** is a high-interaction honeypot consisting of a real system, services and applications, designed to capture extensive information on threats. Honeynets are harder to set up and maintain than individual honeypots, but they provide a more comprehensive view of threats, making them more powerful for research purposes (Mokube and Adams, 2007).

Finally, a **honeyfarm** is an even larger-scale deployment of honeypots and honeynets, often distributed across multiple physical locations or cloud environments. Large organizations and enterprises utilize honeyfarms, which are the most complex to set up and maintain, to capture substantial volumes of data for monitoring and post-analysis.

Let us provide a brief overview of the benefits and drawbacks associated with these systems (Mokube and Adams, 2007). Here are some of the advantages:

1. *Valuable information*. Data collected from a honeypot is highly valuable because it contains real interactions from cyberattacks, which makes the information obtained from it useful for post-analysis. Additionally, this data can help security professionals understand the latest trends and methods in cyberattacks.

2. *Early Threat Detection*. Honeypots can detect threats early in the attack cycle because they are often the first systems targeted by attackers. This allows security teams to respond proactively.

On the other hand, there are certain disadvantages to consider:

1. *Limited vision*. A honeypot can exclusively collect data when an attacker engages with it, meaning that attempts on other system components will remain undetected unless the honeypot itself is directly targeted.

2. *Maintenance*. Keeping honeypots realistic and up-to-date can be challenging. A simple mistake in the maintenance can lead to cyberattackers discovering the presence of the honeypot.

In closing this section, I would like to mention the availability of open-source honeypot software, known for its user-friendly installation and versatility. One of the most widely used options is Cowrie, and you can find installation instructions on its GitHub repository.

2.2 Automated IoT Defence Ecosystem (AIDE)

The concepts presented in the previous section plays a crucial role in our work. In August 2019, the GCA launched the Automated IoT Defence Ecosystem (AIDE), a first-of-its-kind cybersecurity development platform for Internet of Things (IoT) products. The platform gathers data on cyberattack information through the use of a honeyfarm system. The platform's applications can range from just reviewing and visualizing data to the exploration of vulnerabilities and attack signatures. Across this spectrum of uses, the main objective stays the same: enhancing the security of the IoT. The honeyfarm system consists of more than 200 honeypots and honeynets distributed around the world.

The Cybersecurity Agency of Catalonia, in its role as a GCA partner, is keen to use this data for research purposes. They aim to study emerging trends, create and deploy algorithms to detect new threats in the cybersecurity field, investigate vulnerabilities, and establish policies to enhance the cyber-protection of Catalonia. Certainly, the primary goal of this project is to assist the Agency in deciding whether to implement honeyfarms in Catalonia. Despite their cost and maintenance challenges, the project aims to demonstrate the valuable and useful information that can be derived from the data, contributing to the study of cyberdefense.

2.3 API Data extraction

Before moving on to the next section, where we will analyze the conducted EDA, let me provide some information regarding the AIDE API and how the data was extracted. To begin with, please note that all the code used for this master's final thesis can be found on Github[1].

We adhere to a fundamental principle when making requests. Specifically, we focus on records in which the cyberattacker has executed at least one command on the targeted system. Instances where the cyberattacker does not execute any commands can be attributed to one of the following four main reasons: either they are in the process of generating new passwords dictionaries for future brute-force attacks, planning to come back later, realizing that the honeypot was indeed a honeypot, or presuming that the desired server does not exist or is currently unavailable, as reported in (Kheirkhah et al., 2013).

Based on this principle, we have developed a Jupyter notebook for sending requests to the AIDE API. Within the *Requests* Github folder, you will discover the mentioned file that contains two functions: one for obtaining data for EDA and cluster analysis, and another for obtaining data for predictive analysis.

2.4 Exploratory Data Analysis (EDA)

The honeyfarm build by the GCA contains records of cyberattacks, including a variety of features. To facilitate data analysis and clustering, a three-month dataset was collected, spanning from May 1, 2023, to July 31, 2023. Let us provide a brief overview of the features at our disposal.

2.4.1 Features explanation

The three-month dataset collected comprises 5 million records and includes 48 distinct features. During our analysis, we identified certain variables with significant data gaps, either containing a high number of missing values or lacking relevance to our research objectives. As a result, we opted not to include these variables in our analysis. Table 2.1 presents the list of features that we have requested, along with detailed explanations for each. Additionally, Appendix A provides explanations for all 48 available features.

2.4.2 Statistical Analysis and Data Visualization

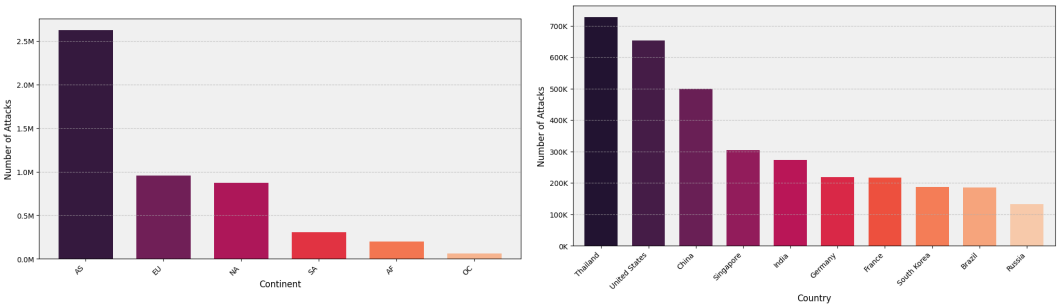
In the repository, you will find a folder called *EDA* where a comprehensive data exploration, complete with charts, awaits. Let us delve into a few of them to gather insights about the data.

Perhaps one of the most straightforward questions that arises when handling this kind of data is about the connection between the cyberattacks and their origin/destination. Concerning the origin of cyberattacks, which refers to where they originate, Figure 2.1 shows that Asia, followed by Europe and North America, serves as the primary

[1]Please click the link to view the code. If you encounter any issues accessing it, feel free to contact me.

Features	Explanation
Commands	Unix commands executed by the attacker that are fully emulated
Loggedin	Username/password combination that successfully logged in
EndTime	Timestamp of the end of the session/attack
StartTime	Timestamp of the start of the session/attack
Geoip.continent	Continent where the honeypot is located
Geoip.country	Country where the honeypot is located
hostGeoip.continent	Continent where the attacker is located
hostGeoip.country	Country where the attacker is located
HostIP	Attacked honeypot IP
HostPort	Attacked honeypot port
PeerIP	Attacker honeypot IP
PeerPort	Attacker honeypot port
Protocol	SSH or Telnet
Version	Version of the SSH library used by the attacker

TABLE 2.1: Requested features along with a brief explanation.



(A) Count per continent.

(B) Count per country.

FIGURE 2.1: Number of cyberattacks launches by region.

sources of cyberattacks. A closer look at the breakdown by countries reveals that Thailand, the United States and China are leading in this aspect. Notice that approximately half of all Asia attacks come from Thailand and China combined. On the other hand, when examining cyberattacks based on their destination, Figure A.1 reveals that Asia experiences the highest number of attacks, with North America and Europe following closely behind. In terms of individual countries, the United States is now the most affected, followed by Thailand and Singapore.

Looking at the charts, it is clear that Asia not only generates the highest number of cyberattacks, but is also the primary target. Given Asia's status as the most populated continent with a significant online presence, it becomes a bigger target for cybercriminals. Additionally, the varying degrees of technological advancement among Asian countries contribute to the disparity, with some having advanced systems that attract attention and others possessing less robust cybersecurity infrastructure, making them more susceptible to attacks.

Taking a closer look at the United States, the country facing the highest impact, Figure A.2 shows that Asia is the primary continent source of attacks for the country, accounting for around 50% of the total attacks directed at the United States. Additionally, near a quarter of all attacks originating from Asia are specifically aimed at the United States. This pattern strongly suggests that the United States consistently emerges as a notable target for cyberthreats from Asia, a trend influenced by complex interactions of historical factors, including the persisting echoes of the Cold War, ongoing political tensions, and broader geopolitical complexities between the two regions.

Another important variable in the study is the protocol employed to establish the connection during an attack. SSH and Telnet are two communication protocols that facilitate cyberattackers in establishing connections with remote systems. Essentially, these communication protocols dictate how data is transmitted among various devices within a network. While both serve a similar function, SSH is considered more secure and modern because it utilizes encrypted communication. In contrast, Telnet sends data, including passwords, in plain text across the network. The absence of encryption makes Telnet highly vulnerable, and its usage is strongly discouraged, particularly when connected to the internet. A pie chart depicted in A.3 illustrates that 96.4% of the attacks utilize SSH, with the predominant SSH version being *SSH-2.0-Go*, probably due to its simplicity, efficiency, and user-friendly nature.

As we delve further into data exploration, one key variable from which we can extract valuable insights is the duration of the attack. By utilizing both the start and end times of the attack, we can easily calculate the time difference to determine the attack duration. It is reasonable to anticipate that longer durations correspond to more threatening attacks, as cyberattackers have additional time to engage in malicious activities. The histograms illustrated in A.4 reveal that the majority of attacks last less than one minute, and in fact, most are completed in less than 5 seconds. It seems that cyberattackers quickly breaks into the system, carry out specific commands, and promptly exit. This behavior may be attributed to various factors, such as attackers aiming to introduce a single command line, assess the system, and quit, perhaps with plans to return later. Alternatively, attackers might realize they have entered a honeypot during their system check. Additionally, when considering attacks lasting more than one minute, the peak duration falls between two and three minutes, with a maximum duration of 15 minutes. Consequently, a clear trend towards shorter-duration attacks is evident.

With this prevailing trend of shorter-duration cyberattacks, a pertinent question emerges: which commands are mainly using cyberattackers to execute their malicious activities within this abbreviated timeframe? As illustrated in Figure A.5, over 50% of these attacks involve, or are exclusively characterized by, the command *echo -e "\\x6F\\x6B"*. Essentially, this command serves to verify the functionality of the breached console by printing the word *ok*. Furthermore, additional commands such as *uname -a* and *uname -s -v -n -r -m* are also frequent. These commands provide a comprehensive summary of system information, encompassing details such as the operating system, machine hardware, kernel version, and more. At this point, it is plausible to suspect that the majority of these attacks involve scanning and probing the system, possibly, as seen before, with the intent of returning for a more substantial attack or due to the realization that the system may indeed be a honeypot.

Finally, to conclude this data exploration, let us delve into an interesting variable: the *loggedin* feature, which reveals the username/password combinations that successfully gained access. Examining passwords, Figure A.6 depicts the most frequently used ones in a word cloud chart. In particular, passwords such as *admin, 12345,* and *password* stand out as the most common. The honeypots deliberately expose easily guessable passwords, allowing attackers to gain entry using extensive password dictionaries.

Note that this study can focus on a particular country rather than the entire dataset. For example, we could have analyzed data on cyberattacks received in Spain and conducted the same study. This research is especially relevant for the Cybersecurity Agency of Catalonia because the decision to implement honeyfarms in Catalonia depends on valuable insights we could derive from the data. If we could identify trends specific to a country, such as Spain, compared to the entire dataset, it would suggest that cyberattacks are targeted at specific locations rather than being launched without a fixed destination. However, during our exploration (available on Github), we discovered that there is no significant difference when studying data in Spain compared to the entire dataset. This implies that there is no evidence to support the idea that these cyberattacks are specifically targeted at Spain. Hence, while the data may yield valuable insights, we cannot extract country-specific findings; only general facts can be derived.

Chapter 3

Methodology

In the earlier section, we introduced important concepts like honeypots, and we conducted an EDA to provide the reader with a clear understanding of the data's structure. Our current objective is two-fold:

1. *Cluster Analysis:* Our first aim is to organize the data into clusters, where items within the same cluster exhibit similarities in both their characteristics and attack patterns.

2. *Predictive Analysis:* The second goal involves conducting a predictive analysis. Specifically, when given a honeypot, we intend to forecast the number of attacks that the honeypot is expected to receive.

To begin, let us delve into the details of how the cluster analysis was executed.

3.1 Cluster Analysis

Cluster analysis, also known as clustering, is an unsupervised machine learning technique that involves grouping a set of objects or data points into subsets, or clusters, based on their similarity. The goal of cluster analysis is to identify inherent structures in the data and organize it into meaningful groups. More generally, as stated in (Gan, Ma, and Wu, 2020), the basic problem of clustering may be stated as follows:

Given a set of data points, partition them into a set of groups which are as similar as possible.

In our specific framework, it's important to highlight that the records of cyberattacks are not labeled. Therefore, employing this approach would be beneficial for categorizing the data and gaining a clearer understanding of its composition. Once this classification is completed, we expect the ability to identify groups of attacks, each characterized by patterns distinct from those in other groups.

To carry out the research, we propose employing a two-stage clustering algorithm for the analysis. The primary objective of the first stage is to establish a general classification, whereas the second stage focuses on creating a more detailed grouping. In the first stage, we exclusively utilize commands fully emulated by attackers. Our approach involves generating embeddings for these commands to train a clustering algorithm. As we will see, this process results in the identification of three distinct groups, corresponding to low, medium, and high threat levels.

In the second stage, we focus on feature extraction for the medium and high threat

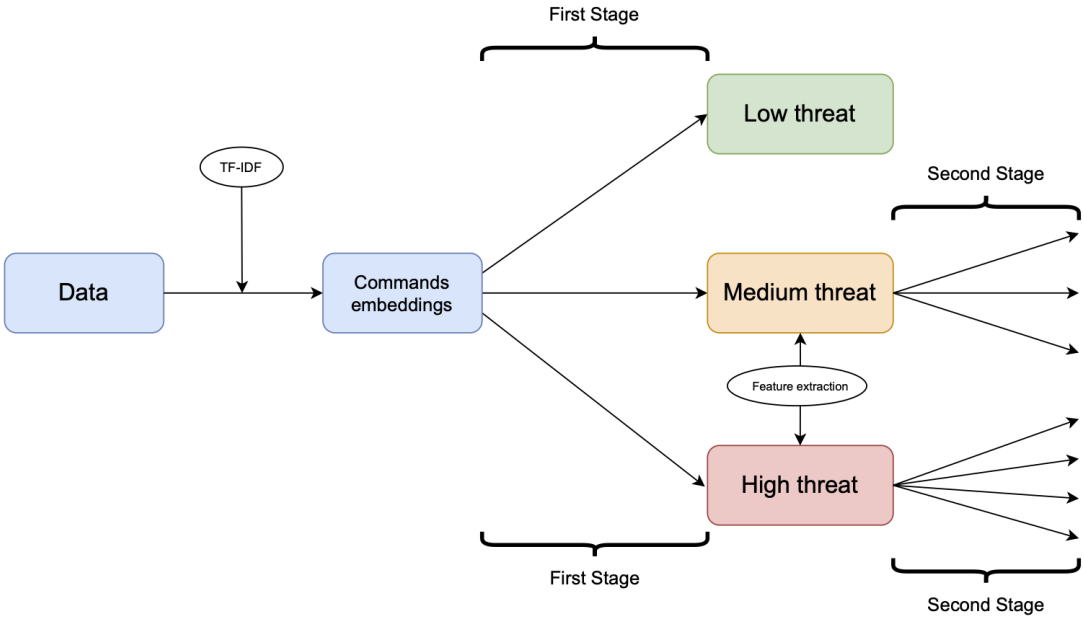


FIGURE 3.1: Diagram illustrating the sequential process of the two-stage clustering analysis.

levels. Subsequently, using the extracted data, another clustering algorithm is applied to achieve a more detailed grouping within each threat level. The clustering flow described is illustrated in Figure 3.1. Now, let us explain in detail the steps taken in both the stages.

3.1.1 First Clustering Stage

In the first clustering stage, we decided to exclusively use the commands that attackers fully emulate. It is reasonable to assume that the commands introduced by attackers contain substantial information about the nature of the attack. Therefore, our initial approach is to focus on utilizing this specific feature to achieve an initial grouping. The command feature provides a comprehensive list of all commands used in each attack. An example of such a command could be:

[lscpu | grep "CPU(s):" echo -e "GPuGJ45K3GGu\\\\ nGPuGJ45K3GGu" | passwd cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://185.224.128.141/linux/bins.sh; chmod +x bins.sh; sh bins.sh; curl –output bins.sh http://185.224.128.141/linux/bins.sh; chmod +x bins.sh; sh bins.sh]

To use the commands effectively, we must create embeddings for them, with the goal of acquiring a vector representation. One well-known method for achieving this is through the use of the Term Frequency-Inverse Document Frequency (TF-IDF) technique. TF-IDF is a numerical statistic that indicates the significance of a word in a document within a corpus. For text embedding, TF-IDF involves representing each document as a high-dimensional vector. In this representation, each dimension corresponds to a term in the vocabulary, and the value at each dimension represents the TF-IDF score of the corresponding term in the document.

Now, we can leverage the computed embeddings to calculate the cosine similarity

Original Command	Normalized Command
/bin/busybox ZSQCG	*/bin/busybox*
root:Q9szwntZCIbJ | chpasswd	*root:root | chpasswd*

TABLE 3.1: Command normalization examples.

between every pair of commands, creating a matrix known as the cosine similarity matrix. This matrix essentially quantifies the degree of similarity or dissimilarity between each pair of commands. In the first stage, we utilize a clustering technique that groups commands based on the mentioned similarity matrix. For this purpose, two separate clustering algorithms are examined, and a model selection process is carried out to determine the most effective model.

However, it is important to note that we are dealing with a dataset of 5 million records. Attempting to compute the similarity matrix would involve storing a 5 million by 5 million matrix, which is not feasible. To address this challenge, we opt for a different approach. Note that it is unnecessary to compute the cosine similarity matrix for the entire set of commands; it is only necessary for the unique commands. Despite this, we still have a substantial number, precisely 158.436, to compute the matrix. As a solution, we opt to use command normalization to further reduce the dimensionality, particularly in cases where two distinct commands may be practically identical except for a slight variation in a single word. While we do lose some specific information, it enables us to compute the matrix and perform clustering to achieve a broad classification[1]. Table 3.1 illustrates some examples demonstrating the techniques employed to normalize commands in the dataset, aiming to reduce dimensionality. Through this normalization process, the dataset is condensed from 158.436 records to 3299. As a result, the cosine similarity matrix now takes the form of a 3299×3299 matrix.

Let us examine the two clustering algorithms used in this initial stage.

3.1.1.1 Spectral Clustering

Spectral Clustering is a clustering method that utilizes a similarity matrix of the data (in our case, the cosine similarity matrix) to identify clusters. For a given set of data points, $x_1 \ldots, x_n$, and a defined measure of similarity, $s_{ij} \geq 0$ between all pairs of data points x_i and x_j, we can represent the data using a similarity graph $G = (V, E)$. Each vertex v_i in this graph represents a data point x_i and each vertex is connected by an edge weighted by s_{ij}. Clustering using this framework involves creating a partition such that the edges between different groups have very low weights (meaning that points in different clusters are dissimilar from each other) and the edges within a group have high weights (meaning that points within the same cluster are similar to each other) (Von Luxburg, 2007).

