A Big Data Analysis Model for Futuristic Insights in Cybersecurity Networks

Mangilanyane Precious Leutle

201626764

orcid.org/ 0000-0002-4543-7375

A Thesis Submitted in Fulfilment of the Requirements for the Degree of Master of Computing

Supervisor: Dr. Olalekan Samuel Ogunleye

School of Computing and Mathematical Sciences

Faculty of Agriculture and Natural Sciences

May 2026

UNIVERSITY OF MPUMALANGA

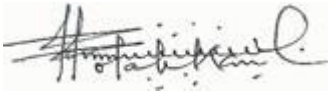
DECLARATION

I Mangilanyane Precious Leutle, declare that this dissertation titled "**A Big Data Analysis Model for Futuristic Insights in Cybersecurity Networks**" is my own work submitted to the Faculty of Agriculture and Natural Sciences, University of Mpumalanga for the award of the degree Master of Computing is my own original work and has never been submitted for assessment to any other university or for another qualification. I further declare that all sources used or consulted have been acknowledged by citation and inclusion in the reference list.

Also, note that part of this work has been accepted for publication as a conference paper titled "***Machine Learning Model for Classification in Cybersecurity Networks***" at the IMITEC2025 Conference.

Signed:  Date: 08th October 2025

Mangilanyane Precious Leutle
Student

Signed:  Date: 08th October 2025

Dr. Olalekan Samuel Ogunleye
Supervisor

DEDICATIONS

This dissertation is dedicated to my late mother, Lilly Matjatji Leutle, someone who always believed that I was meant for greatness. From an early age, she motivated me to prioritize my studies for a better future, independence, and countless opportunities.

I want to express my gratitude to my family and mentors for their endless love and encouragement. I could not have completed my postgraduate studies without your unwavering support. I truly appreciate your guidance and everything you have done for me.

To Prof. Thoko Mayekiso, Vice Chancellor of the University of Mpumalanga, thank you for your support and guidance throughout my academic journey. I appreciate your assistance and support in rewriting this story.

To God, a most special tribute is made for granting me the strength and courage to persevere through all the challenges that came with working and studying.

ACKNOWLEDGEMENTS

I would like to express my appreciation to my supervisor, Dr. Olalekan Samuel Ogunleye, for his support and guidance throughout my studies. I would also like to express my gratitude to my mentors, colleagues, and friends for their support, guidance, and encouragement throughout my studies. Lastly, I would like to thank my late supervisor, Professor Billy Kalema, for his support and guidance in this study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

ABBREVIATIONS

COVID-19	Coronavirus Disease 2019
AI	Artificial Intelligence
IoT	Internet of Things
4IR	Fourth Industrial Revolution
ICT	Information and Communications Technology
ML	Machine Learning
5V's	Volume, Velocity, Variety, Veracity, and Value
HIDS	Host Intrusion Detection System
NIDS	Network Intrusion Detection System
IDS	Intrusion Detection System
DDOS	Distributed Denial of Service
CIA	Confidentiality, Integrity, Availability
TOE	Technology, Organization, and Environment
REC	Research Ethics Committee
UMP	University of Mpumalanga
NIST	National Institute of Standards and Technology
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege
DREAD	Damage, Reproducibility, Exploitability, Affected users, and Discoverability

ABSTRACT

Big data has changed how organizations collect, process, store, and use information for informed decision-making. Big data is produced every day in various forms and is characterized by the 5Vs, namely: volume, velocity, variety, veracity, and value. Processing such data is a challenge, as many organizations use traditional transactional processing systems that are increasingly overwhelmed by the data currently generated by the wide adoption of Information Technology and the Internet. This makes security governance and data analysis a challenge. As such, organizations have found themselves having to enhance their cybersecurity measures to prevent malicious data from penetrating the network. By analyzing big data, organizations can gain better insights to achieve value. This motivated the study to use machine learning to develop a big data analysis model to gain futuristic insights into cybersecurity networks. Futuristic insights refer to the model's proactive approach to predicting previously unknown threat patterns based on learned historical characteristics, rather than the traditional reactive signature-matching approach. Machine learning has been explored as a proactive approach to cyber threat detection in recent years; however, there are limited studies in this field. This study examined the application of machine learning algorithms to identify cyber threats within a network. This study follows an experimental research strategy. This strategy involved synthetically generating data to mimic real-world data that consists of current cyber threats in cybersecurity networks. The original dataset consisted of 50,000 records and six features, which were later engineered into new features because some of the original features had limited predictive power for the target variable. The algorithms known to work well for classification tasks were trained, including Random Forest, Gradient Boosting Classifier, KNeighbors, Logistic Regression, Decision Tree, Gaussian Naive Bayes, and Multilayer Perceptron. The algorithms were further tuned to ensure that they all reached optimal performance. The performance evaluation results, accuracy, precision, F1-score, and recall, were used to determine the best-performing algorithm for building the final model. The study revealed that the best-performing model achieved 97% accuracy.