The general approach consists of applying a standard clustering technique to a pertinent set of eigenvectors derived from the Laplacian matrix of the similarity matrix.

[1]Here, we anticipate that the clustering will provide a broad classification as we may lose some information in the process.

There are various ways to define the Laplacian matrix, but we will use the random walk normalized Laplacian in this context. If we denote the similarity matrix as $S = (s_{ij})$, the random walk normalized Laplacian is defined as follows:

$$L_{rw} = I - D^{-1}S \text{ where } D_{ii} = \sum_j s_{ij} \text{ and } D_{ij} = 0 \text{ for } i \neq j.$$

In summary, the algorithm proceeds through the following steps:

1. Compute the random walk normalized Laplacian from the similarity matrix.
2. Compute the first k eigenvectors and consider the matrix formed by these eigenvectors.
3. Use a standard clustering technique (e.g., K-means).

This approach is currently available in Python through scikit-learn. It is important to note that we have already generated the similarity matrix, so there is no need for the method to construct it. Therefore, it is essential to configure the *affinity* parameter to *precomputed*.

Finally, to determine the optimal number of clusters for this method, eigengap heuristic on the Laplacian matrix is used, as suggested in (Von Luxburg, 2007). This method includes computing the eigenvalues of the Laplacian, ordering them in ascending order, and identifying the point where a gap occurs. For example, using the random walk Laplacian computed with our cosine similarity matrix, Figure 3.2 show the existence of a gap between the 3th and 4th eigenvalues, i.e., $|\lambda_4 - \lambda_3|$ is relatively large. Then, the eigengap heuristic suggests that this gap signifies the presence of 3 clusters in the dataset. Therefore, in our approach, we select 3 as a number of clusters.

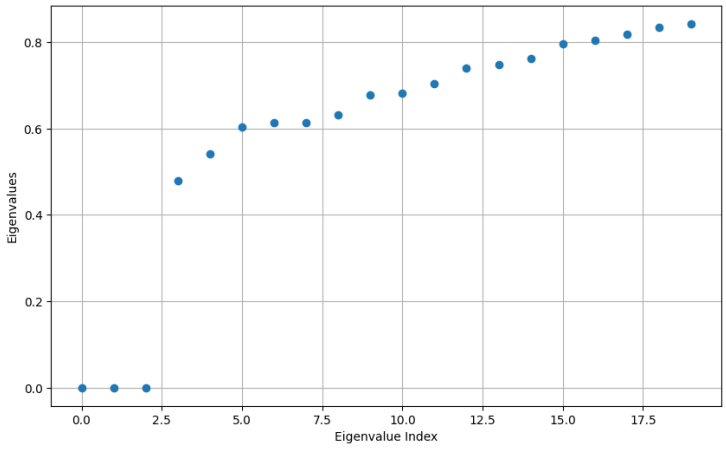


FIGURE 3.2: Eigenvalues of the random walk normalized Laplacian matrix in ascending order.

3.1.1.2 Agglomerative Clustering

Hierarchical Clustering is a clustering algorithm that builds a hierarchy of clusters. Unlike partitioning clustering algorithms, such as K-means, hierarchical clustering does not require specifying the number of clusters beforehand. Instead, it creates

a tree-like structure of nested clusters, known as a dendrogram (Gan, Ma, and Wu, 2020). There are two main approaches to hierarchical clustering:

1. *Agglomerative*: It starts with each data point as a single cluster and then, pairs of clusters are merged as one moves up the hierarchy.

2. *Divisive*: It starts with all data points in a single macro-cluster, and splits are performed recursively as one moves down the hierarchy.

The Agglomerative Clustering algorithm follows these basic steps. Initially, a dissimilarity matrix is required instead of a similarity matrix. Utilizing the cosine similarity matrix, S, can be achieved by computing $1 - S$. Subsequently, clusters with the closest similarity are merged at each level, and the dissimilarity matrix is accordingly updated. This iterative process continues until the final maximal cluster is reached or until a specific desired number of clusters is attained. To maintain consistency with the previous approach and ensure a meaningful comparison of methods, we opt for choosing 3 number of clusters.

As before, this approach is currently available in Python through scikit-learn. Since we have already generated the dissimilarity matrix, it is essential to configure the *metric* parameter to *precomputed*.

3.1.1.3 Model selection

After training both models, each one categorizes the data into three distinct groups. To determine the most suitable model for the initial stage, various evaluations are conducted.

Clustering visualization

There are multiple methods available for visualizing high-dimensional data, with t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) representing the current state of the art. Choosing between them depends on factors such as dataset size and the need to maintain the overall structure of the data. In this context, UMAP is selected for visualization due to its efficiency with large datasets and its ability to preserve the global structure of the data. UMAP aims to represent high-dimensional data in a lower-dimensional space (e.g., 2D or 3D), while retaining the inherent structure and relationships among data points. On the UMAP's Github page, you can find a comprehensive explanation of the method. According to the documentation, the method aims to create a fuzzy topological representation of the data[2]. Subsequently, the algorithm optimizes a lower-dimensional representation while preserving the topological structure of the data.

For each clustering method, a UMAP is generated to create a 2D visualization of the data points. This representation serves as a visual aid for comparing the performance of both algorithms and determining which one appears to be more effective.

[2] A fuzzy topological space extends the concept of a topological space for fuzzy sets, i.e., sets whose elements have degrees of membership.

Davies-Bouldin index

The Davies-Bouldin index (DBI) is a metric used for clustering evaluation. DBI is computed using *similarities* between each pair of clusters. The highest value is assigned to each cluster and then the index can be obtained by averaging all the clusters similarities (Liu et al., 2010). To compute the specified similarities, R_{ij}, the following procedure can be employed: calculate s_i as the average distance from each point in cluster i to the centroid of that cluster. Also, compute d_{ij} as the distance between the centroids of clusters i and j. Finally, determine the similiraties R_{ij} using the formula:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}.$$

The smaller the index is, the better the clustering result.

Calinski–Harabasz index

The Calinski–Harabasz index (CHI) is another metric used for clustering evaluation. CHI is defined as the ratio of the between-cluster separation to the within-cluster dispersion. Hence,

$$CHI = \frac{\sum_{i=1}^{k} n_i ||c_i - c||^2}{\sum_{i=1}^{k} \sum_{x \in C_i} ||x - c_i||^2} \times \frac{n-k}{k-1},$$

where k is the number of clusters, n_i is the number of points in cluster C_i, c_i is the centroid of C_i, and c is the overall centroid of the data. The second fraction acts as a normalization factor (Liu et al., 2010).

The higher the index is, the better the clustering result.

3.1.2 Second Clustering Stage

After completing model selection and opting for a specific model for the first clustering stage, three distinct clusters emerge. Chapter 4 will delve into how the initial clustering stage provides a general classification of the data, revealing the severity of the cyberattack threat. The second clustering stage focuses on medium and high threats, aiming to execute an additional clustering process for every group. This approach assists in obtaining a more detailed categorization of the most severe attacks. Now, rather than only relying on the commands feature, we will extract numerical features from the data to the execution of the clustering algorithm.

Moving forward, the methodology employed in the second clustering stage will be consistent[3] for both the medium threat group and the high threat group. Therefore, we will provide a general explanation of the approach.

Initially, we perform feature extraction to obtain a comprehensive dataset suitable for applying clustering algorithms. In Table 3.2, the extracted features used for the second clustering stage are presented along with a brief explanation for each of them. Note that the resulting dataset comprises a combination of numerical, binary, and

[3]This means that the same feature extraction, clustering algorithm, and other processes are applied uniformly to each group.

Feature	Description
Attack duration	Duration of the attack in seconds
Link download	Binary variable showing if a link is downloaded or not
chmod found	Binary variable showing if chmod command is found
Command word count	Word count of introduced commands
Password length	Character length of the used password
Protocol	Binary variable showing the used protocol, SSH or Telnet
Continent of the attacker	Categorical variable showing the continent of the attacker

TABLE 3.2: Feature extraction along with a brief explanation conducted in the second clustering stage.

categorical data types. Effectively classifying mixed data poses a significant challenge. In this stage, we have experimented with two distinct clustering approaches, and we will subsequently select the model that performs the best.

3.1.2.1 K-prototypes Clustering

Numerous clustering algorithms, such as the well-known K-means, are designed exclusively for numerical data. However, real-world datasets often comprise a mix of numerical and categorical variables, like ours. K-means encounters challenges when confronted with categorical variables because its methodology relies on mean calculations, which lack significance for categorical attributes. To address this issue, the K-modes algorithm emerges as a solution for clustering with categorical variables. Unlike K-means, K-modes utilizes a simple matching dissimilarity measure to handle categorical objects, replacing cluster means with modes. K-prototypes is a combination of K-means and K-modes, enabling the clustering of objects described by mixed numeric and categorical attributes (Huang, 1998).

We employ the K-prototypes clustering technique to analyze our data. To determine the optimal number of clusters, we apply the elbow method. This involves plotting the explained variation against the number of clusters and selecting the point where the curve exhibits an *elbow* as the optimal number of clusters.

3.1.2.2 FAMD & K-means

Another commonly used method for clustering mixed data involves a two-step approach, where cluster analysis is applied to the outcomes of a dimensionality reduction technique. A suitable dimensionality reduction technique for mixed data is Factor Analysis of Mixed Data (FAMD). In FAMD, each categorical variable undergoes one-hot encoding, while numerical features are standardized. Subsequently, each dummy variable is divided by the square root of the proportion of objects belonging to the associated category. Principal Component Analysis (PCA) is then applied to the resulting matrix. This process is analogous to PCA when dealing exclusively with continuous variables and to Multiple Correspondence Analysis (MCA) when dealing solely with categorical variables (Velden, Iodice D'Enza, and Markos, 2019).

To determine the optimal number of components for the FAMD, a scree plot is employed. This graphical representation shows the eigenvalues of the components on the y-axis against the component number on the x-axis. The objective is to detect the *elbow* of the graph and consider components before the elbow as significant, while those after it are considered less important and may be discarded.

After completing the dimensionality reduction step, we train a K-means algorithm using the output of the FAMD. Similar to the previous method, we employ the elbow method to determine the optimal number of clusters for the algorithm.

3.1.2.3 Model selection

In this scenario, model selection between K-prototypes and FAMD & K-means approaches relies on visual techniques. Previously mentioned indices such as Davies-Bouldin or Calinski-Harabasz cannot be applied because the data is of mixed nature. Additionally, in the FAMD & K-means approach, the data undergoes through a dimensionality reduction before applying K-means. Consequently, the dataset used for K-means differs from the initial dataset used for K-prototypes. Therefore, utilizing validation metrics would not be consistent, and the emphasis is placed on visualize the outcomes of the models for model evaluation.

3.1.3 Clustering prediction

In Chapter 4, we will explore how the initial clustering stage results in a broad classification of data into three distinct groups—indicating low, medium, and high threat levels. Here, the objective is to develop a machine learning model that can determine the threat group of a new cyberattack when it enters the system. This feature proves especially beneficial for the Catalonia Cybersecurity Agency. It enables the immediate categorization of data collected by the honeypot into low, medium, or high risk. This real-time classification enables cybersecurity engineers to focus on and analyze data with the highest threat level—the most dangerous category.

To achieve this goal, we developed a machine learning model exclusively utilizing the commands introduced by the attacker. To validate the model, we implemented a train/test split. Prior to the split, a rigorous command normalization was conducted, and only the unique commands are considered. This approach aims to maximize data variability by ensuring that the commands exhibit as much diversity as possible. Despite these considerations, it is important to note that some similarities persist among certain commands. The complete mitigation of all similarities is unreachable, potentially impacting the evaluation when a similar command in the test set matches one in the training set.

Because the lowest threat cluster comprises approximately 95% of the total data, the training set is inherently imbalanced. To address this issue, using the TF-IDF embeddings of the commands, a Random Forest classifier and Support Vector Machines were trained with the *class_weight* parameter set to *balanced*. This adjustment enables the algorithm to automatically modify the weights of classes in inverse proportion to their frequencies in the input data. Essentially, classes with lower frequencies receive higher weights, while classes with higher frequencies are assigned lower weights.

3.2 Predictive Analysis

The predictive analysis focuses on forecasting the number of cyberattacks on a specific honeypot. To enhance the value of our project, we will extend our predictions to forecast the overall number of cyberattacks for a particular country. For example, in predicting cyberattacks for United States, we will aggregate data from all honeypots located in United States, treating them as a unified time series for analysis. This task presents a challenge due to the absence of distinct patterns or seasonality in the resulting time series. Additionally, numerous unpredictable peaks further complicate the forecasting process. Observe Figure 3.3 for a visual illustration of this complexity.

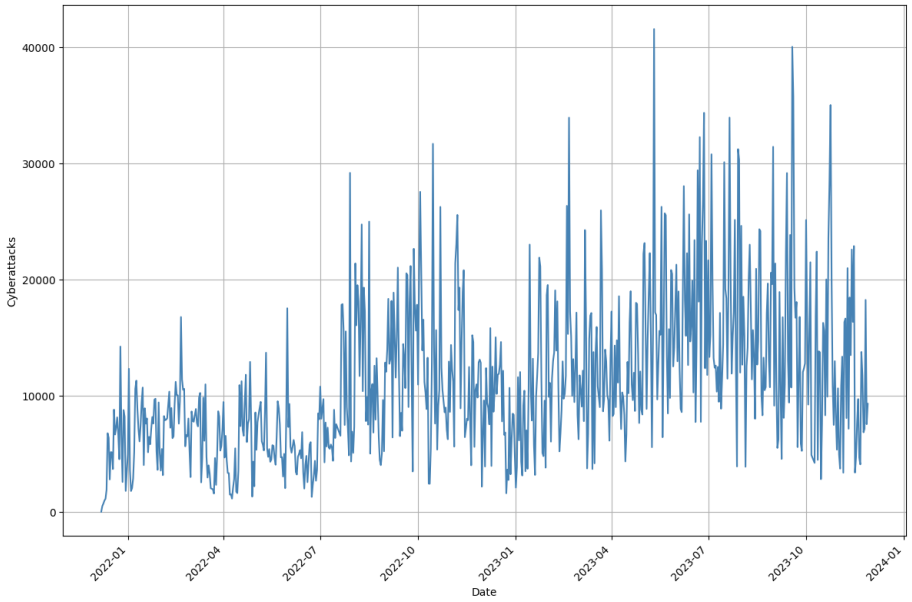


FIGURE 3.3: Daily Cyberattack Incidents in the United States from 2022 to 2024.

Numerous techniques are commonly employed for this task; let's briefly list them:

1. *Classical models:* Classical time series models are traditional statistical models that have been widely used for forecasting and analyzing time series data. These models are based on the assumption that historical patterns in the data can help predict future values. They are particularly useful for time series forecasting when the data exhibits certain patterns or trends. While classical time series models are powerful tools, it's important to note that they may not perform well in the presence of complex relationships or when dealing with large and diverse datasets.

2. *Machine Learning models:* Time series forecasting can be approached as a ML problem by framing it as a supervised learning task. In this context, we have the option to incorporate date-related features like the day of the week, month, and year, among others. The target variable is the actual value of the time series. Additionally, we can introduce past versions of the target variable (lagged values) as a new feature, enhancing the model with autocorrelation effects.

3. *Deep Leaning models:* Time series forecasting with deep learning models, particularly Long Short-Term Memory networks (LSTMs), introduces a powerful

and flexible approach to capturing intricate temporal dependencies within sequential data. LSTMs are a type of recurrent neural network (RNN) designed to effectively model long-range dependencies and capture patterns in time series data. This deep learning paradigm excels at learning from sequences of data, making it well-suited for tasks such as predicting future values in time series.

We have examined five distinct time series, each associated with a different country. Our objective is to assess the three different mentioned approaches in order to determine the most suitable one for our type of data. Spain, the United States, Singapore, Germany, and Japan are the countries that have been selected for our analysis.

3.2.1 Classical models

We utilized two classical models to predict the occurrence of cyberattacks in a particular country: ARIMA (AutoRegressive Integrated Moving Average) and Prophet. Let us provide a brief overview of each method.

ARIMA

The ARIMA model, presented by the statisticians George Box and Gwilym Jenkins in 1970 (Box et al., 2015), is a commonly employed method for forecasting time series data. It integrates autoregression, differencing, and moving averages to effectively identify and predict patterns in sequential data. The term *autoregressive* refers to the model's dependence on its own past values, *integrated* denotes the differencing of the time series to make it stationary, and *moving average* involves using the weighted average of past error terms. The ARIMA model is characterized by three main parameters, ARIMA(p, d, q), where p is the order of autoregression, d the degree of differencing, and q the order of moving average.

Mathematically, let Y_t the values of the time series at time t and B the backward shift operator, i.e., $BY_t = Y_{t-1}$. The ARIMA model can be symbolically expressed using the following equation:

$$\phi(B)(1 - B)^d Y_t = \theta(B)\epsilon_t,$$

where ϕ is the autoregressive process, θ is the process of moving average and ϵ_t is white noise.

The model can be fine-tuned by modifying the values of p and q. We used a Python function called *auto_arima* to automatically identify the optimal parameter combination. The selection of the best model was based on the Akaike Information Criterion (AIC). AIC proves valuable in comparing the effectiveness of various models, helping determine which one is better at capturing underlying data patterns while considering the simplicity of the model. Its computation involves the following formula:

$$\text{AIC} = 2k - 2ln(\hat{L}),$$

where $\hat{L}$ is the maximized value of the likelihood function and k the number of parameters in the model.

The parameter d represents the number of differentiations required for the series

to be stationary. A stationary time series is a type of time series data in which statistical properties, such as mean and variance, do not change over time. In other words, the properties of the time series remain constant across different time periods. To assess stationarity, the Dickey–Fuller test is employed, using a significance level of $\alpha = 0.05$. The null hypothesis postulates that the series is non-stationary, while the alternative hypothesis suggests that the series is stationary.

Prophet

Prophet, created by Facebook's Data Science team in 2017 (Taylor and Letham, 2018), stands out as a powerful and adaptable tool for forecasting time series data. Its strength lies in scalability, speed, and accuracy, making it well-suited for various applications. The core idea behind the algorithm involves representing time series data through a blend of trend, seasonality, and noise elements. Through this decomposition process, the algorithm excels at producing precise forecasts that effectively capture the inherent patterns within the data.

The model employs a Bayesian framework, wherein the algorithm calculates the posterior distribution of the model parameters instead of singular point estimates. This approach allows the algorithm to produce probabilistic forecasts, offering a measure of uncertainty around the central forecast.

Mathematically, it can be seen as a nonlinear regression model of the following form:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t,$$

where $g(t)$ is the trend function which models non-periodic changes in the value of the time series, $s(t)$ represents periodic changes (e.g., weekly and yearly seasonality), $h(t)$ represents the effects of holidays which occur on potentially irregular schedules over one or more days, and ϵ_t is a white noise error term.

The model employs Bayesian inference to determine the posterior distribution of its parameters. This involves utilizing a Markov Chain Monte Carlo (MCMC), which samples from the posterior distribution of the model parameters.

Model Evaluation

In both methods, the evaluation process follows the same procedure. The test set comprises the last month of the time series, while the training set consists of the preceding data. The assessment involves visualizing the actual time series alongside the predicted one. Additionally, the Mean Absolute Percentage Error (MAPE), a widely accepted metric for evaluating forecasting accuracy, is calculated using the formula below:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \times 100,$$

where A_t is the actual value and F_t is the forecast value. MAPE is expressed as a percentage, and the goal is to minimize it. A lower MAPE indicates a smaller percentage error between the predicted and actual values, suggesting better accuracy in the forecasting model.

3.2.2 Machine Learning models

A more recent approach to time series forecasting utilizes Machine Learning models. The key idea is to set the actual value of the time series as the target variable and incorporating features relevant to the problem. This allows the use of any available regression model.

For each time series, we establish an initial dataset, referred to as the baseline dataset, to serve as our starting point. This baseline dataset comprises date-related features, including the corresponding month, year, day of the year, and whether it is a working day. These features play a crucial role in enabling the regression model to identify the seasonality within the data.

Following the creation of the baseline dataset, we introduce two additional types of features. The first type involves incorporating lagged values of the target variable. At any given time t, the model can knows what happens at the time $t - i$, where $i \geq 1$. Consequently, we augment the model with i new features representing the lagged values at time $t - i$.

The second type of feature involves the integration of rolling window statistics. For example, at time t, the model can assess the mean of the preceding two days. To implement this, we generate windows of length j days and calculate statistics such as the mean, minimum, maximum, variance, and more. These features are considered as parameters to be tuned in the model, aiming to identify the optimal values for i and j that enhance the performance of the regression model.

The training pipeline proceeds as follows: for each time series, corresponding to a selected country, we perform a train/validation/test split. The test set comprises the last month of data, while the validation set includes the preceding month's data. Additionally, we generate a grid of parameters $i, j \in \{0, 5\}$ to create the corresponding dataset. Again, i represents the number of lagged features, and j denotes the window for rolling statistics features.

Using this parameter combination, we train eight distinct regression models, incorporating algorithms such as XGBoost and Support Vector Machines. Hyperparameter tuning for these models is conducted using the optuna framework on the validation set. Subsequently, we calculate MAPE on the test set and store the results. Consequently, we obtain a dictionary where, for each combination of parameters i and j, the MAPE for every regression model is recorded.

Finally, we identify the optimal parameters i and j by selecting the combination with the minimum MAPE value found. Subsequently, we construct the chosen dataset and proceed to a visual model evaluation of every used model. This evaluation involves plotting both the actual time series and the predicted time series for each regression model.

3.2.3 Deep Learning models

Deep learning has emerged as a powerful technique for solving complex problems in various domains, and time series forecasting is one area where it has shown remarkable success. Among different Deep Learning algorithms, RNN has been commonly

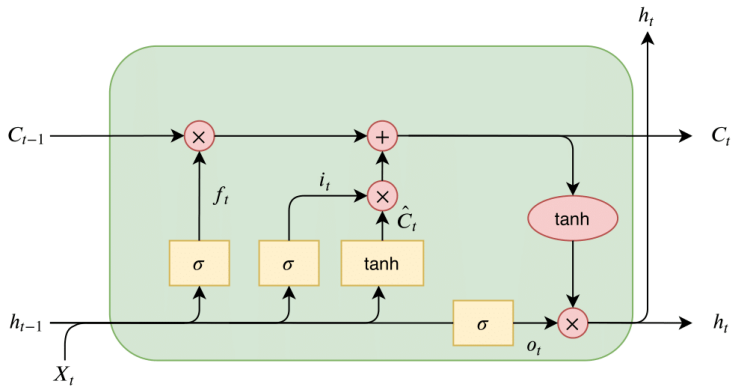


FIGURE 3.4: Architecture of an LSTM cell.

used in forecasting applications. RNNs are a type of artificial neural network designed for sequential data processing and analysis, particularly well-suited for tasks where the input data has a temporal or sequential structure. Unlike traditional feed-forward neural networks, which process input data in a fixed-size manner, RNNs have the ability to maintain a hidden state that captures information about previous inputs in the sequence.

LSTM networks are a type of RNN designed to overcome the limitations of traditional RNNs in capturing and learning long-range dependencies in sequential data. The primary challenge that LSTMs address is the vanishing gradient problem, which occurs during the training of RNNs. The vanishing gradient problem makes it difficult for traditional RNNs to learn and remember information from distant time steps, leading to a loss of context in long sequences. LSTM networks include memory cells that can maintain information over long periods, making them particularly effective for processing and learning from sequential data, such as time series, natural language, and speech.

More precisely, the key innovation in LSTMs is the incorporation of a memory cell with three crucial gates—input, forget, and output—to control the flow of information. The input gate, i_t, serves as a regulator, deciding which information merits storage in the memory cell, $\hat{C}_t$, while the forget gate, f_t, determines what elements should be discarded. Simultaneously, the output gate, o_t governs the generation of the subsequent hidden state, h_t. The cell state update, C_t, a critical aspect of the LSTM, involves the addition and removal of information from the memory cell. This update process is executed by combining the decisions made by the input and forget gates. Consequently, the hidden state, representing the output of the LSTM cell, encapsulates the pertinent information for the subsequent time step, further enhancing the network's ability to comprehend and learn intricate sequential patterns. Figure 3.4 illustrates a diagram of an LSTM cell.

In this situation, we harness the capabilities of these networks for our specific task. The training pipeline proceeds as follows: for each time series, corresponding to a selected country, we perform a train/validation/test split. As before, the test set comprises the last month of data, while the validation set includes the preceding month's data. To construct the dataset, we determine the optimal combination of the parameters i, j, described in the preceding section. After establishing the dataset for each time series, it is essential to format the data appropriately for the intended

model. The data is segmented into sequences of tensors, each with a length of 3, and the label is assigned to be the subsequent value in the series. Additionally, the data undergoes scaling to ensure optimal model performance. Once the data is structured in a sequential format, we proceed to build a straightforward LSTM model. This model consists of two LSTM layers, each comprising 64 neurons, along with dropout for regularization. Following the LSTM layers, a dense layer with 32 neurons is incorporated, culminating in an output layer.

Chapter 4

Results

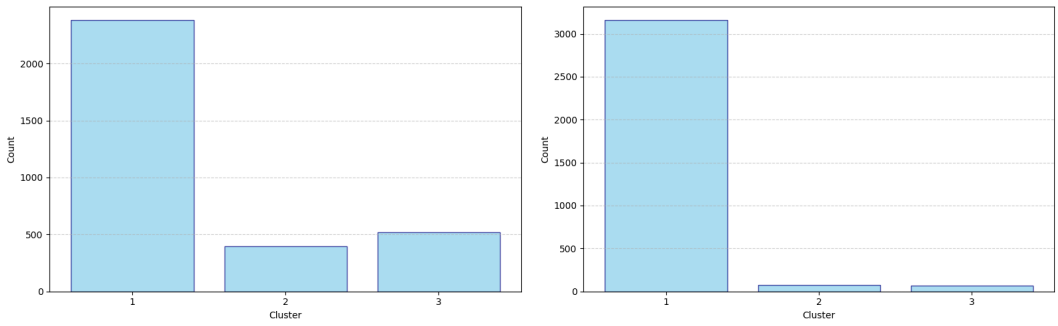
This chapter intends to replicate the methodology explained in Chapter 3 and investigate the results obtained. We will begin by delving into the cluster analysis.

4.1 Cluster Analysis

4.1.1 First Clustering Stage

In the first clustering stage, we explored two distinct methods. Spectral clustering was applied to the similarity matrix, while Agglomerative clustering was tested on the dissimilarity matrix, both generated through the embeddings commands given by the TF-IDF. As mentioned in the preceding chapter, to determine the best model, our attention will be directed towards specific visualization techniques, along with the utilization of the Davies-Bouldin index and the Calinski–Harabasz index.

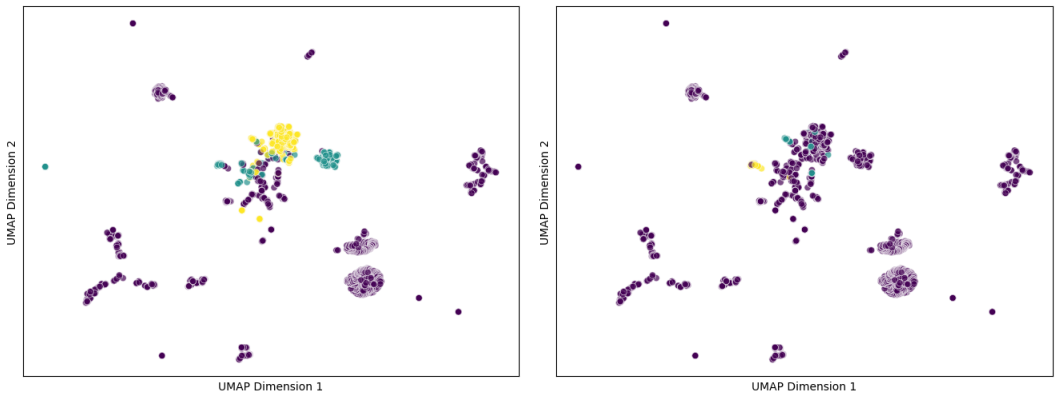
Let us start by observing the distribution of records within each cluster. Figure 4.1 illustrates this distribution for each approach. It is noteworthy that a significant proportion of data aligns in a single large cluster, encompassing approximately 70% of the data in the Spectral method and 95% in the agglomerative technique. It seems that Agglomerative clustering tends to be more conservative in assigning records to different clusters, whereas Spectral clustering tends to be more lenient.



(A) Spectral Clustering (B) Agglomerative Clustering

FIGURE 4.1: Cluster distribution for each model.

Continuing the discussion on clustering visualization, Figure 4.2 displays a 2D UMAP representation. This visualization presents the data in two dimensions, with each cluster distinguished by colors, providing a clear insight into the clustering process. Notably, in the Agglomerative clustering, the predominant color is purple, representing the first cluster, which dominates the plot, with only a few points in



(A) Spectral Clustering (B) Agglomerative Clustering

FIGURE 4.2: 2D UMAP visualization illustrating distinct clusters. Cluster 1 is denoted by purple points, cluster 2 by green points, and cluster 3 by yellow points.

Metric/Algortihm	Spectral Clustering	Agglomerative Clustering
Davies-Bouldin	3.09	**2.95**
Calinski–Harabasz	**221.17**	36.16

TABLE 4.1: A comparison of evaluation metrics for Spectral and Agglomerative clustering.

the center belonging to a different cluster. In contrast, the Spectral cluster reveals a concentrated ball of yellow points (cluster 3) and some green points (cluster 2) that appear to transition from purple to yellow, despite the overall dominance of purple in the plot. This suggests, for the Spectral clustering, the existence of a prominent cluster, cluster 1, and a distinctly separate cluster, cluster 3. Additionally, there is the identification of another cluster positioned between the two aforementioned clusters.

Table 4.1 displays the scores of the Davies-Bouldin and Calinski–Harabasz indices. While the Davies-Bouldin indices are nearly identical, the Calinski–Harabasz index is six times superior for Spectral clustering. This indicates a more favorable ratio of between-cluster separation to within-cluster dispersion. Hence, Spectral clustering exhibits superior results compared to Agglomerative clustering. The indices demonstrate this advantage, and the UMAP representation provides insights into the underlying patterns. Notably, there appears to be a predominant cluster (cluster 1) that attracts a majority of the data. Additionally, there is a distinct and well-separated cluster (cluster 3), along with the presence of an intermediate cluster.

By examining each cluster more closely, we can discern specific command patterns within each. Cluster 1, the most prevalent group, primarily comprises commands focused on gathering system information. Cluster 3 encompasses commands associated with downloading and executing files with malicious software. Cluster 2 falls between these extremes, featuring commands related to package installation, permission changes, and similar actions. Let us precisely define these groups:

1. *Cluster 1*: It represents the **lowest threat**. It consists of commands primarily aimed at collecting system information and conducting system scans.

2. *Cluster 2*: This group is considered to be at an **intermediate level**. Commands include not only acquiring system information but also involve installing some packages and changing file permissions.

3. *Cluster 3*: This poses the **highest threat**. The commands within this category involve the downloading and execution of files with root permissions, potentially introducing malicious software.

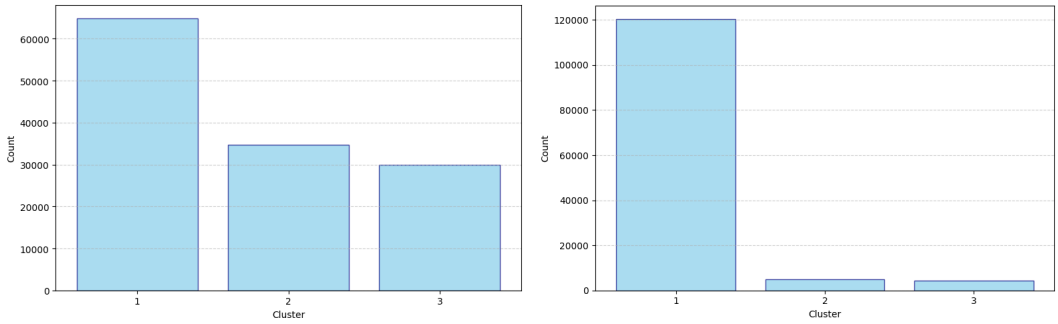
4.1.2 Second Clustering Stage

In Chapter 3, we examined two specific approaches for further classify data within the intermediate and high threat groups: K-prototypes and FAMD & K-means. Our emphasis on these groups stems from their elevated threat levels, making them particularly crucial for cybersecurity engineers. It is essential to note that feature extraction was carried out to enhance the clustering process, and the details can be found in Table 3.2. Let us begin by discussing the outcomes achieved in the intermediate cluster.

4.1.2.1 Intermediate threat

Let us explore the results from applying the K-prototypes and FAMD & K-means methods within this group. Initially, we need to identify the ideal number of clusters, a step initiated by the K-prototypes algorithm using the elbow method. As illustrated in Figure B.1, the optimal number of clusters is determined to be 3. For consistency and result comparison, we also opt for 3 clusters when employing the FAMD & K-means algorithms.

After determining the optimal number of clusters for both methods, we can examine how records are distributed within each cluster. Figure 4.3 displays this distribution for each method. It appears that the first method exhibits greater variability and is more flexible in distributing records across clusters, whereas the second method confines about 93% of the data to a single cluster. This suggests that certain features may dominate the clusters, preventing effective separation by the method.



(A) K-prototypes (B) FAMD & K-means

FIGURE 4.3: Cluster distribution for each model.

Figures B.2 and B.3 display six charts containing detailed information for each cluster in both the K-prototypes and the FAMD & K-means methods. In the last approach, the dominance of binary features (chmod found, link download, and protocol used) in the clustering process is evident. This dominance suggests potential

bias in the results, raising concerns that important patterns in other features might be overlooked. Furthermore, this oversimplification of the data structure could result in a loss of information and nuance present in other features. Hence, K-prototypes is selected for this scenario. Now, let us provide comments on the results we have obtained.

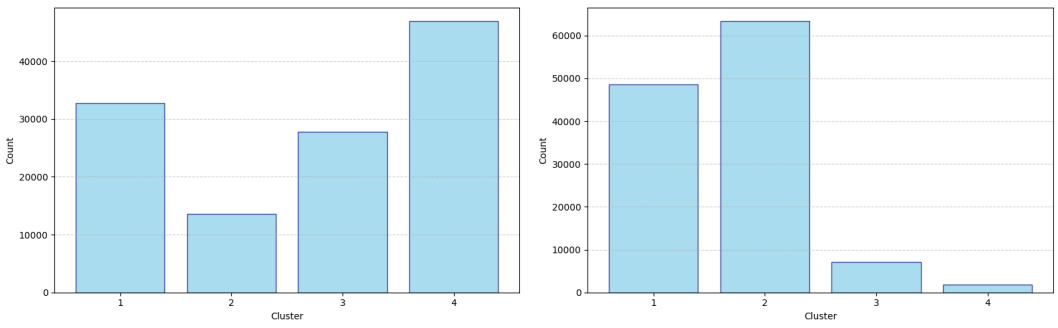
Notice that the application of K-prototypes results in the creation of three distinct groups. Among these, group 2 exhibits shorter attack durations compared to the other two. Conversely, cluster 1 appears to be more dangerous, as it not only has longer-lasting attacks but also features the highest average number of commands introduced. Additionally, cluster 1 stands out with the highest count of downloaded and executed links, indicating a higher level of threat. In summary, cluster 1 appears to be the most dangerous within this context, while cluster 2 is relatively less threatening.

Upon examining Figure B.4, which illustrates the distribution of clusters across continents indicating the origin of the attacks, it becomes evident that cluster 2, the less threatening, is predominantly associated with Europe, whereas the others are linked to Asia. This discrepancy may be attributed to the stricter cybersecurity laws in Europe, making it more challenging for cyberattackers to launch attacks without facing consequences. Conversely, Asia's leniency in this regard is reflected in the observed behavior of the clusters.

4.1.2.2 High threat

Let us examine the outcomes obtained by applying these methods to the high-threat group. To determine the optimal number of clusters, we utilize the elbow method with the K-prototypes algorithm. As depicted in Figure B.5, the optimal parameter is found to be 4. As before, to maintain consistency and facilitate result comparison, we also choose 4 clusters when utilizing the FAMD & K-means algorithm.

Let us explore how the records are distributed within each cluster. Figure 4.4 displays this distribution for each method. In this case the behaviour is similar for both methods. The pattern suggests that approximately half of the data is grouped within cluster 4 for K-prototypes and cluster 2 for FAMD & K-means. Following this, there is another substantial cluster, cluster 1, the same in both cases, along with two smaller clusters.



(A) K-prototypes (B) FAMD & K-means

FIGURE 4.4: Cluster distribution for each model.

Figures B.6 and B.7 depict six charts illustrating information for each cluster in both the K-prototypes and the FAMD & K-means methods. Despite apparent differences in results, the conclusions drawn are quite similar.

In the case of K-prototypes, cluster 4, the largest cluster, comprises the least threatening group. This group exhibits the shortest attack duration, introduces fewer commands, and, notably, lacks instances of the *chmod* command. The absence of this command is significant as it indicates a lack of execution with permissions. On the other hand, clusters 1 and 3 are identified as more dangerous, characterized by longer attack durations, a higher number of introduced commands, and a systematic downloading and execution of files in the system. These clusters represent the most dangerous groups.