Keywords: *Big data, cybersecurity, cyber threat detection, cyberattacks, machine learning, algorithms, classification, Random Forest, supervised learning*

CHAPTER ONE: INTRODUCTION

This chapter introduces the field of study and outlines the background that motivated this study. The chapter outlines the research problem that the study intends to address based on the derived research objectives and questions. The chapter also highlights the justification for the study, detailing why investigations into big data analysis for drawing insights are necessary. Furthermore, the chapter highlights the contributions this study makes to theory and practice. Lastly, the chapter provides an outline of the entire dissertation by recapping what a reader can expect to find in each chapter.

1.1 Introduction to the field of study

Cybersecurity is the practice of protecting information by counteracting, detecting, and responding to attacks (Jaipong et al., 2023). Cybersecurity involves protecting information and systems from cyber threats while preserving the confidentiality, integrity, and availability of information. It assists in alleviating cyberattacks and data breaches that may lead to negative implications such as data loss and exposure, data corruption, denial of service, financial losses, and damage to an organization's reputation (Cheng & Wang, 2022). Therefore, it is paramount that organizations comply with cybersecurity practices to protect their data; hence, there is a need to analyze the data that flows through their networks.

Real-time analysis of data flowing through cybersecurity networks is essential for identifying and characterizing all threats present in network traffic. The analysis can assist in detecting and mitigating threats in real time, reducing the risk of cyberattacks and data breaches (Wang & Jones, 2021). However, the critical challenge is that, with the current ongoing digitization and automation, organizations receive vast volumes of internal and external data. These huge volumes of data flowing through cybersecurity networks arrive at high speeds and in various formats. Analysis of such data requires real-time processing, and considering that it originates from various sources, thereby enhancing its veracity (Singh, 2023). This voluminous data, also known as big data, requires special

techniques to draw better insights from it. Proper analysis of this big data enhances cybersecurity (Leenen & Meyer, 2019).

Big data is characterized by 5Vs of volume, velocity, variety, veracity, and value (Alzahrani, 2020). These characteristics of big data have been defined differently by various researchers. Researchers such Gupta, 2023; Wook et al., 2021 all concur that velocity refers the speed at which data is generated, volume the size of data that is being generated and stored, variety the data heterogeneity and sources at different levels such as structured, unstructured, and semi-structured, veracity, the uncertainties of the generated data that is also related to the trustworthiness, quality, and reliability of the data and value the usefulness that is obtained after analysis. The purpose of this study is to develop a Big Data Analysis Model for Futuristic Insights in Cybersecurity Networks.

1.2 Background of the study

In recent years, numerous cyberattacks have occurred in South Africa across various sectors, such as the public, telecommunications, finance, and construction (Pieterse, 2021). The trend and spike in cyberattacks began during the COVID-19 pandemic and continued into 2020. During COVID-19, cyber threats worsened due to organizations going online and having their employees working remotely, increasing vulnerabilities (Alexei et al., 2021). Furthermore, telecommuting is more prevalent, with technological devices being used to carry out organizational operations (Saurombe et al., 2022). This results in exposure to such cyber threats.

The exposure of many organizations to the networked world led to various cyberattacks whose technical and economic effects are still lived today. For instance, a South African municipality fell victim in July 2019 when it was attacked by ransomware that affected the city's electricity utility (Moya, 2019). Later in October, the same municipality suffered a network breach after receiving a ransom note from hackers. These two incidents caused downtime to multiple customer-oriented systems that affected service delivery. Later incidents included the banking sector, which experienced a surge in distributed denial of service (DDoS) attacks (McKane, 2019). South African public universities that were pioneers of online teaching during the COVID-19 pandemic had their bank accounts

compromised, as well as a leakage of students' information (Mungadze, 2021). These were followed by other incidents in later years, such as the Rail, port, and pipeline company and the Justice Department (Moyo, 2021).

The Fourth Industrial Revolution (4IR) is making digital technologies an integral part of society, where the physical, biological, and digital worlds are increasingly interconnected. However, the literature indicates that increased online presence presents cybersecurity threats that negatively impact organizations (Mcanyana et al., 2020). With the increase in digitization and 4IR, organizations' online presence is yet to increase. However, limited solutions have been developed to safeguard against cyberattacks (Dagada, 2024). Organizations that handle voluminous data face more cybersecurity challenges than their counterparts that generate small amounts of data due to various sources and numerous users who send data with many uncertainties (Sharma & Barua, 2023). This is because such organizations are a lucrative and appealing target for cyberattacks due to various factors, including a large number of stakeholders, open access, and outdated systems (Charandura, 2022).

Cybersecurity standards and frameworks mitigate cyberattacks and risks (Taherdoost, 2022). Cybersecurity frameworks offer users the flexibility to select specific components or the entire model, approaches, or specialized practices, providing universally accepted guidelines and recommendations applicable within the organization. On the other hand, cybersecurity standards are employed to ensure that organizations adhere to their established cybersecurity policies and strategies (Baron et al., 2019). Some open standards and frameworks that organizations could implement. However, these standards and frameworks concentrate on the dos and don'ts and cannot identify threat patterns in real-time, especially with increased volumes of data (Baron et al., 2019; Taherdoost, 2022).