Contrastingly, in the FAMD & K-means approach, cluster 2, the largest cluster, stands out as the one with the longest attack duration, a higher command count, and a systematic downloading and execution of files in the system. Furthermore, cluster 1 is identified as the less threatening group in this approach, exhibiting a shorter attack duration and no instances of executing files using the *chmod* command. Therefore, a connection is evident between the two approaches. Specifically, cluster 4 in K-prototypes aligns with cluster 1 in FAMD & K-means, while clusters 1 and 3 in K-prototypes correspond to cluster 2 in FAMD & K-means.

Moreover, when examining Figures B.8 and B.9, which depict the distribution of clusters across continents (indicating the origin of the attacks) for K-prototypes and FAMD & K-means, it is evident that in K-prototypes, cluster 4 is primarily associated with Europe, coinciding with cluster 1 in FAMD & K-means, representing the least threatening group in both methodologies. Moreover, clusters 1 and 3 in K-prototypes, which constitute the more threatening groups, are primarily associated with North America, Asia and Europe. This aligns with cluster 2, identified as the most threatening in FAMD & K-means.

Therefore, despite the differences in how groups are delineated, the conclusions remain consistent. In both cases, distinct threatening groups emerge, with the less threatening group governed by Europe and the more threatening ones by North America and Asia, along with some instances from Europe.

4.1.3 Clustering prediction

To conclude the Cluster Analysis results section, let us examine the outcomes of machine learning clustering predictions, using as a target, the results given by first clustering stage. As detailed in Chapter 3, we tested both Random Forest and Support Vector Machines. Table 4.2 displays the F1-score, recall, and precision for each model assessed on the test set. Examining the metrics, it is evident that Support Vector Machines consistently outperform Random Forest.

Nevertheless, it is important to note that the model's impressive performance might be obscured by the quality of the train/test split. Despite the introduction of a specific command normalization to ensure working with unique commands, there is a possibility that some commands may remain similar, potentially resulting in the consequences of data leakage in the test set.

Metric	Random Forest	Support Vector Machines
F1-Score	0.952	0.960
Recall	0.943	0.954
Precision	0.965	0.966

TABLE 4.2: F1-score, recall and precision comparison between Random Forest and Support Vector Machines.

4.2 Predictive Analysis

Let us now examine the results obtained from exploring the time series forecasting problem with three distinct approaches: the classical approach, the machine learning approach, and the deep learning approach. Due to space limitations, our visual evaluation will exclusively concentrate on time series data from the United States. If you are interested in the detailed results pertaining to the time series data from Spain, Singapore, Germany, and Japan, please consult Appendix C.

4.2.1 Classical models

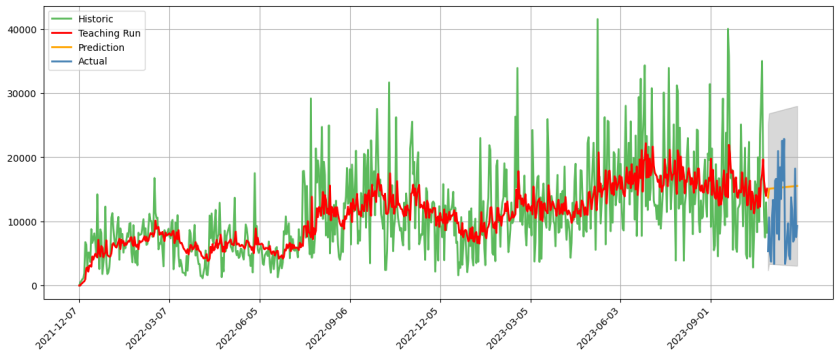
As previously discussed in Chapter 3, we employed ARIMA and Prophet methodologies to forecast the quantity of cyberattacks expected in each country for the upcoming month. Remember that the time series data from the United States was the only one that exhibited non-stationarity. Consequently, in this case, we need to set the differencing parameter d to 1 in the ARIMA model, since the series was stationary after one order of differencing.

In Figure 4.5 and Figure 4.6, we illustrate the outcomes generated by ARIMA and Prophet, respectively, for the United States' time series. It is evident that both methods yield a prediction in the form of a constant line, resulting in a notably uninformative and unrealistic forecast. Certainly, these traditional methods consistently produce such outcomes across all tested time series, as detailed in Appendix C. Consequently, for this specific task, these classical models exhibit poor performance and prove to be unhelpful.

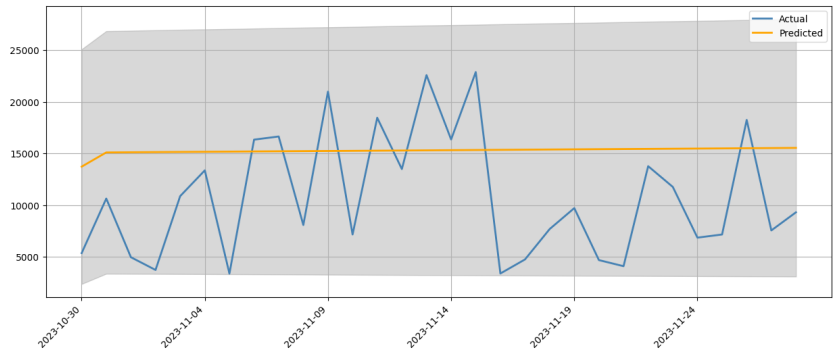
Table 4.3 presents a comparison of the MAPE metric for both models across all the studied time series.

Country	ARIMA	Prophet
USA	105.96	128.45
Spain	121.49	110.63
Singapore	146.16	176.55
Germany	175.97	198.15
Japan	129.22	160.67

TABLE 4.3: Comparison of the MAPE metric for both models across all the studied countries.

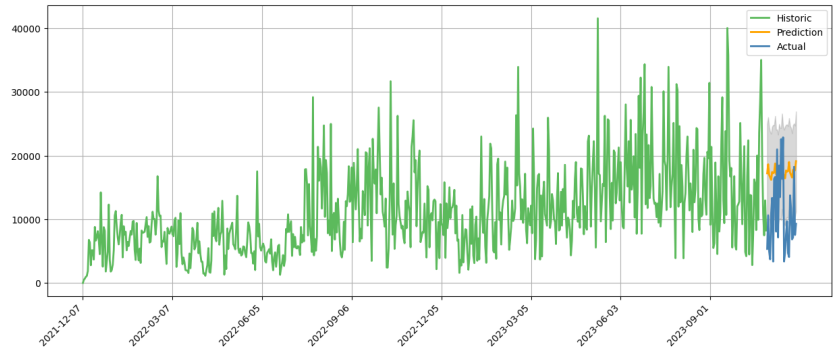


(A) Actual Time series, Trained values and Predicted Values.

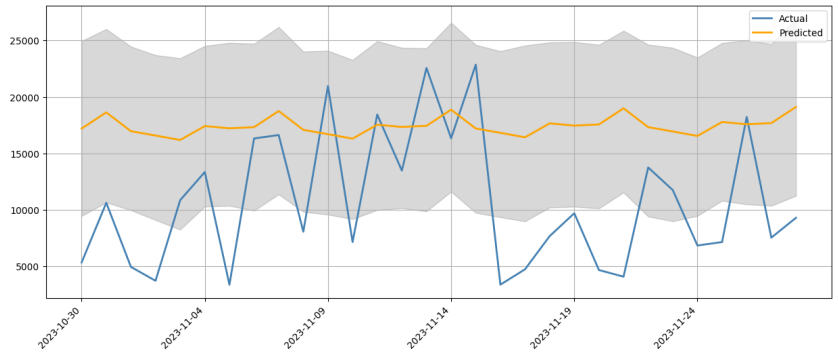


(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

FIGURE 4.5: Analysis for the ARIMA model on the United States times series.



(A) Actual Time series and Predicted Values.



(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

FIGURE 4.6: Analysis for the Prophet model on the United States times series.

4.2.2 Machine Learning models

Following the disappointing outcomes from classical models, we are now placing our expectations on a different approach. As outlined in the preceding chapter, this method initially focuses on identifying the optimal combination of parameters, denoted as i and j, where i represents the number of lagged features, and j denotes the window for rolling statistics features. For the United States time series, the selected best combination is $i = 1$ and $j = 5$. This indicates that a feature corresponding to the first lagged value is introduced into the dataset, and rolling statistics for the first 5 windows are computed and incorporated as new features.

While various regression models have been examined, XGBoost and Support Vector Machine consistently demonstrate exceptional performance compared to other models in the majority of cases. In Figure 4.7 and Figure 4.8, we illustrate the outcomes generated by XGBoost and Support Vector Machines, respectively, for the United States' time series. Note that the predictions excel beyond those generated by the classical approach. What's particularly noteworthy is that, while the models may not accurately predict the peaks, they are proficient at forecasting the overall trend of the series, which is really interesting in the cybersecurity field.

Table 4.4 presents a comparison of the MAPE metric for both models (XGBoost and SVM) across all the studied time series. We observed a overall enhancement compared to the classical approach.

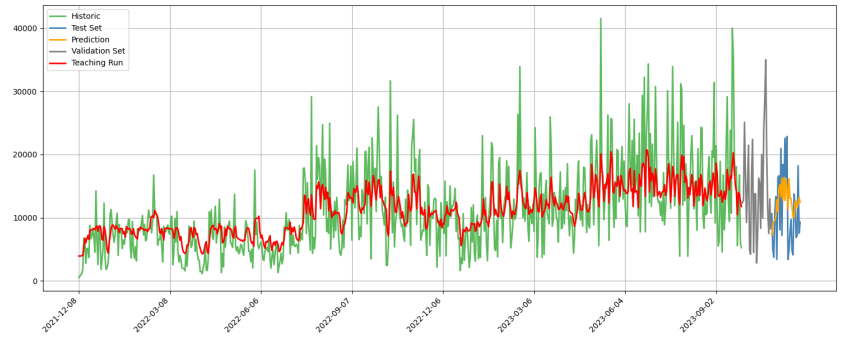
Country	XGBoost	SVM
USA	76.25	66.26
Spain	137.86	65.84
Singapore	134.62	112.11
Germany	127.59	87.45
Japan	85.56	59.94

TABLE 4.4: Comparison of the MAPE metric for both models across all the studied countries.

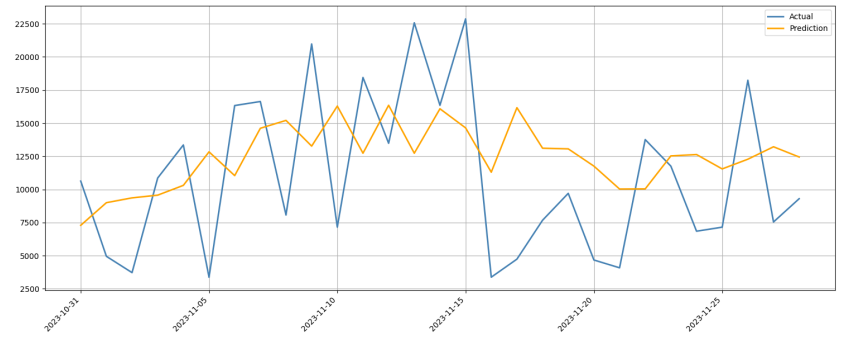
4.2.3 Deep Learning models

We will now investigate the LSTM approach to determine if it can enhance the Machine Learning approach. As in the previous analysis, the initial step involves identifying the optimal combination of parameters, represented by i and j, where i indicates the number of lagged features, and j the window for rolling statistics features. While one might anticipate that lagged features are unnecessary since LSTM inherently handles such features, our examination revealed that incorporating lagged features enhances model performance, particularly for Singapore and Germany. Refer to Appendix C for details.

For the United States time series, the selected best combination is $i = 0$ and $j = 5$. This implies that no lagged features are introduced, and rolling statistics for the initial 5 windows are calculated and integrated as new features. In Figure 4.9, we present the results produced by the LSTM model for the United States time series. It is noticeable that the LSTM predictions appear to be even more accurate than those

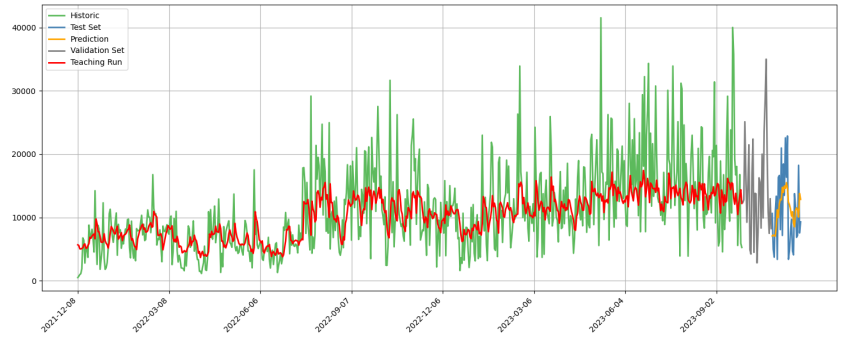


(A) Actual Time series, Trained values and Predicted Values.

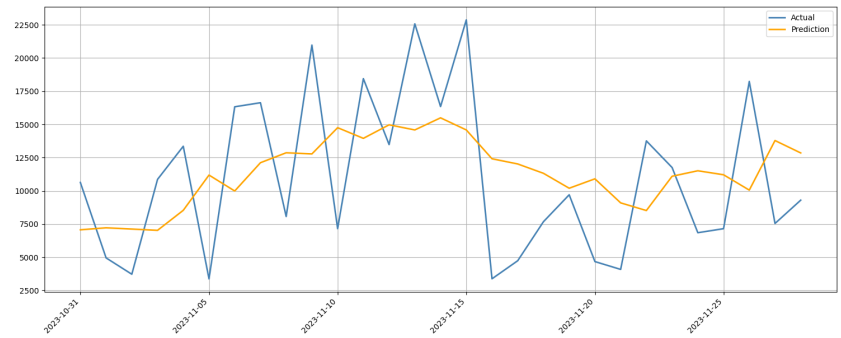


(B) Actual vs. Predicted (Test set).

FIGURE 4.7: Analysis for the XGBoost model on the United States times series.

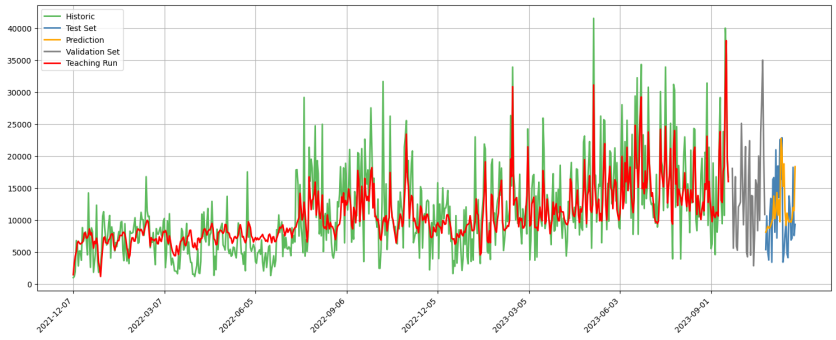


(A) Actual Time series, Trained values and Predicted Values.

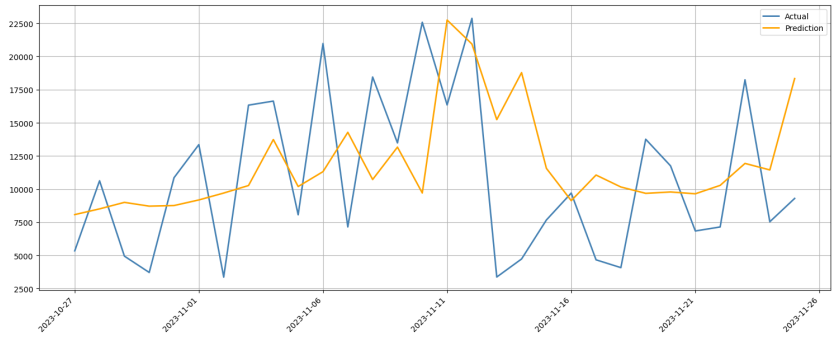


(B) Actual vs. Predicted (Test set).

FIGURE 4.8: Analysis for the SVM model on the United States times series.



(A) Actual Time series, Trained values and Predicted Values.



(B) Actual vs. Predicted (Test set).

FIGURE 4.9: Analysis for the LSTM model on the United States times series.

generated by XGBoost and SVM. The model not only predicts the trend of the series but also demonstrates the ability to accurately predict certain peaks.

However, this approach does not consistently surpass the Machine Learning approach. In certain time series instances, the results are remarkably similar, and there is not a distinct advantage in favor of using this model over others. What holds true is that this model stands as a genuine competitor to the preceding one. It is likely that with access to more than two years of data, this model would exhibit superior performance, as DL models typically excel with a substantial amount of data.

Table 4.5 presents a comparison of the MAPE metric for all the previous models inclduing the LSTM approach. We observe that SVM performs the best in this metric; however, a visual assessment reveals that XGBoost and LSTM are also formidable contenders.

Country	ARIMA	Prophet	XGBoost	SVM	LSTM
USA	105.96	128.45	76.25	66.26	73.51
Spain	121.49	110.63	137.86	65.84	114.27
Singapore	146.16	176.55	134.62	112.11	167.57
Germany	175.97	198.15	127.59	87.45	193.20
Japan	129.22	160.67	85.56	59.94	138.07

TABLE 4.5: Comparison of the MAPE metric across all studied approaches and countries.

Chapter 5

Conclusions

In this master's thesis, we examined cyberattack data collected from a globally distributed network of honeypots. Our main goals were to conduct an Exploratory Data Analysis (EDA) to gain insights into the data, employ clustering analysis to identify patterns indicative of cyberattacks, and develop predictive models to forecast the future number of cyberattacks on a honeypot. The aim was to strengthen our understanding of current cyberthreats and introduce the use of machine learning and deep learning techniques to improve cyberdefense measures.

The clustering analysis revealed distinct patterns within the cyberattack data. In the initial clustering phase, three distinct groups emerged, ranging from low to high threat levels. The lowest level involved commands primarily focused on gathering system information, while the highest level encompassed commands that download and execute files with root permissions, posing a potential risk of introducing malicious software. Additionally, a detailed classification was conducted for the intermediate and high threat levels, revealing a further hierarchy of threats within these identified groups.

In the predictive analysis, we investigate different forecasting models to predict the number of cyberattacks on a specific honeypot. Our exploration involves three distinct approaches: classical models, machine learning models, and deep learning models. The experiments indicate that both machine learning and deep learning models surpass the performance of classical ones. The top-performing models successfully forecast the overall trend of the examined time series, yet encounter challenges in accurately predicting the peaks of the series.

The integration of emerging technologies, such as artificial intelligence and machine learning, holds immense potential for advancing the field of cybersecurity. Future work could explore the practical application of the developed forecasting algorithm in real-time cybersecurity scenarios. Implementing the model in a live environment would provide valuable insights into its real-world effectiveness and operational feasibility. Furthermore, upcoming research will concentrate on enhancing the architecture of the deep learning model. This is necessary as our current model is relatively simple and lacks complexity.

Appendix A

Features explanation and EDA

This appendix provides comprehensive information on the 48 features present in the AIDE data and charts related to the EDA conducted in section 2.4.

The available features are categorized into three main sections, starting with general fields in the first section, which are described in detail in Table A.1. The second section contains fields related to the attackers, observe Table A.2. Finally, the last section contains the same information than Table A.2, but about honeypots instead of attackers.

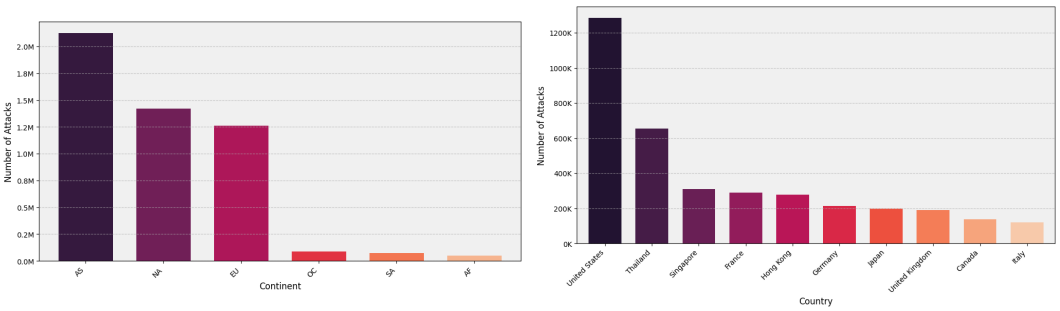
Features	**Explanation**
Commands	Unix commands executed by the attacker that are fully emulated
UnknownCommands	Commands attempted by the attacker that are not fully emulated
Loggedin	Username/password combination that successfully logged in
Credentials	Username/password combination that did not successfully log in
_id	Document identifier
_index	Name of the document containing index
_type	Document type; default is _doc
@timestamp	Document index date/timestamp in ISO8601 format in UTC time zone
Session	Unique session identifier
EndTime	Timestamp of the end of the session/attack
StartTime	Timestamp of the start of the session/attack
HostIP	Attacked honeypot IP
HostPort	Attacked honeypot port
PeerIP	Attacker honeypot IP
PeerPort	Attacker honeypot port
Protocol	SSH or Telnet
Version	Version of the SSH library used by the attacker
Urls	List of URLs from where malware files were downloaded
Hashes	Malware file hashes
Tags	OpenSearch additional information

TABLE A.1: General features along with a brief explanation.

Features	**Explanation**
Geoip.as_org	Host provider e.g. Elastic, NV
Geoip.asn	Autonomous System Number e.g., 98765
Geoip.city_name	Name of the city e.g., Seattle
Geoip.continent_code	Continent code e.g., EU
Geoip.country_name	Name of the country e.g., United States
Geoip.dma_code	DMA code e.g., 819
Geoip.ip	The IP e.g., 12.34.56.78
Geoip.latitude	Latitude e.g., 47.6062
Geoip.location	Location e.g., {"lat": 47.6062, "lon": -122.3321}
Geoip.longitude	Longitude e.g., -122.3321.
Geoip.postal_code	Postal code e.g., 08330
Geoip.region_code	Region code e.g., WA
Geoip.region_name	Name of the region e.g., Washington
Geoip.timezone	Time zone e.g., America/Los_Angeles.

TABLE A.2: Features realted to the attackers along with a brief explanation.

In relation to Exploratory Data Analysis (EDA), Figure A.1 displays two bar charts illustrating the number of cyberattacks categorized by continent and country destinations. The bar chart in Figure A.2 illustrates the number of cyberattacks received by the United States, categorized by continent. Figure A.3 displays two pie charts illustrating the attackers' use of protocols and SSH versions. Figure A.4 presents two histograms depicting the distribution of attack duration. Moving on to Figure A.5, it showcases a pie chart revealing the most frequently used commands introduced by the attackers. Lastly, Figure A.6 features a word cloud chart displaying the most commonly used successful passwords for system access.



(A) Count per continent.

(B) Count per country.

FIGURE A.1: Number of cyberattacks by destination region.

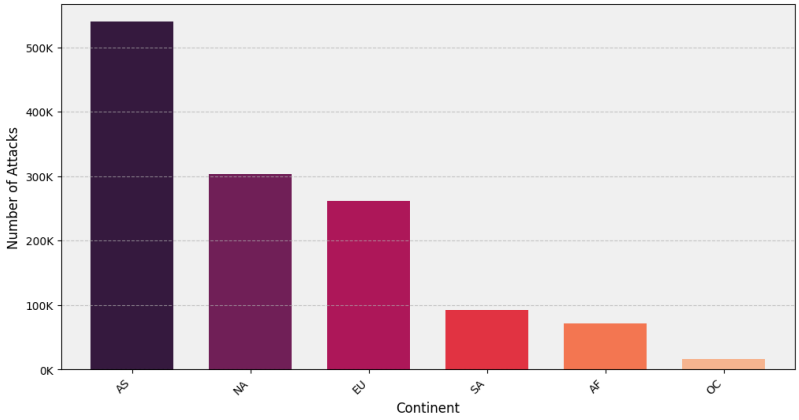
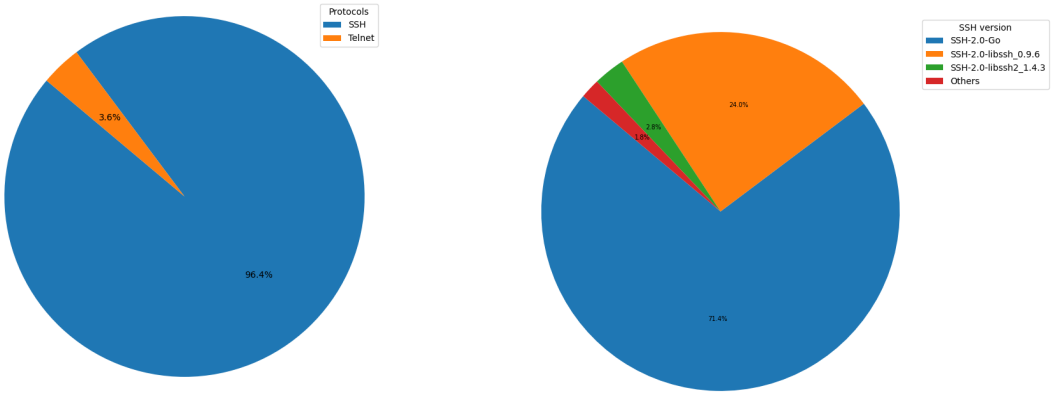


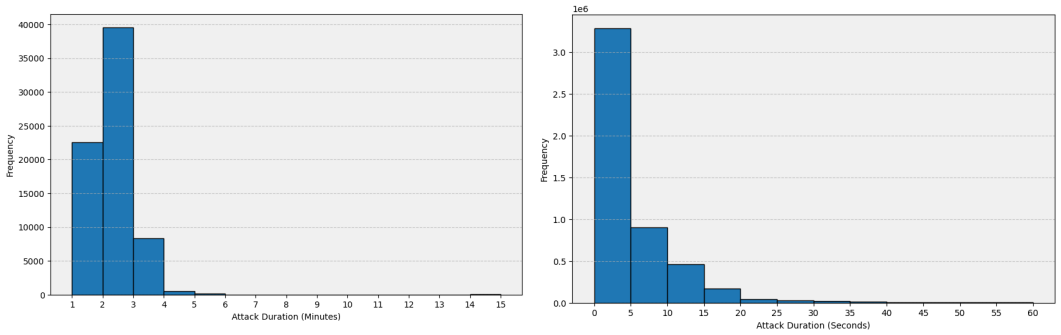
FIGURE A.2: United States suffered cyberattacks per continent.



(A) Distribution of the used protocols, SSH vs Telnet.

(B) Distribution of the used SSH versions.

FIGURE A.3: Comparison between used protocols and used SSH versions.



(A) Frequency of the duration of the attacks (in minutes).

(B) Frequency of the duration of the attacks in the first minute (in seconds)

FIGURE A.4: Histograms with attack duration information.

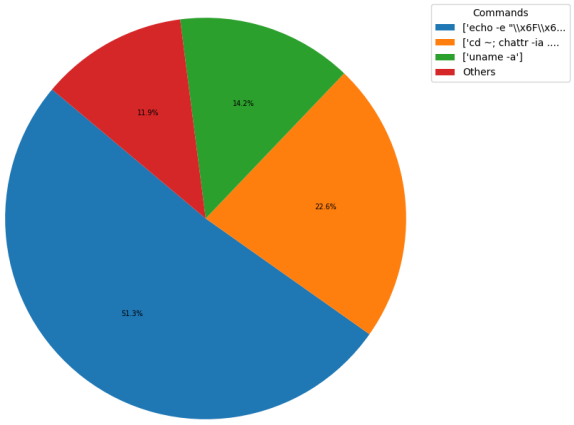


FIGURE A.5: Most frequently used commands by cyberattackers.

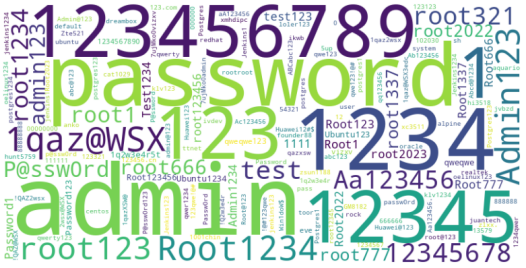


FIGURE A.6: Most frequently used passwords by cyberattackers.

Appendix B

Cluster Analysis extended Results

This appendix includes charts that depict the outcomes of the second-stage clustering analysis.

Figure B.1 displays the elbow method applied to the K-prototypes within the intermediate threat group. The data presented in Figures B.2 and B.3 provides details on the clustering algorithms carried out within the intermediate threat group for two different approaches: K-prototypes and FAMD & K-means. Figure B.4 illustrates the distribution of clusters given by K-prototypes across continents (indicating the origin of the attacks) for the intermediate threat group.

Figure B.5 displays the elbow method applied to the K-prototypes within the high threat group. The data presented in Figures B.6 and B.7 provides details on the clustering algorithms carried out within the high threat group for two different approaches: K-prototypes and FAMD & K-means. Finally, Figures B.8 and B.9 illustrates the distribution of clusters given by K-prototypes and FAMD & K-means respectively, across continents (indicating the origin of the attacks) for the high threat group.

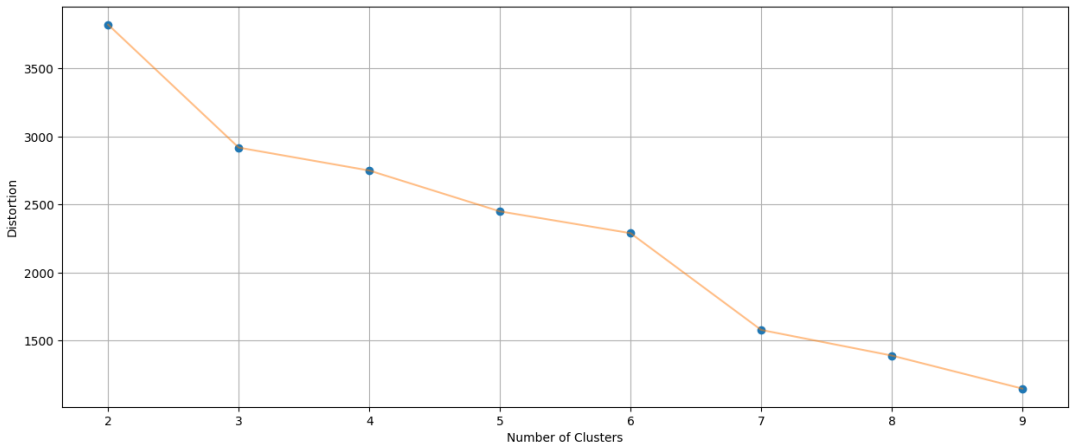


FIGURE B.1: Elbow method applied to K-prototypes within the intermediate threat group.

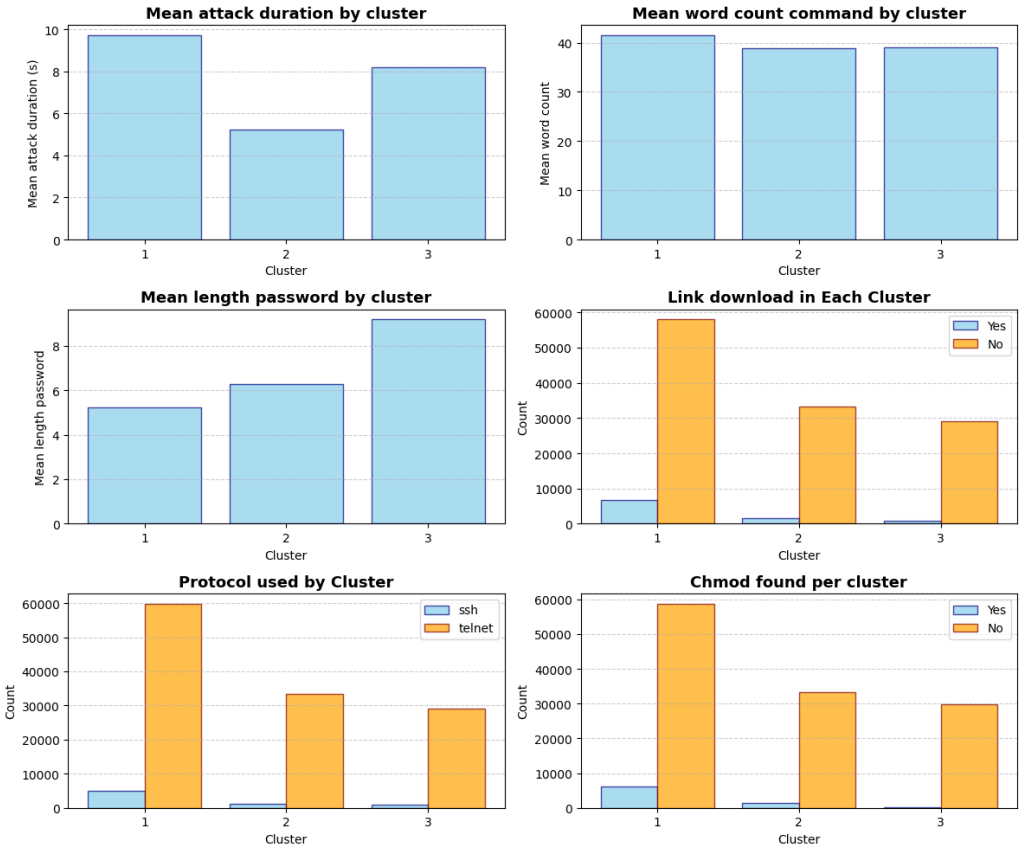


FIGURE B.2: Information about clusters for the intermediate threat group using K-prototypes approach.

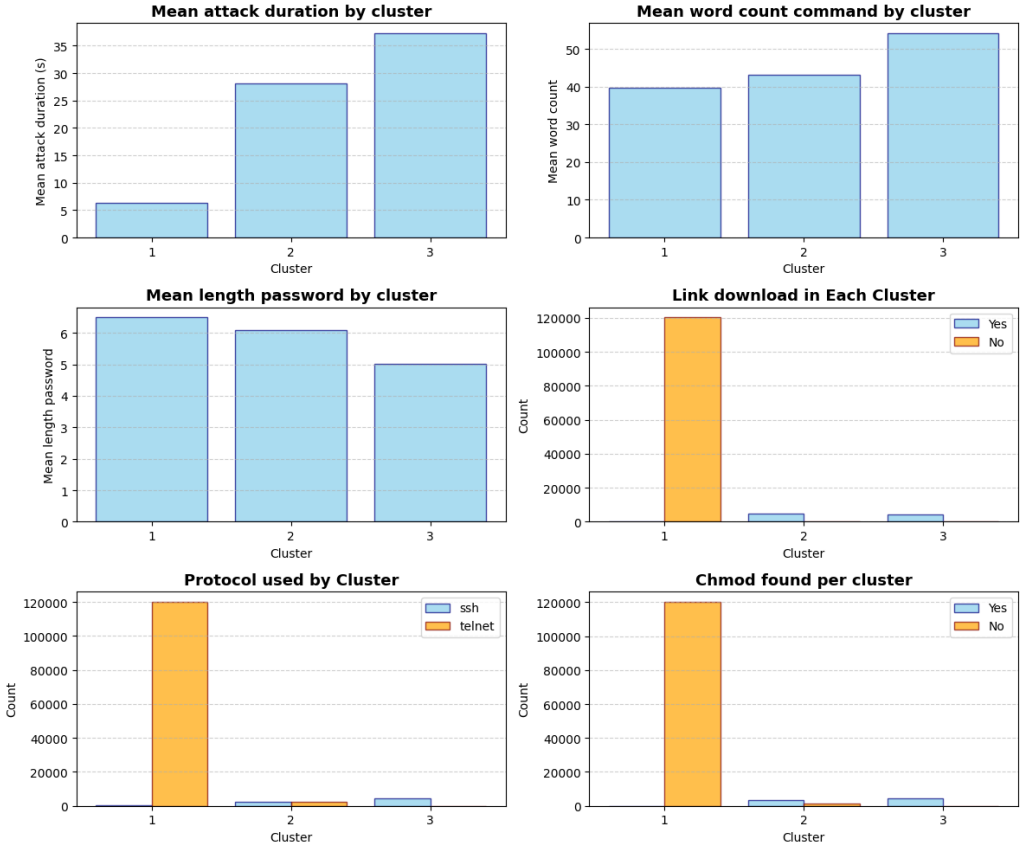


FIGURE B.3: Information about clusters for the intermediate threat group using FAMD & K-means approach.

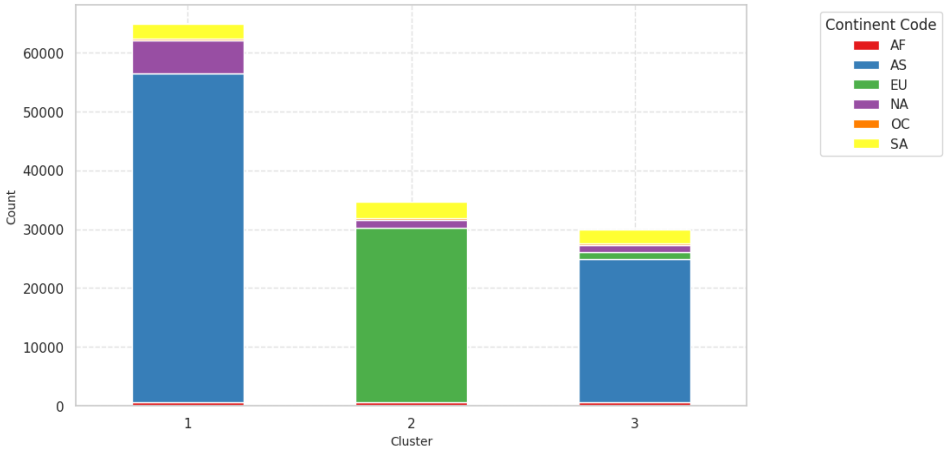


FIGURE B.4: Distribution of clusters, given by K-prototypes, across continents (indicating the origin of the attacks) for the intermediate threat group.