1.3 Research Problem

The 21st century has witnessed an increase in cyberattacks across many organizations, especially those dealing with big data, specifically datasets exceeding 50,000 records and high-velocity network traffic (Curtis & Oxburgh, 2023). This wave of cyberattacks has been rampant in many security networks because organizations generate lots of data internally and also receive huge volumes that come from external networks. For organizations to detect cyberattacks that utilize this data, real-time analysis of such data is needed. Analyzing the big data flowing through cybersecurity networks is crucial for identifying threats that could negatively impact the organization's various business units. Despite this, many organizations have limited capacity to analyze this data. Furthermore, the literature suggests that fewer models have been developed to support organizations in analyzing this data, and those that are available often lack contextualization. Analysis of the big data that flows through the cybersecurity networks, especially, is essential for identifying security threats that may have been embedded in the data and compromise the organization's information integrity. As more devices connect to the Internet, collecting, storing, processing, and sharing information becomes a challenge in terms of information security; hence, the need to draw insights from the data that flows within the cybersecurity network.

1.4 Research Objectives

The objectives this study sought to achieve were;

1.4.1 Main objective

The main objective of this study is to develop a big data analysis model for futuristic insights into cybersecurity networks.

1.4.2 Specific objectives

The specific objectives which this study intended to achieve were;

1. To determine the factors that influence big data analysis in cybersecurity networks.

2. To establish methods and techniques that have been used to analyze security threats in cybersecurity networks.
3. To use machine learning classification techniques to identify big data patterns in cybersecurity networks.
4. To use the identified patterns to develop a predictive model for cybersecurity networks.

1.5. Primary Research Question

How can a model for big data analysis in cybersecurity networks be developed?

1.5.1 Research Questions

The following are the secondary research questions that this study was set to answer.

1.5.2 Secondary questions

1. What are the factors that influence big data analysis cybersecurity in networks?
2. What methods and techniques have been used to analyze security threats in cybersecurity networks?
3. What machine learning techniques can be used to identify big data patterns in cybersecurity networks?
4. How can the identified big data patterns be used to develop a predictive model in cybersecurity networks?

1.6 Justification of the study

The increase in the volume and complexity of data offers organizations a competitive advantage; however, it also presents inherent risks, ranging from data loss, corruption, theft, and misuse to malicious attacks or mismanagement (Srivastva, 2023). Other challenges include, though not limited to, cyberattacks, data integrity compromise, and delayed decision-making that leads to poor service delivery (Bongiovanni, 2019). These challenges and risks also compromise the privacy and security of individuals, as well as that of information systems, causing disruptions to essential services. Therefore, big data

analysis can help overcome these challenges by identifying new patterns and insights by using analytical methods that predict and detect cyberattacks in security networks (Lavanya et al., 2020).

The pervasive interconnection of modern technologies continues to grow in size and complexity, exposing new vulnerabilities in networks, while security defenses lag in organizations (Hero et al., 2023). The prevalence of cyber incidents globally contributes to the ongoing rise in cybersecurity concerns (Pieterse, 2021). Several factors can be attributed to this dramatic increase, including the COVID-19 pandemic and the advent of the 4IR. The emergence of the COVID-19 pandemic compelled organizations to advance the adoption and use of Fourth Industrial technologies, while cybercriminals targeted the public sector and large firms to cause severe damage to information systems infrastructure (Chigada & Madzinga, 2021). The 4IR fosters digital convergence, where different devices, databases, and digital networks are interconnected in the cloud (Datta, 2023) . With the challenges presented by the pandemic and 4IR technologies, there is an increased need to leverage cybersecurity opportunities to safeguard information systems and networks that store big data. Thus, big data analysis paradigms such as stream computing may be utilized for processing and analyzing large amounts of data in real-time(Kumar et al., 2022)

1.7 Contributions of the study

The contributions this study makes are twofold, namely, theoretical and practical contributions.

1.7.1 Theoretical contribution

Theoretically, this study developed a model for big data analysis in cybersecurity networks. The developed model has helped identify cyber threats associated with big data within the network. This model made a significant contribution to the computing domain, where researchers in network security and big data analysis will leverage and extend it to further cybersecurity research.

1.7.2 Practical contribution

Practically, the developed model will be utilized by policymakers and management to derive valuable insights that will aid in identifying cyber threats in their networks. This model will make a significant contribution to the policy and management domain, enabling these entities to make informed decisions and put necessary measures in place to mitigate the risk of cyberattacks.

1.8 Research delineation

This study will focus on cybersecurity threat detection in the domain of big data. The predictive model will be developed and empirically validated using synthetically generated datasets, which are secondary datasets, rather than real-world South African networks.

1.9 Thesis outline

Chapter One introduced the background of the study and the research problem. It further focused on the research statement, research objectives, research questions, justification of the study, ethical considerations, and importance of the research study.

Chapter Two covered a literature review that contains credible sources, thoughts, and opinions on the significance of big data analysis in cybersecurity networks. Cybersecurity is discussed in detail, focusing on its challenges and the application of cybersecurity principles in data flow within cybersecurity networks. Furthermore, big data is also discussed in detail with a focus on big data analysis, security challenges associated with big data, and analysis of big data threats.