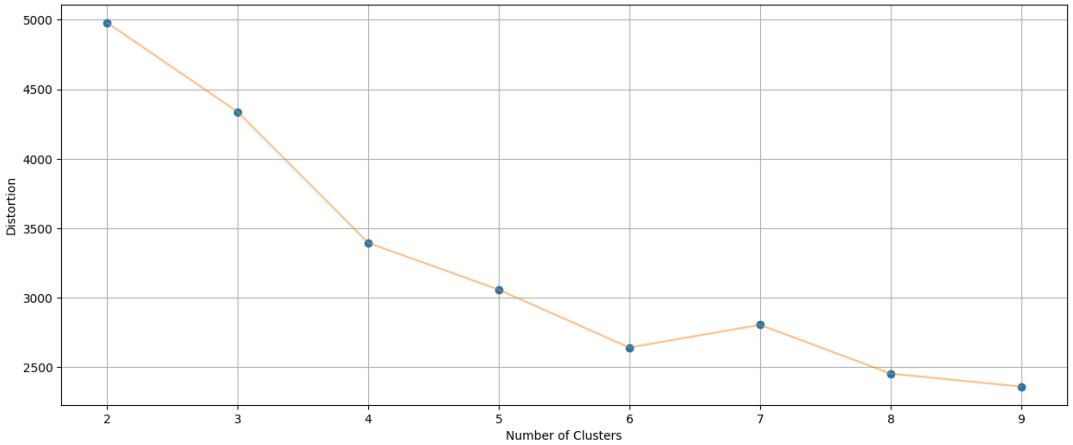


FIGURE B.5: Elbow method applied to K-prototypes within the high threat group.

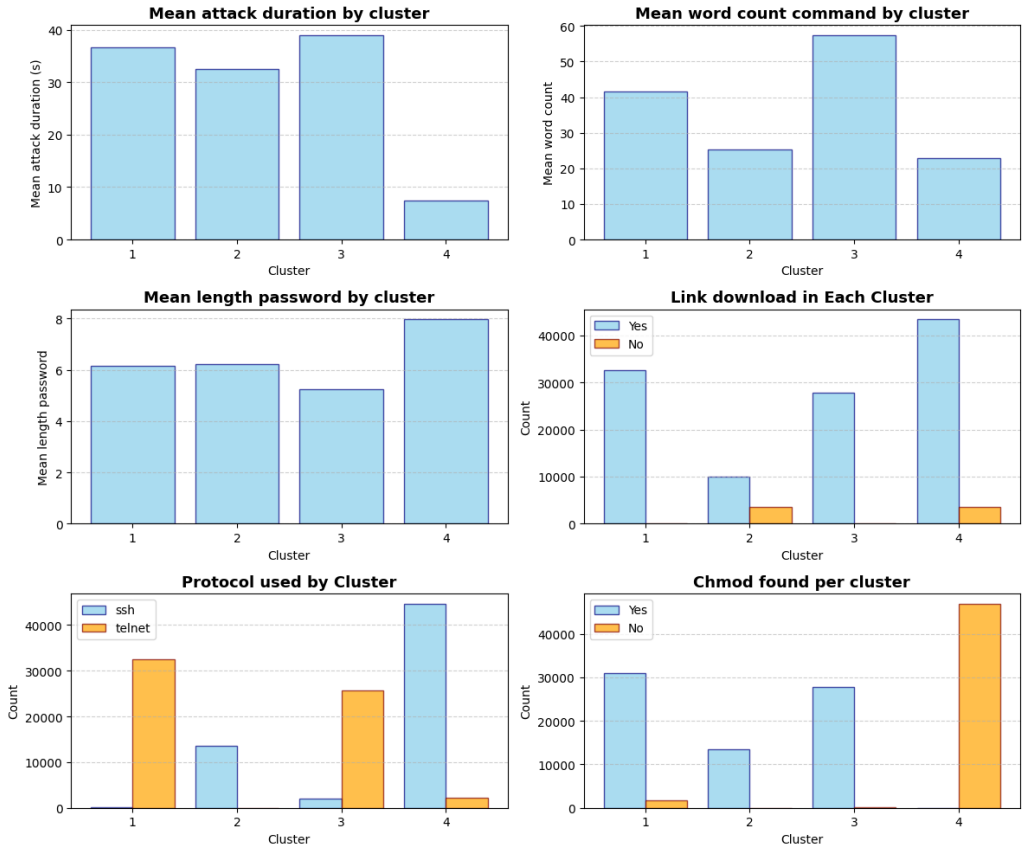


FIGURE B.6: Information about clusters for the high threat group using K-prototypes approach.

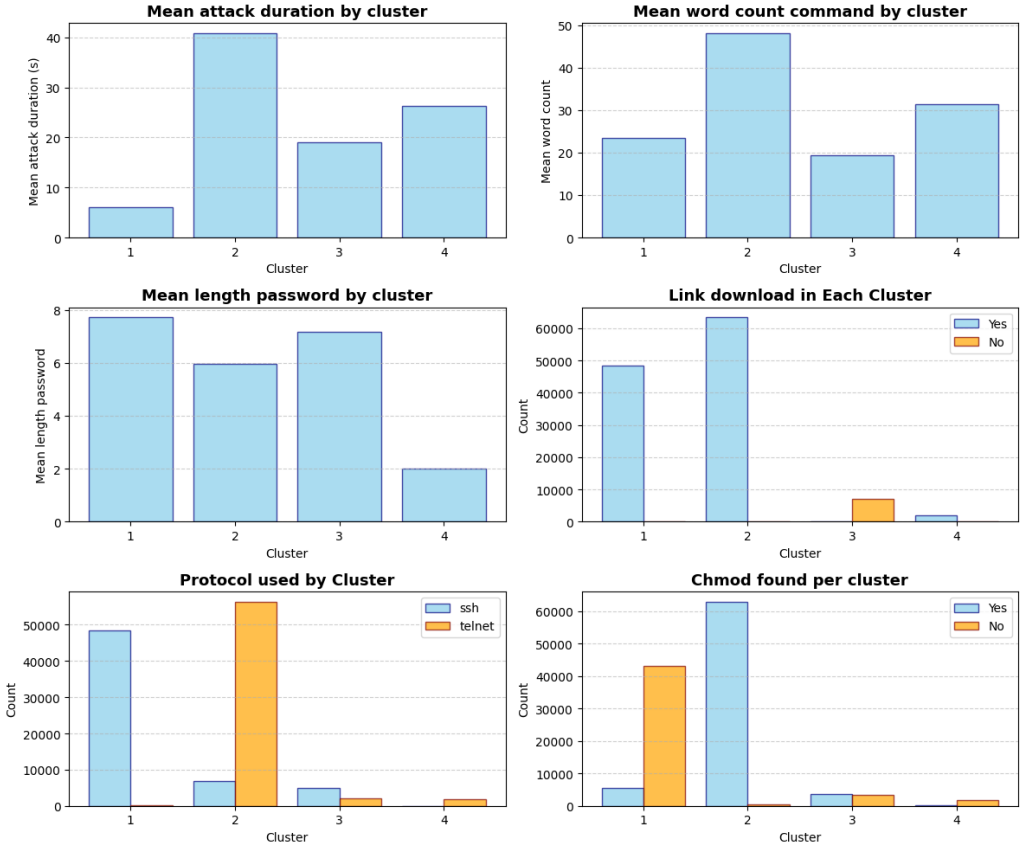


FIGURE B.7: Information about clusters for the high threat group using FAMD & K-means approach.

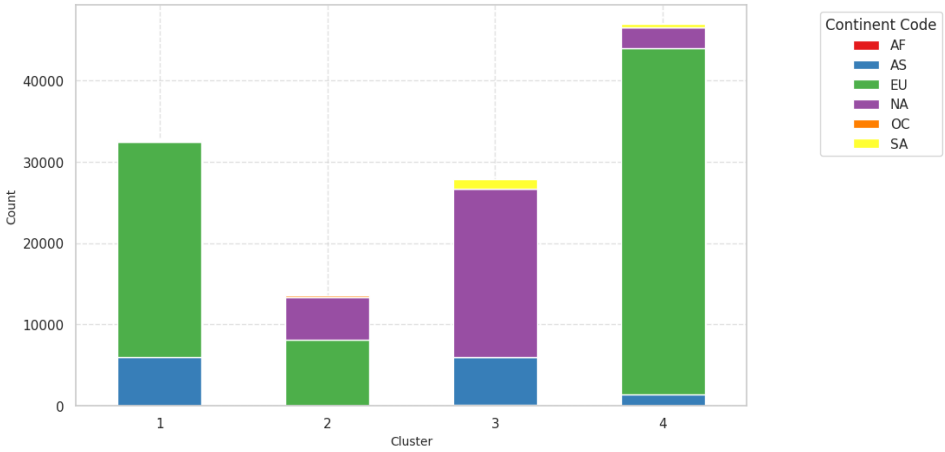


FIGURE B.8: Distribution of clusters, given by K-prototypes, across continents (indicating the origin of the attacks) for the high threat group.

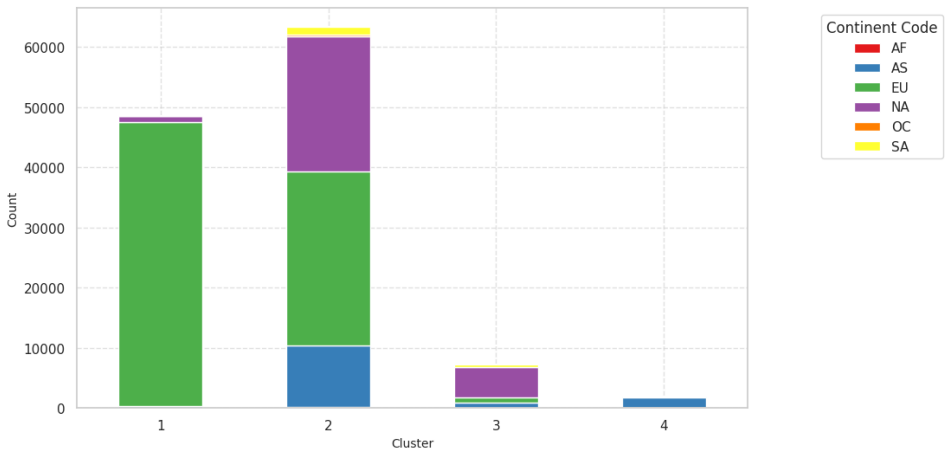


FIGURE B.9: Distribution of clusters, given by FAMD & K-means, across continents (indicating the origin of the attacks) for the high threat group.

Appendix C

Predictive Anlysis extended Results

In Chapter 4, we examined the outcomes of our predictive analysis, with a particular emphasis on visually evaluating the time series data for the United States. It is important to recall that we also investigated time series data for Spain, Singapore, Germany, and Japan. The MAPE metrics were presented in the previous chapter for all the countries (consult Table 4.5); now, let us delve into the visual assessment[1].

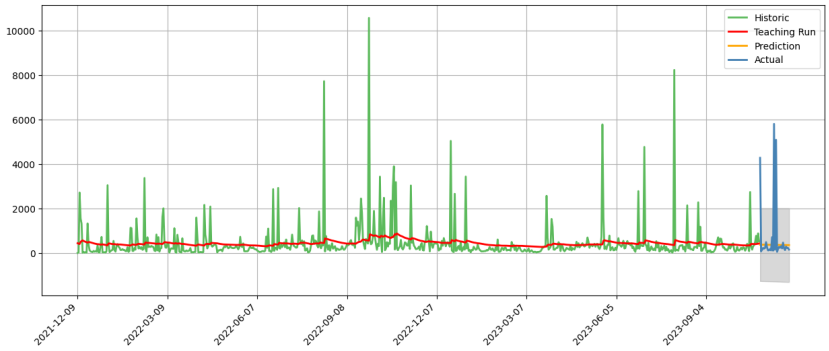
Let us begin by examining the time series data from Spain. Predicting this series is challenging because it exhibits a consistent trend with unexpected peaks. In Figures C.1 and C.2, we can see the outcomes produced by traditional methods, ARIMA and Prophet, respectively. Figure C.3 displays the results generated using SVM, and Figure C.4 shows the results obtained with LSTM. It is evident from the figures that no model successfully captures the two challenging peaks in the test set, resulting in overall poor model performance. Additionally, it is noteworthy that all models yield a similar prediction—a constant line—attributed to the instability of the time series, characterized by a persistent trend with unexpected peaks.

Now, let us examine the time series data from Singapore. In Figures C.5 and C.6, we observe the outcomes obtained through traditional methods, ARIMA and Prophet, respectively. Figure C.7 and C.8 present the results produced by XGBoost and SVM, respectively, while Figure C.9 displays the outcomes obtained with LSTM. It is noteworthy that the classical models exhibit an inferior performance compared to the machine learning or deep learning approaches. Specifically, none of the approaches effectively capture the peaks, but the trend is accurately captured in the machine learning and deep learning approach.

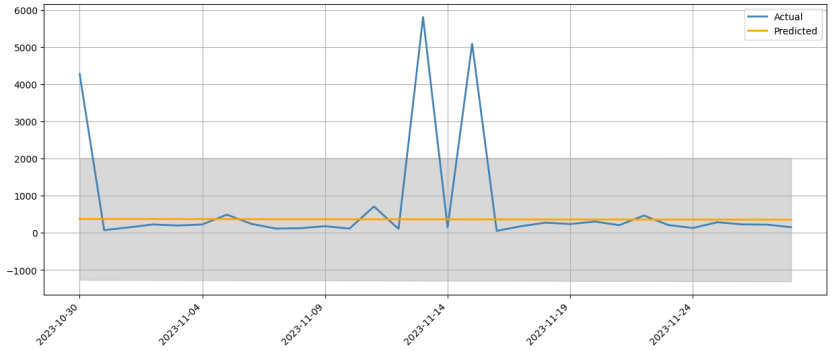
Let us proceed with the analysis of the time series data from Germany. In Figures C.10 and C.11, we examine the results obtained through traditional methods, specifically ARIMA and Prophet, respectively. Figure C.12 and C.13 showcase the outcomes generated by a voting ensemble model[2] and SVM, respectively, while Figure C.14 displays the results obtained with LSTM. In this case, we choose to highlight the ensemble model as it emerged as one of the top-performing models for this particular time series. The results are similar to the preceding ones, wherein classical models demonstrate notably poor performance, while machine learning and deep learning models exhibit strong overall performance, capturing the general trend of

[1] In the machine learning analysis, various models were evaluated through visual assessments. To view the performance of all these models, please refer to the corresponding Jupyter notebook. Here, we will exclusively showcase the most effective models, in most of the cases XGBoost and SVM.

[2] The ensemble model comprised a combination of a Random Forest, an XGBoost, and an SVM.



(A) Actual Time series, Trained values and Predicted Values.

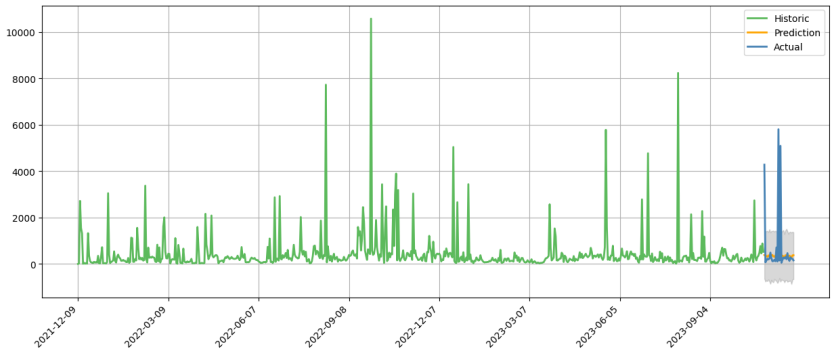


(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

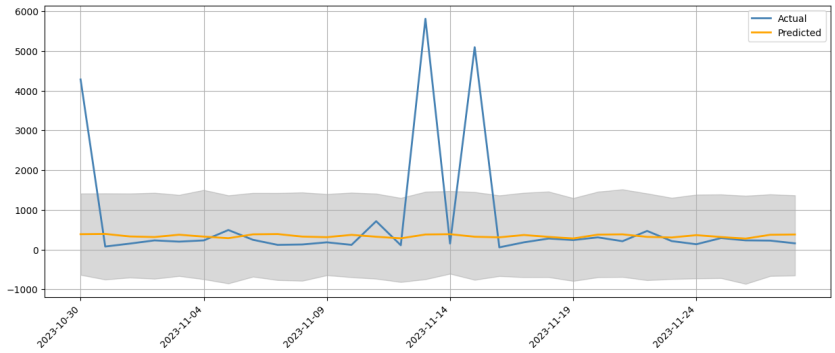
FIGURE C.1: Analysis for the ARIMA model on the Spain times series.

the series but not the peaks.

Concluding our analysis, we delve into the study of the time series data from Japan. In Figures C.15 and C.16, we examine the outcomes obtained through traditional methods, specifically ARIMA and Prophet, respectively. Figure C.17 and C.18 highlight the results generated by XGBoost and SVM, respectively, while Figure C.19 displays the outcomes obtained with LSTM. Notably, classical methods exhibit a poor performance, whereas machine learning and deep learning approaches demonstrate strong effectiveness. An interesting observation is that the models appear to overestimate the results, providing predictions higher than the actual values.

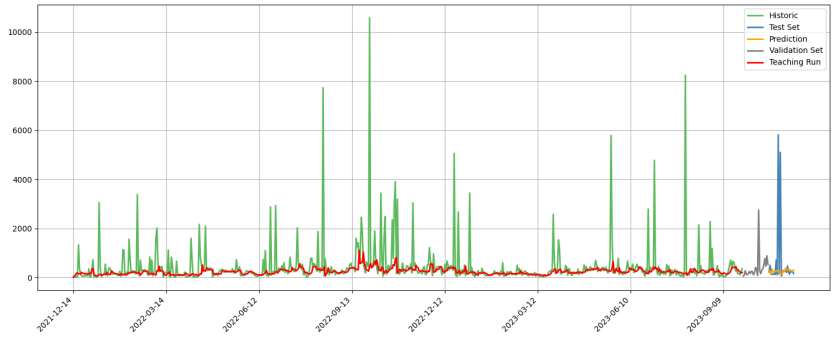


(A) Actual Time series and Predicted Values.

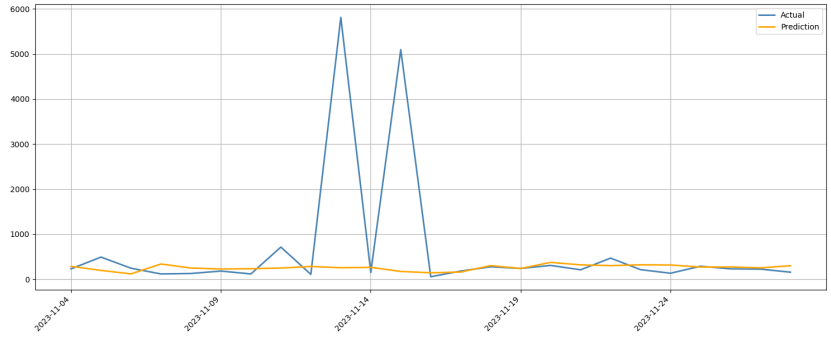


(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

FIGURE C.2: Analysis for the Prophet model on the Spain times series.

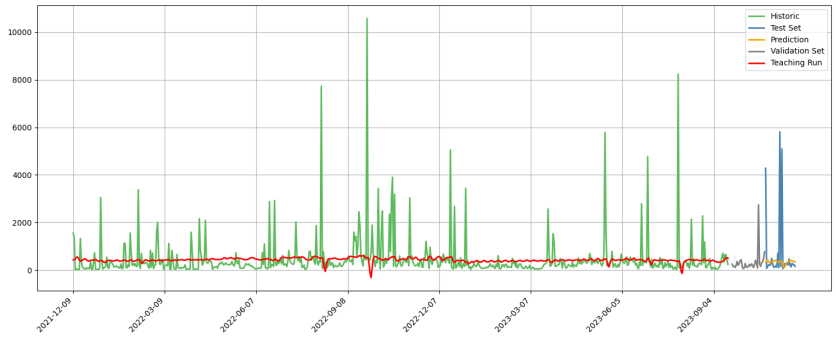


(A) Actual Time series, Trained values and Predicted Values.

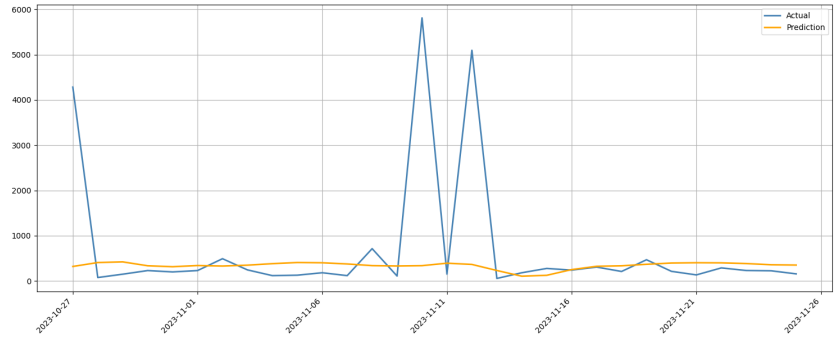


(B) Actual vs. Predicted (Test set).

FIGURE C.3: Analysis for the SVM model on the Spain times series.

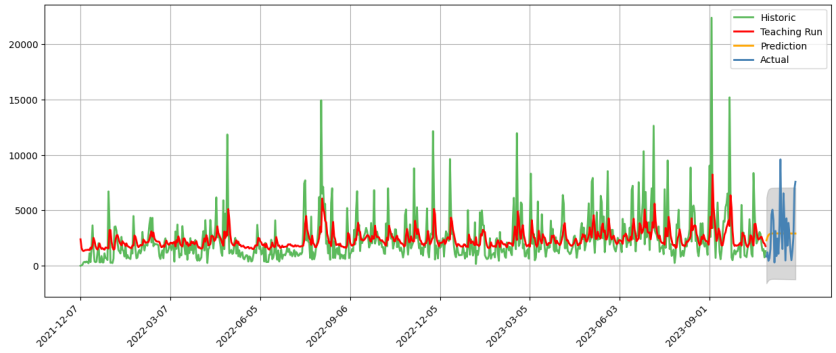


(A) Actual Time series, Trained values and Predicted Values.

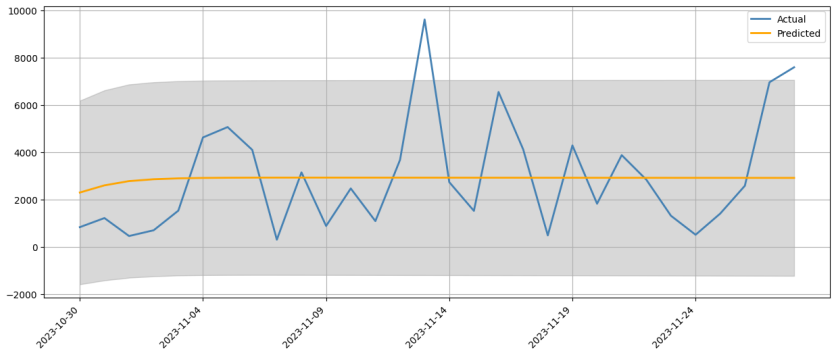


(B) Actual vs. Predicted (Test set).

FIGURE C.4: Analysis for the LSTM model on the Spain times series.

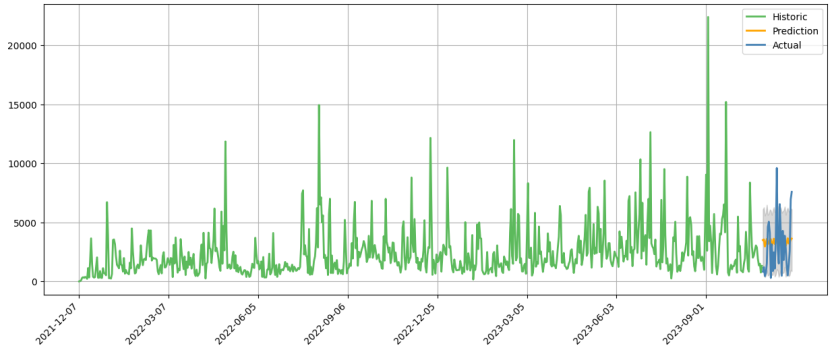


(A) Actual Time series, Trained values and Predicted Values.

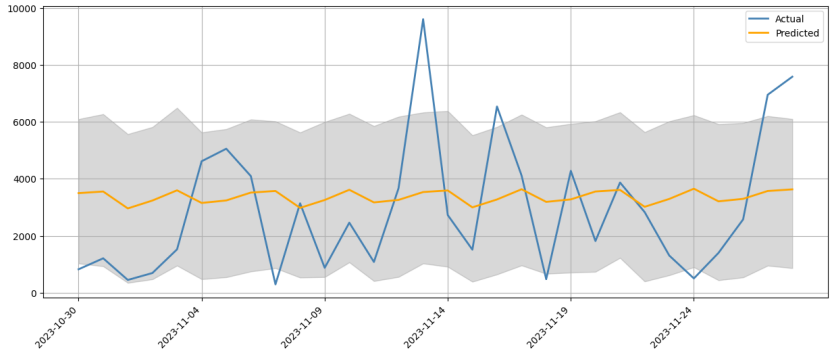


(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

FIGURE C.5: Analysis for the ARIMA model on the Singapore times series.

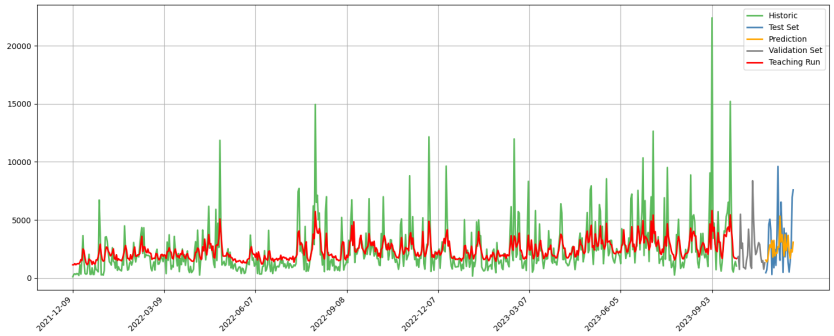


(A) Actual Time series and Predicted Values.

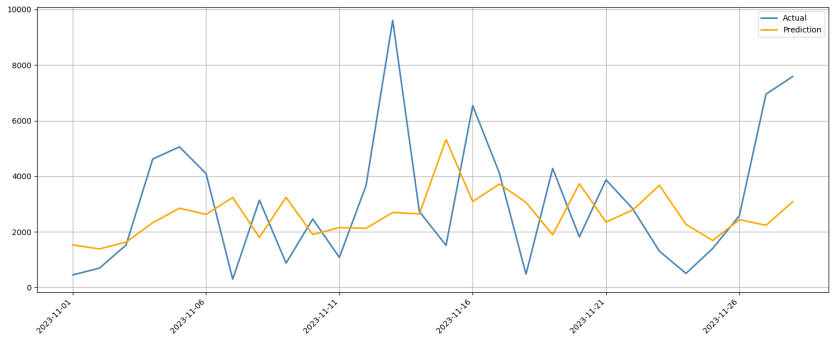


(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

FIGURE C.6: Analysis for the Prophet model on the Singapore times series.

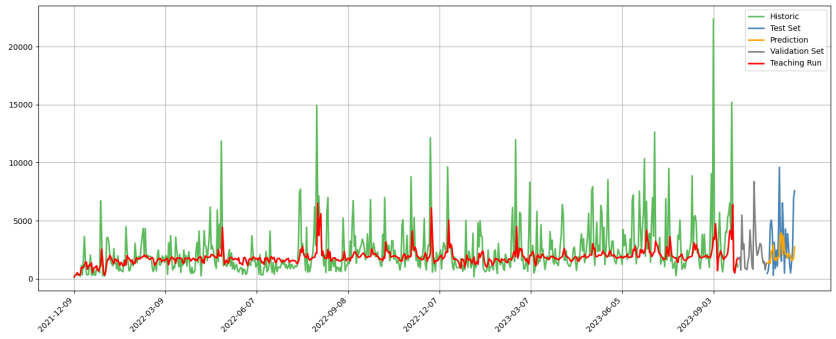


(A) Actual Time series, Trained values and Predicted Values.

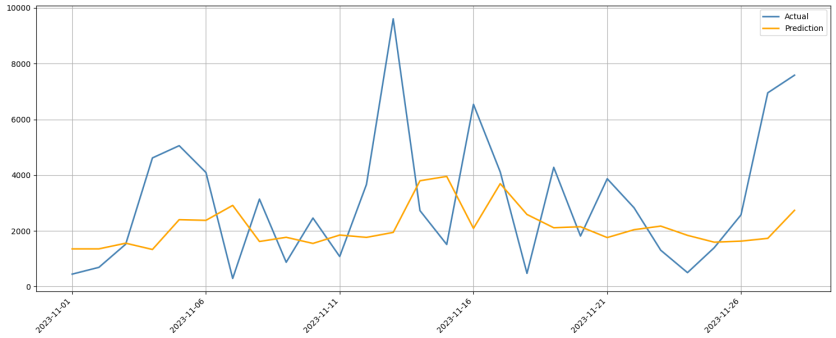


(B) Actual vs. Predicted (Test set).

FIGURE C.7: Analysis for the XGBoost model on the Singapore times series.

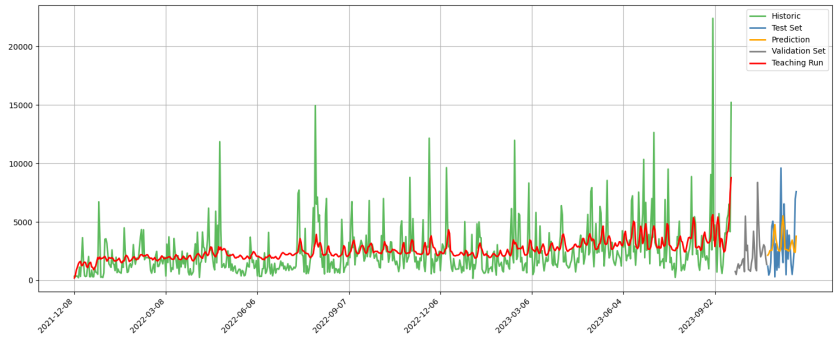


(A) Actual Time series, Trained values and Predicted Values.

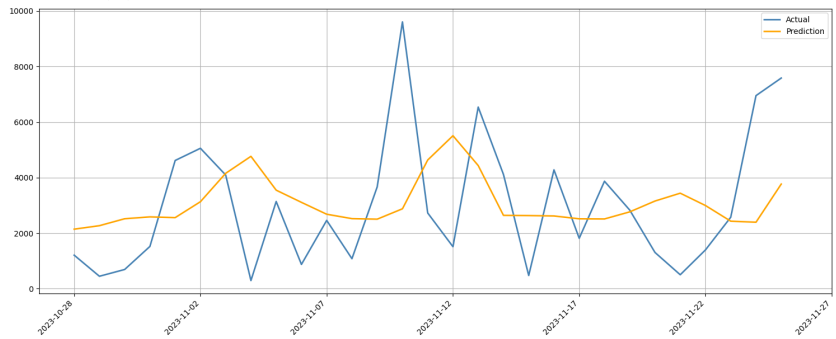


(B) Actual vs. Predicted (Test set).

FIGURE C.8: Analysis for the SVM model on the Singapore times series.

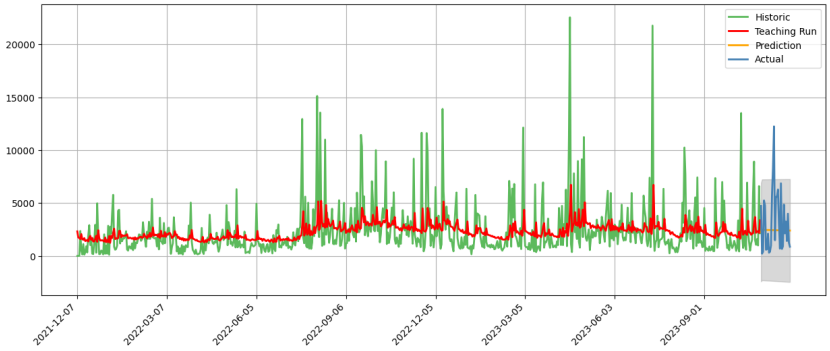


(A) Actual Time series, Trained values and Predicted Values.

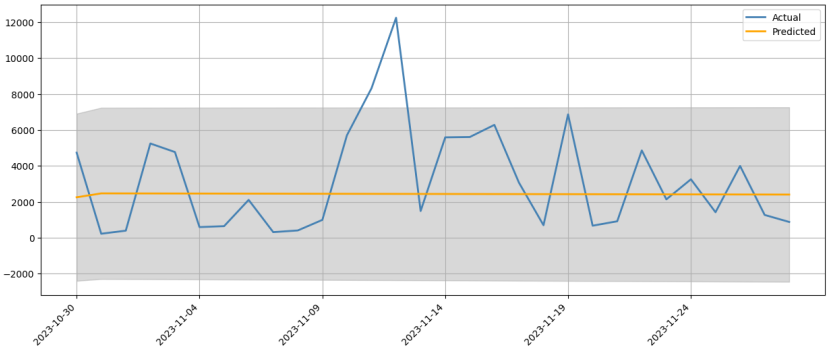


(B) Actual vs. Predicted (Test set).

FIGURE C.9: Analysis for the LSTM model on the Singapore times series.

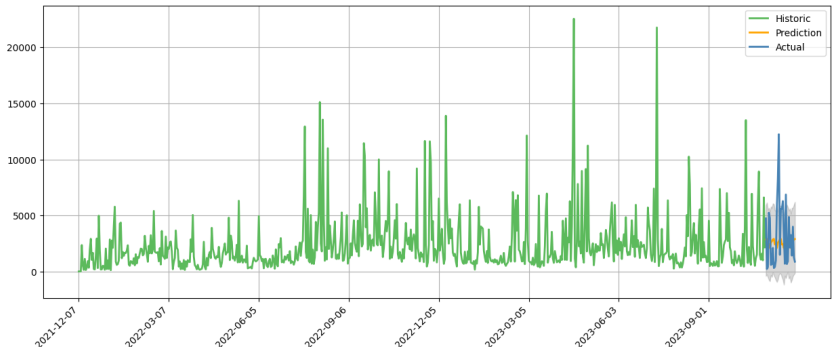


(A) Actual Time series, Trained values and Predicted Values.

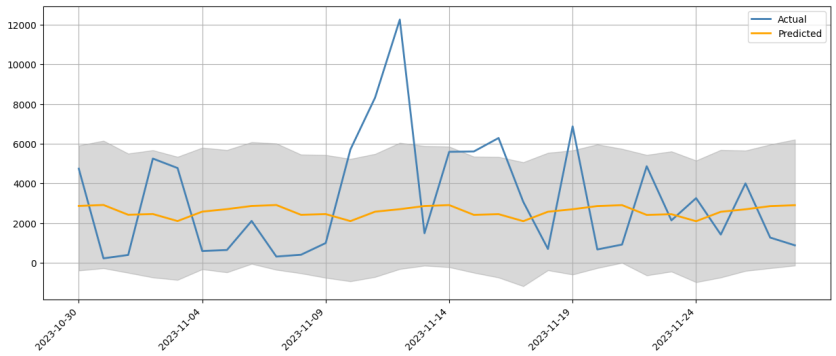


(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

FIGURE C.10: Analysis for the ARIMA model on the Germany times series.

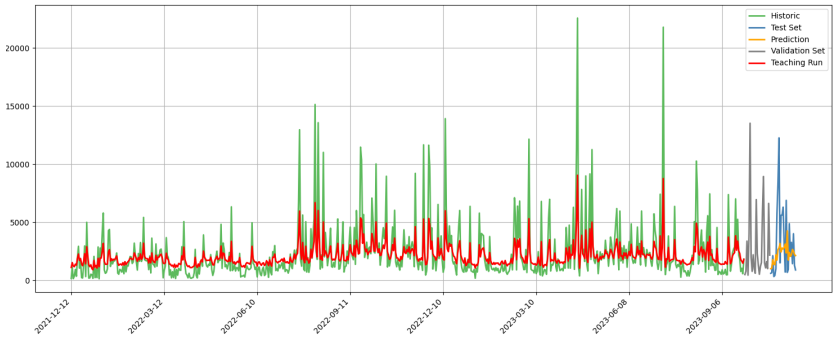


(A) Actual Time series and Predicted Values.

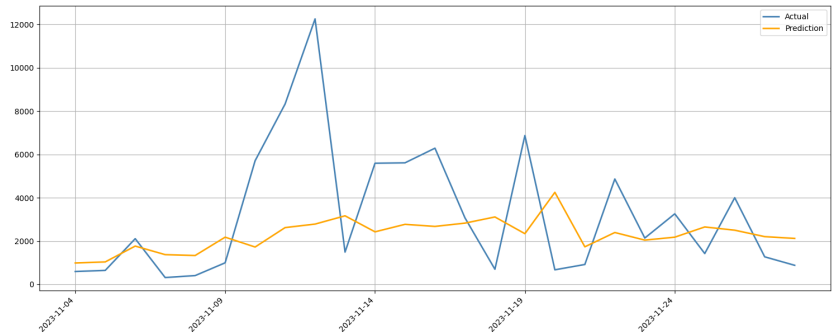


(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

FIGURE C.11: Analysis for the Prophet model on the Germany times series.