Chapter Three dealt with the research methodology. It briefly explained the different methods for conducting the study, the reasons used to conduct the study, and why these methods were chosen. It provided a step-by-step guide on how the research was to be conducted to achieve the main objective: to develop a big data analysis model for futuristic insights in cybersecurity networks and the study's secondary objectives.

Chapter Four discussed the setup of the experimental environment, with a focus on the tools and software to be used. The chapter further discusses the execution of the methodology and the process followed to obtain the results of the experiment and develop the model.

Chapter Five deals with the conclusions related to the research questions, a summary of this study, contribution of this study to Big Data analysis. The chapter concludes with suggestions for future research and the study's limitations.

CHAPTER TWO: LITERATURE REVIEW

This chapter discusses literature on data analysis in cybersecurity networks. It further provides a literature review of the concept of cybersecurity, including its challenges and applications. The chapter further discusses big data, including its characteristics, analysis, and analytics. Furthermore, the chapter discusses data analysis in cybersecurity networks and the computer science frameworks that informed the development of this study's conceptual model.

2.1 The Concept of Cybersecurity

Cybersecurity has become a critical field due to the pervasive role of technology in contemporary society. The emergence of advanced technologies, including cloud computing, Internet of Things (IoT) devices, artificial intelligence (AI), and blockchain, has introduced new vulnerabilities in information systems (Verma, 2024). These systems store sensitive data, and breaches can have a significant impact on organizational operations. In addition to information systems, stakeholders, including businesses, individuals, and governments, face evolving cyber threats. Therefore, a comprehensive understanding of cybersecurity is crucial to safeguard entities that rely on digital technologies. Cybersecurity is defined as the "prevention of damage to, protection of, and restoration of computers, electronic communications systems, electronic communications services, wire communication, and electronic communication, including information contained therein, to ensure its availability, integrity, authentication, confidentiality, and nonrepudiation" (Task Force, 2020). On the other hand, Shaikh and Siponen (2023) define cybersecurity as the protection of information and systems from cyber threats, while maintaining the confidentiality, integrity, and availability of information assets and systems. Essentially, cybersecurity ensures that authenticated entities can access certain information (Ahmed et al., 2023). Thus, the security backbone of any organization relies on three core principles: confidentiality, integrity, and availability. These three core principles form the foundation on which most data security models are built and have revolutionized the field. The following section introduces several models known as data security models. The models deal with data security management, which is in line with

the objectives of this study. Figure 2.1 illustrates the three principles of security, commonly referred to as the confidentiality, integrity, and availability (CIA) triad or security triangle. This illustration is further discussed in detail below.

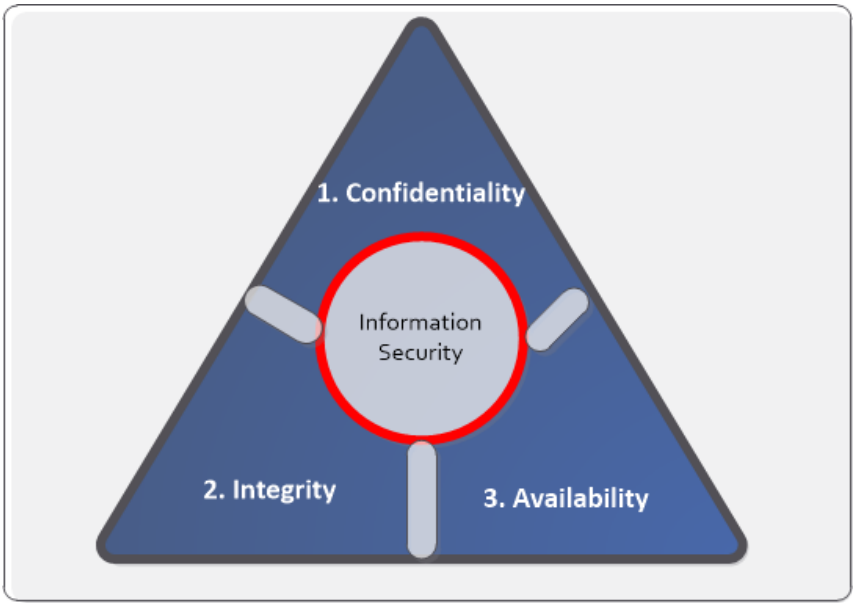


Figure 2.1: The confidentiality, integrity, and availability (CIA) triad (Verma, 2024)

2.1.1 The Confidentiality, Integrity, and Availability Model

The security triangle known as the Confidentiality, Integrity, and Availability (CIA) triad has been used to define information security. Over the years, it has also evolved to define cybersecurity (Ham, 2021). The CIA triad has been expanded over the years to include additional aspects such as non-repudiation, possession, and utility. These CIA triad elements have served as a foundation for the theoretical understanding of information security and guidelines through which security practices were developed and implemented in organizations (Mpekoa,2024). The following are definitions of each element of information security (Mitchell & Osazuwa, 2023):

a) **Confidentiality:** A mechanism used to restrict the access and disclosure of information to unauthorized systems, individuals, and entities. It ensures that information is readily available to only authorized individuals and entities.

b) **Integrity:** A mechanism used to restrict any destruction or modification of information in a manner that is not authorized. It ensures that information has not been tampered with or changed through logs or tracking.

c) **Availability:** A mechanism that is used to ensure that information systems and assets are available to authorized individuals and entities in a timely manner.