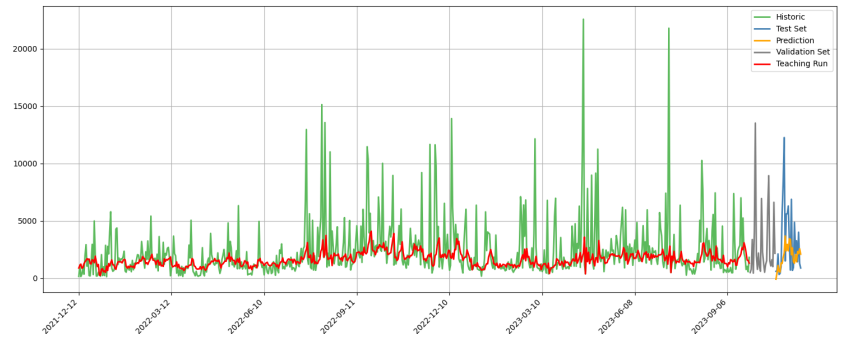


(A) Actual Time series, Trained values and Predicted Values.

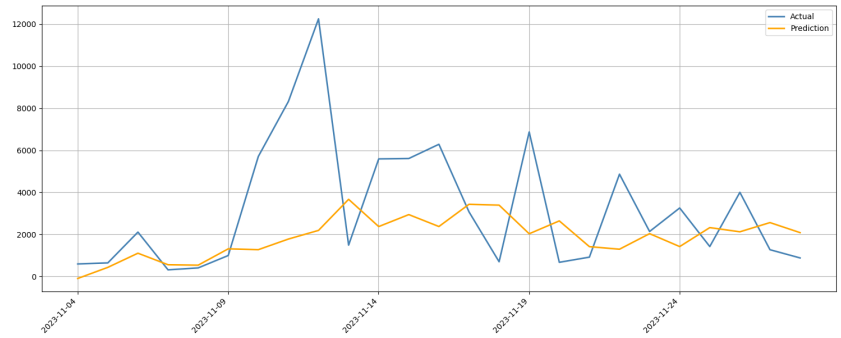


(B) Actual vs. Predicted (Test set).

FIGURE C.12: Analysis for the Voting ensemble model on the Germany times series.

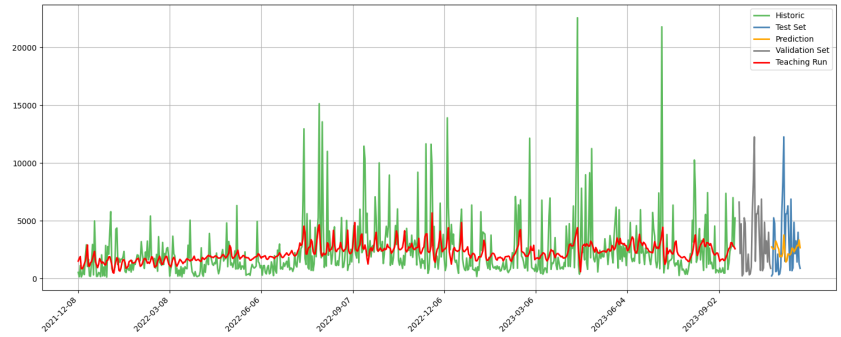


(A) Actual Time series, Trained values and Predicted Values.

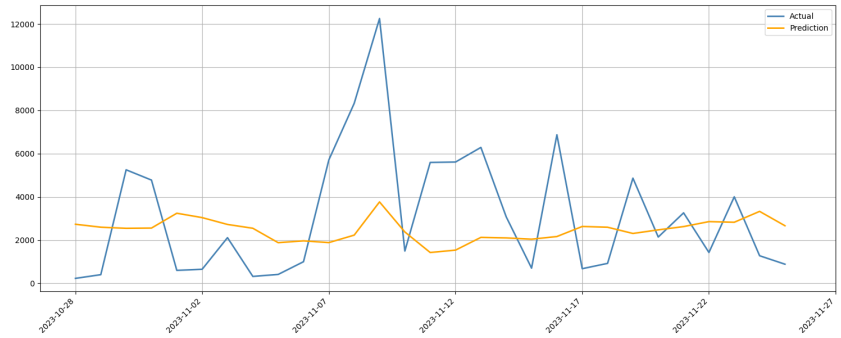


(B) Actual vs. Predicted (Test set).

FIGURE C.13: Analysis for the SVM model on the Germany times series.

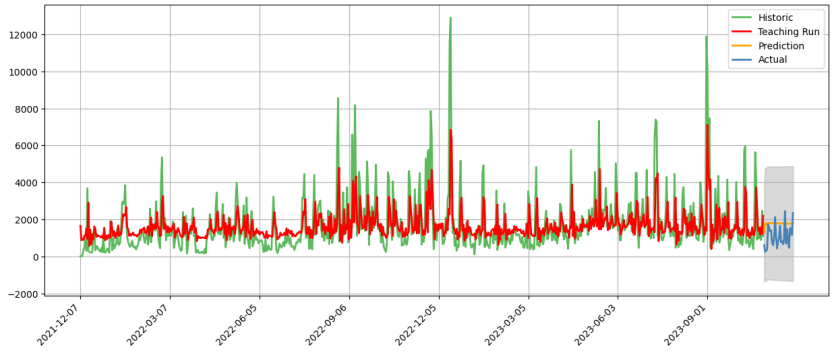


(A) Actual Time series, Trained values and Predicted Values.

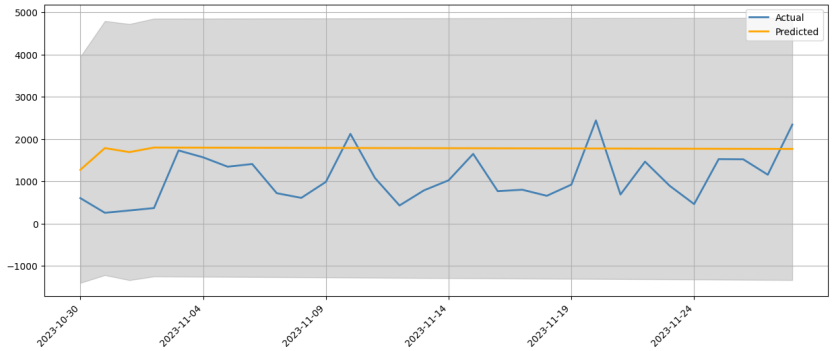


(B) Actual vs. Predicted (Test set).

FIGURE C.14: Analysis for the LSTM model on the Germany times series.

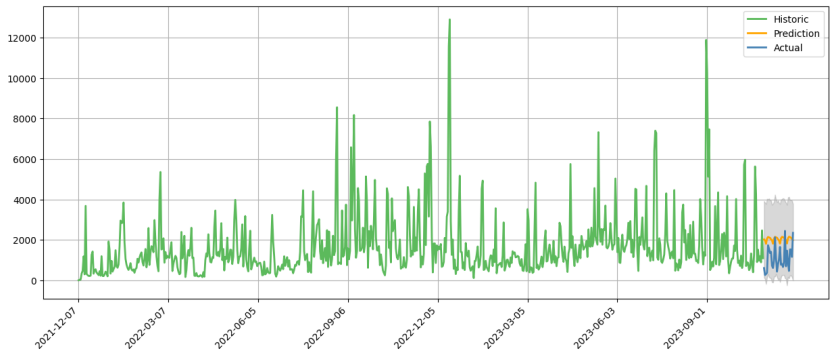


(A) Actual Time series, Trained values and Predicted Values.

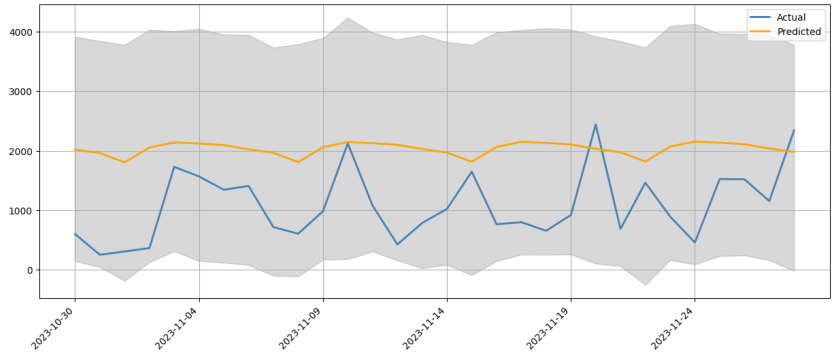


(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

FIGURE C.15: Analysis for the ARIMA model on the Japan times series.

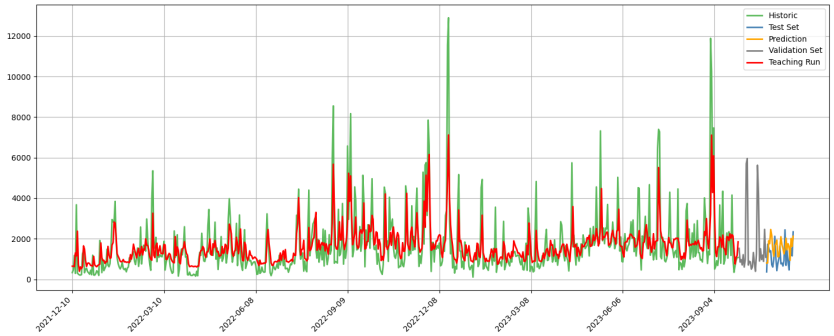


(A) Actual Time series and Predicted Values.

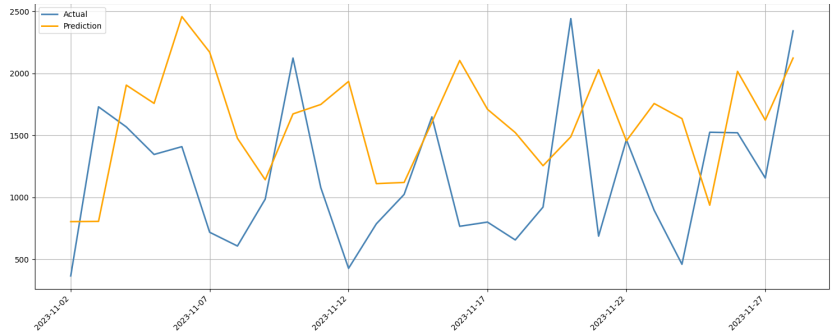


(B) Actual vs. Predicted (Test set). The grey area is the confidence interval given by the model.

FIGURE C.16: Analysis for the Prophet model on the Japan times series.

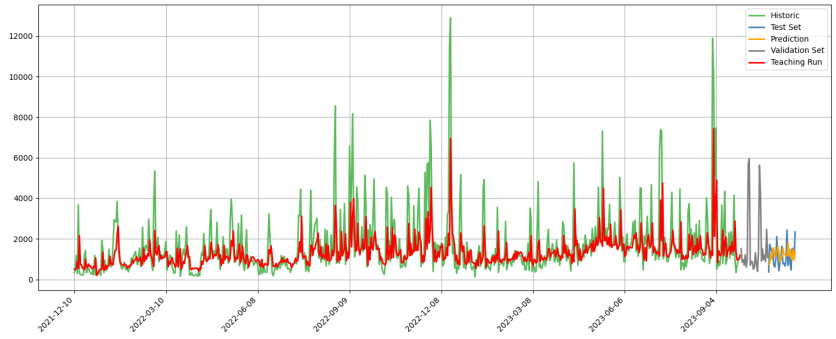


(A) Actual Time series, Trained values and Predicted Values.

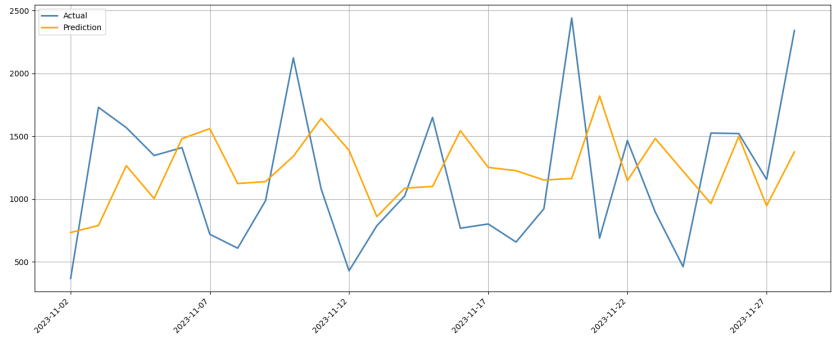


(B) Actual vs. Predicted (Test set).

FIGURE C.17: Analysis for the XGBoost model on the Japan times series.

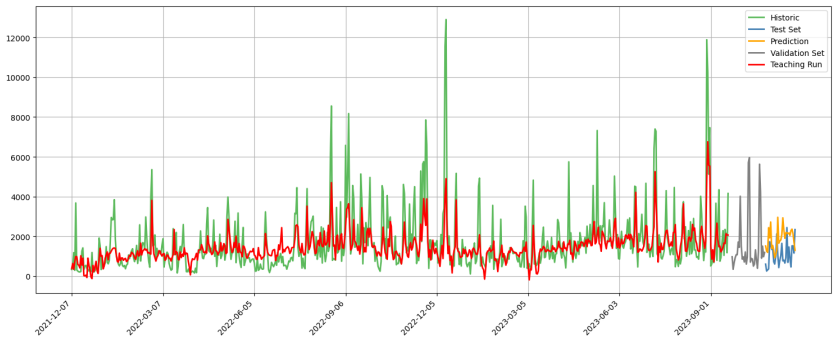


(A) Actual Time series, Trained values and Predicted Values.

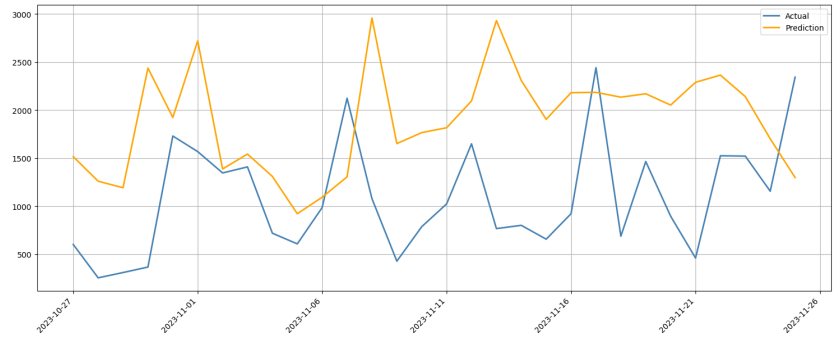


(B) Actual vs. Predicted (Test set).

FIGURE C.18: Analysis for the SVM model on the Japan times series.



(A) Actual Time series, Trained values and Predicted Values.



(B) Actual vs. Predicted (Test set).

FIGURE C.19: Analysis for the LSTM model on the Japan times series.

Appendix D

Source code

All the code used in this research project is available on the GitHub repository: https://github.com/davidrosado4/cyber-meets-ml. The repository contains detailed documentation, including instructions for replicating the experiments and analyses conducted in this thesis.

Due to privacy concerns, the raw data used in this research cannot be publicly shared. However, feel free to contact me directly at rosadodav4@gmail.com for access to a representative sample of the data or any additional information needed to replicate the study. Please note that any use of the provided data sample is subject to the same privacy that governed the original data usage.

Bibliography

Box, George EP et al. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Dasgupta, Dipankar, Zahid Akhtar, and Sajib Sen (2022). "Machine learning in cybersecurity: a comprehensive survey". In: *The Journal of Defense Modeling and Simulation* 19.1, pp. 57–106.

Dua, Sumeet and Xian Du (2016). *Data mining and machine learning in cybersecurity*. CRC press.

Fang, Xing et al. (2019). "A deep learning framework for predicting cyber attacks rates". In: *EURASIP Journal on Information security* 2019, pp. 1–11.

Fraunholz, Daniel et al. (2017). "Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot". In: *2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security)*. IEEE, pp. 1–7.

Gan, Guojun, Chaoqun Ma, and Jianhong Wu (2020). *Data clustering: theory, algorithms, and applications*. SIAM.

Huang, Zhexue (1998). "Extensions to the k-means algorithm for clustering large data sets with categorical values". In: *Data mining and knowledge discovery* 2.3, pp. 283–304.

Husák, Martin et al. (2021). "Predictive methods in cyber defense: Current experience and research challenges". In: *Future Generation Computer Systems* 115, pp. 517–530.

Kheirkhah, Esmaeil et al. (2013). "An experimental study of ssh attacks by using honeypot decoys". In: *Indian Journal of Science and Technology* 6.12, pp. 5567–5578.

Liu, Yanchi et al. (2010). "Understanding of internal clustering validation measures". In: *2010 IEEE international conference on data mining*. IEEE, pp. 911–916.

Martínez Torres, Javier, Carla Iglesias Comesaña, and Paulino J García-Nieto (2019). "Machine learning techniques applied to cybersecurity". In: *International Journal of Machine Learning and Cybernetics* 10, pp. 2823–2836.

Melese, Solomon Z and PS Avadhani (2016). "Honeypot system for attacks on SSH protocol". In: *International Journal of Computer Network and Information Security* 8.9, p. 19.

Mokube, Iyatiti and Michele Adams (2007). "Honeypots: concepts, approaches, and challenges". In: *Proceedings of the 45th annual southeast regional conference*, pp. 321–326.

Rege, Manjeet and Raymond Blanch K Mbah (2018). "Machine learning for cyber defense and attack". In: *Data Analytics* 2018, p. 83.

Tabari, Armin Ziaie, Xinming Ou, and Anoop Singhal (2021). "What are attackers after on iot devices? an approach based on a multi-phased multi-faceted iot honeypot ecosystem and data clustering". In: *arXiv preprint arXiv:2112.10974*.

Taylor, Sean J and Benjamin Letham (2018). "Forecasting at scale". In: *The American Statistician* 72.1, pp. 37–45.

Valero, José María Jorquera et al. (2020). "Identification and classification of cyber threats through ssh honeypot systems". In: *Handbook of Research on Intrusion Detection Systems*. IGI Global, pp. 105–129.

Velden, Michel Van de, Alfonso Iodice D'Enza, and Angelos Markos (2019). "Distance-based clustering of mixed data". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 11.3, e1456.

Von Luxburg, Ulrike (2007). "A tutorial on spectral clustering". In: *Statistics and computing* 17, pp. 395–416.

Zuzčák, Matej and Petr Bujok (2021). "Using honeynet data and a time series to predict the number of cyber attacks". In: *Computer Science and Information Systems* 18.4, pp. 1197–1217.