The CIA triad forms the foundation of every type of security in place within an organization. The difference types of security include: (i) network security, which focuses on the practice of safeguarding fundamental networking infrastructure from unauthorized access or harm within a computer network and its related devices, (ii) information security, which focuses on protecting electronic systems, servers, networks, mobile devices, computers, and data from malicious attacks, (iii) application security, which focuses on the security measures placed at application level with the purpose of preventing data or code within an application from being stolen or hijacked, (iv) operational security, which focuses on procedures for managing and protecting data assets, (v) cloud security also known as cloud computing security consists of a collection of applications, policies, technologies, and controls used for virtual infrastructure to ensure data protection of online systems (Ahsan et al., 2022).

2.1.2 The Parkerian Hexad Model

The Parkerian Hexad is an information security model that comprises six elements: confidentiality, integrity, availability, possession, authenticity, and utility (Parker, 1998). It includes three additional elements to the well-known CIA triad: possession/control, authenticity, and utility. The Parkerian Hexad was designed to fill the gap in the CIA triad. The CIA triad primarily focuses on the security of technology, protecting information assets, and overlooks the human aspect (Sawik, 2020). Security is about people, not just the technology that they use (Parker, 2010). Thus, the Parkerian Hexad is a more comprehensive and complete model. Figure 2.2 illustrates the six principles of security of the Parkerian Hexad. This illustration is further discussed in detail below.

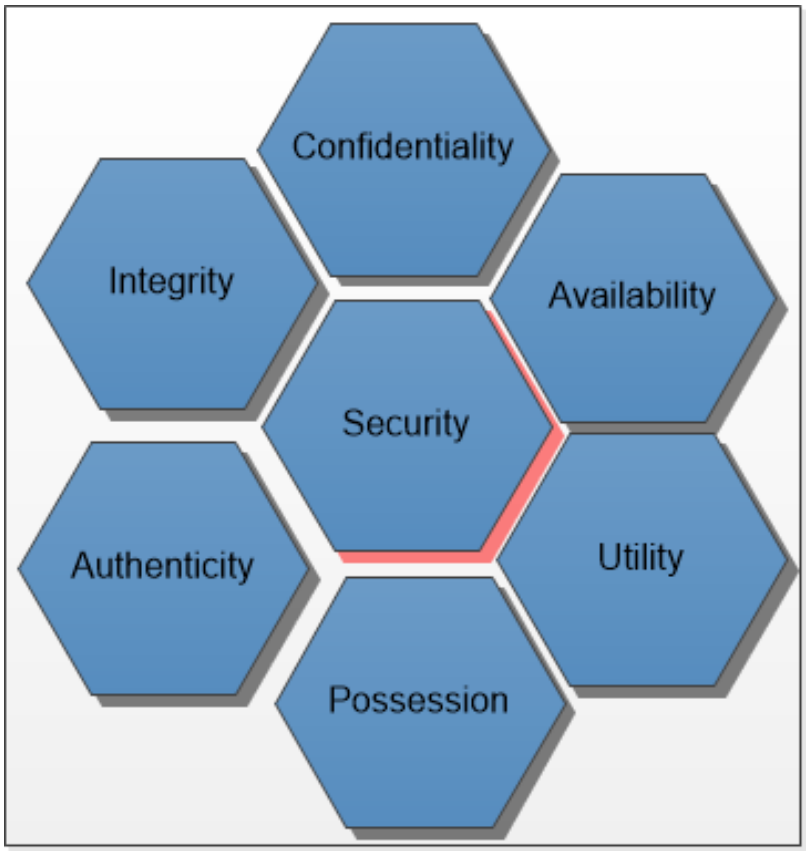


Figure 2.2: Parkerian Hexad (Parker, 1998)

The Parkerian Hexad adds three elements to the CIA triad of confidentiality, integrity, and availability. The three newly added elements are: possession/control, authenticity, and utility (Parker, 1998). In the Parkerian Hexad, integrity refers to the data's consistency or correctness in terms of its inherent properties, irrespective of the authorized actions taken on it. Parker's definition of integrity contradicts that of the CIA triad, which refers to authorized but erroneous modifications as a breach of integrity. However, confidentiality and availability are defined the same across both models. Additionally, Parkerian Hexad defines possession/control as having authorized control and access to information. Authenticity refers to accurate labeling or acknowledgement of the owner of the relevant data. Utility refers to the usefulness of the data. Parker (2010) suggests that the CIA triad is simplistic for complex networks nowadays and may expose environmental threats that they are not equipped to handle. Thus, the CIA triad was enhanced and the Parkerian

Hexad model was introduced, transforming how organizations evaluate and comprehend information security.

2.1.3 NIST Cybersecurity Framework

There is no one-size-fits-all approach to the security needs of different organizations; satisfactory cybersecurity protection can be achieved by adopting a cybersecurity framework that outlines evaluation processes, scope, implementation, and provides a general structure and guidelines for safeguarding crucial digital assets (Antunes et al., 2021). A cybersecurity framework is defined as "a set of rules, procedures, and standards that are designed to combat threats that may be found online" (Chidukwani et al., 2022). A cybersecurity framework helps organizations reduce their vulnerability to cyberattacks and ensures that plans are in place to prevent and mitigate vulnerabilities in computer networks, thereby enhancing countermeasures against cyberattacks (Alshar'e, 2023). Furthermore, they are flexible, allowing users to adopt certain parts or the entire model, technical practices, or approaches, while providing acceptable guiding principles and recommendations for implementation within an organization (Taherdoost, 2022).

2.1.4 Challenges and applications of cybersecurity

In today's digital transformation era, every organization operating in a digital environment has a digital footprint, regardless of its sector or business nature, which increases its susceptibility to cyberattacks and data breaches (Aslan et al., 2023). Cyberattacks and threats became more prevalent during the COVID-19 pandemic, which worsened the digital landscape as business entities moved online for their essential operations, where their crucial data assets lie (Khan et al., 2020). Additionally, a study by the Council for Scientific and Industrial Research (CSIR) reveals that South Africa is ranked as the 8th most targeted country in the world for ransomware (Pieterse, 2021). It is further anticipated that there will be an increase in public sector attacks. The public sector operates in a data-driven environment where there is an upsurge in the volume and complexity of data. This data presents risks of data loss, misuse, or theft through malicious activities or mismanagement. Many of these negative implications threaten the

privacy and security of citizens and cause significant disruptions to crucial services. These inherent risks are attributed to the lack of protection in cyberspace, which has empowered cybercriminals to effectively exploit various hardware and software weaknesses using cyber espionage tools (Li & Liu, 2021). These tools are used to initiate common cyberattacks, including computer network attacks, social engineering, application attacks, cryptography attacks, hijacking attacks, phishing attacks, malware attacks, botnet attacks, password attacks, and man-in-the-middle attacks (Aslan et al., 2023).

In the contemporary digital landscape, cybersecurity has become a pivotal aspect that encompasses various domains, including technology, organization, and national security (Anyanwu et al., 2024). Different sectors, including government, healthcare, education, finance, and telecommunications, are increasingly dependent on Information and Communication Technology (ICT) for the effective execution of their day-to-day operations. With ICT, people can work remotely from anywhere around the world, sharing data and resources, making the organization efficient and productive. Cybersecurity ensures that business operations run smoothly by protecting stakeholders' data from unauthorized and malicious activities while safeguarding the business's interests and reputation (Nnadozie & Zakka, 2024). This is done by making the organization's network robust to cyberattacks and setting an environment that mitigates cyber threats. The prevalence of cyber threats is expanding rapidly with the rapid technological advances; these further heighten challenges in the cybersecurity landscape, as innovative additional vulnerabilities arise with each new innovation, providing a platform for malicious activity (Ahmed et al., 2023). Cutting-edge technologies, including blockchain, artificial intelligence, Internet of Things (IoT), and Cloud computing, are being widely adopted in cybersecurity (Aslan et al., 2023).

2.1.5 Blockchain

Blockchain technology can detect falsified data through its characteristics such as transparency, operational resilience, auditability, and immutability (Alazab & Romdhani, 2020). Furthermore, it can serve as a preventive mechanism against cyberattacks, data breaches, identity theft, and fraudulent transaction activities. Although blockchain

presents these prospective solutions, it also presents some network vulnerabilities, such as partition attacks, delay attacks, sybil attacks, distributed denial-of-service (DDoS) attacks, time-jacking attacks, and transaction malleability attacks (Alazab & Romdhani, 2020). All these network vulnerability attacks exploit blockchain technology's network protocols and decentralized nature, disrupting network functionality, compromising network and security, and affecting the blockchain consensus mechanism.

2.1.6 Artificial Intelligence

Artificial Intelligence (AI) development represents a significant milestone in cybersecurity, introducing a new perspective on how entities counterattack and mitigate cyber threats(Kaur et al., 2023). AI provides defense mechanisms that are automated and intelligence-driven to achieve cybersecurity, which is unprecedented, unlike traditional cybersecurity approaches that often depend on manual interventions and predefined rulesets that are more prone to bias and error due to human involvement. AI can identify patterns and detect anomalies in users' interactions that may reveal intentions of malicious behavior before considerable damage occurs(Yilmaz & Can, 2024). Contemporary deep learning-based approaches perform vulnerability assessment tasks such as code clone detection, vulnerability detection and repair, and vulnerability severity evaluation (Wan et al., 2024). AI research landscape encompasses a rich tapestry of methodologies such as active learning, distance learning, transfer learning, ensemble learning, self-supervised learning, Bayesian learning, hierarchical learning, structured learning, metric learning, continual learning, and feature learning (Barbierato & Gatti, 2024). In incident response, AI algorithms assist organizations in acting swiftly to identify, contain, and neutralize threats, greatly reducing the opportunity for exploitation by automatically responding (Chahal, 2023).

AI is used for threat detection, vulnerability assessment and management, and incident response in cybersecurity. Nevertheless, AI has potential negative consequences that raise cybersecurity concerns (Gonaygunta et al., 2024).

AI technologies have become prime targets for cyber threats. Various threats emerge to exploit the vulnerabilities of AI systems, including the security of the supporting infrastructure, their dependence on data integrity, and the transparency of their algorithms (Familoni, 2024). A major risk in the modern threat environment is the vulnerability of AI systems to adversarial attacks that expose the susceptibility of AI algorithms. This can cause the system to inaccurately predict or classify data. This also leads to ethical concerns concerning accountability and privacy as AI algorithms detect and respond to security threats in an autonomous manner(Gonaygunta et al., 2024) Moreover, the reliability and threat identification are also a concern, AI depends heavily on trained existing data, which may not always produce accurate results, and may make it challenging to detect new threats (Li, 2024).

2.1.7 Internet of Things (IoT)

The Internet of Things (IoT) has penetrated various industries that handle sensitive information, including the healthcare sector and financial sector, as well as diverse domains of life, such as homes, cities, retail, and consumer electronics. It has created a groundbreaking approach enabling interconnected machines and devices in a network to communicate and collaborate, driving innovative process advancements in organizations (Ahmed et al., 2023). According to Natarajan et al. (2024), various characteristics, such as traffic patterns, network capacity, and portability, define IoT networks. IoT's widely distributed, locally intelligent network of smart devices offers services adapted to individual customer needs, enhancing organizations' productivity. In addition, IoT enhances the process of managing and tracking assets and products. IoT serves as a vast data source. Thus, its applications require rapid processing speeds for real-time decision-making, extensive data repositories for storage, and robust broadband

infrastructure for data streaming (AlSalem et al., 2023). While IoT provides benefits, it also presents several challenges.

The excessive adoption of IoT devices presents privacy and data breach concerns to organizations, which affects the flow of work, activities, and network services (Tawalbeh et al., 2020). Interoperability is a challenge due to the varying security protocols among different devices. This means the IoT network consists of heterogeneous devices, and protocols lack standardization (Mishra & Pandya, 2021). Furthermore, traditional security approaches are impractical and incompatible with IoT devices due to their limited processing capabilities, which do not provide the rapid processing speed required for real-time decision making. Connectivity is also a concern; the majority of crucial IoT devices are immersed in data transfer, and the data is vast, traveling at high speeds in different formats. Ensuring seamless data transfer of this nature among different IoT devices proves to be a technical challenge. Furthermore, collecting, processing, analyzing, and manipulating data in this manner is a challenge (Chaudhary et al., 2023). Finally, factors such as power, cost, and size in IoT are prioritized over security in IoT devices, exposing IoT networks, devices, and stakeholders to cyber threats that can lead to significant financial and reputational implications.

2.1.8 Cloud Computing

Cloud Computing is a modern approach to providing computing services and resources across different organizations of different sizes. Cloud computing can be categorized into four types: private, public, hybrid, or multi-cloud. Furthermore, it offers various computing services, including its Infrastructure-as-a-Service (IaaS), Platforms-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The type of service an organization decides to use depends on size, level of control and scalability, level of customization in IT infrastructure, and costs. Cloud computing enables organizations to scale their IT resources in response to changing business demands and market conditions, thereby eliminating hardware upgrade costs and enhancing operational efficiency (Islam et al., 2023) . Furthermore, it serves as a massive data storage and management medium. Cloud computing offers the

ideal platform for modern technologies to handle vast data streams and processes in real-time.

Mohamad et al. (2023) highlight that data security and privacy are challenges in cloud computing. Cloud service providers are expected to ensure data integrity; however, the cloud, like any other technology that hosts data, is susceptible to data security threats such as data breaches, data leaks, and malware attacks, as it rapidly sends and receives large volumes of data. These security threats lead to a lack of trust from users and loss of income, status, and reputation for the cloud service provider. Users may consider switching to another cloud service provider due to security threats experienced by another cloud service. However, issues surrounding initial security considerations, data migration, and network compatibility limit cloud migration flexibility. Additionally, the simultaneous operation of public and private clouds, as well as cloud application development, has proven to be a challenge for distinctive cloud service providers. The absence of consistent data, uniform interfaces, and analytical benefits worsens integration, scalability, and disaster recovery issues (Islam et al., 2023).

The challenges in cybersecurity are mostly attributed to the usage of ubiquitous technologies and the growth of interconnected devices in commercial and noncommercial organizations. The common challenge is big data governance, which involves aligning crucial assets such as standards, people, processes, strategies, policies, and technology to maximize the value of data as a strategic asset while maintaining integrity, consistency, security, availability, usability, reliability, and audibility throughout the life cycle of big data (Yang et al., 2019).

2.2 Data flow in cybersecurity networks

In cybersecurity networks, data flows from internal and external sources. Internal networks involve data flowing within cybersecurity networks from known sources with homogeneous characteristics(Kabir, 2023) . Thus, the attacks that are likely to occur are already established, and measures are in place to mitigate these threats. On the other hand, external data refers to information flowing from outside the organization's boundaries.

Such data originates from various sources, flowing at different speeds and volumes, and presents unexpected threats with limited measures in place to mitigate them (Qureshi et al., 2022). In many instances, external data comes in different formats and with unpredictability (García Lozano et al., 2020). Hence, a model is needed to identify threats in cybersecurity networks that come with big data.

Traditionally, network data flow was monitored at network points where numerous network segments converge into centralized aggregation points (Long & MITRE Corporation, 2021). These points accommodate a single monitoring device to track and log network traffic connected to various network devices. Security network monitoring commonly involves carrying out these three functions at centralized network points: tracking data flow in the network to determine network activity rapidly, conducting real-time intrusion detection (IDS) that signals or determines suspicious activity immediately, and intercepting and capturing all packets in capture files to perform forensics and traditional intrusion detection analysis. The captured files are a universally accepted, abundant information source for forensics used by the Cybersecurity Operation Center for operations such as training, functions, and tooling that rely on these files. However, the advent of cutting-edge technologies and the generation of data at high speeds and large volumes from various sources presents challenges of complexity and integration for traditional systems (Garoufallou & Gaitanou, 2021).

Over the years, traditional network infrastructure has been designed in a three-tier architecture that comprises computing, storage, and networking (Mohammed, 2016). The network is structured to follow a “north-south” flow, where data flows from the end user to the integrated stack, then to the server, and finally to the database. The traffic pattern had to change with the introduction of big data in networks. Big data traffic has higher server and storage node needs than traditional data flow between servers and end users. Big data flow is no longer generated solely by external sources, but also by applications and multiple devices. Big data flow refers to virtual machine or machine-to-machine network traffic, also known as “east-west. According to Dighriri et al. (2017), the traffic volume in the uplink is larger than that of the downlink. Furthermore, the network's devices are machine-to-machine, not human-controlled devices. Machine-to-machine devices

generate higher volumes of traffic than traditional mobile devices. Data streams in different machine-to-machine applications follow distinctive statistical patterns that are challenging to capture (Casale et al., 2010). This exacerbates the challenges of data veracity. Data veracity is affected by data integration, feature extraction, and measurement system limits. Moreover, the ambiguity, uncertainty, collusion by sources, and data falsification exacerbate the situation (Djafri et al., 2022). According to Aaser and Mcelhaney (2021), external data environments are fragmented and experiencing rapid growth. Thus, organizational data environments, systems, and infrastructure must be updated to effectively utilize and implement external data.

2.3 Methods and techniques that have been used to analyse security threats in cybersecurity networks

Cybersecurity threats have been a challenge since the inception of computer networks (Kumar, 2023). Since the 1990s, threat modelling has been a field of interest to researchers; threat modelling methodologies suitable for various methods have been studied (Kim et al., 2022). According to the National Institute of Standards and Technology (NIST) special documentation, “a threat model encompasses the ability to address both the offensive and defensive dimensions of a logical entity, be it data, a host, an application, a system, or an environment.” (Das et al., 2024). STRIDE and DREAD are widely recognized threat modelling models for assessing the security of products and services during their software development life cycle (Kim et al., 2022; Zografopoulos et al., 2021). The STRIDE threat model (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege) classifies threats in correspondence with the core principles of information security that form the CIA triad: confidentiality, integrity, and availability, and further includes additional principles to the triad, these include: authorization, authentication, and nonrepudiation (Kim et al., 2022). It assists organizations in detecting vulnerabilities in applications and potential attacks. Additionally, it may serve as a basis for risk analysis processes to determine the most significant threats and develop the most suitable mitigation measures (Yokoyama & Arima, 2023). The DREAD model evaluates and rates threats based on the risk identified

in the STRIDE methodology (Zhang et al., 2022). It assesses five risk aspects to evaluate the probability of security compromise from different perspectives. These aspects are Damage (D), Reproducibility (R), Exploitability (E), Affected users (A), and Discoverability (Di). STRIDE has become a tool for identifying cybersecurity threats, although other threat modelling models exist, such as Common Vulnerability Scoring System (CVSS), Process for Attack Simulation and Threat Analysis (PASTA), and Attack Tree (Das et al., 2024). Despite the availability of these models, as the Internet became more prevalent and technology advanced, the number and complexity of threats in cybersecurity networks increased drastically. These advancements played a significant role in developing methods and techniques in various domains to mitigate the negative implications of security threats. The following are the methods and techniques used to analyze security threats:

2.3.1 Log Analysis

Log analysis is the most common technique for analyzing security threats. Logs contain data on the sequence of events within a system, application, or network, such as date, time, type of event, or executed action (Strippel, 2019). In cybersecurity networks, log analysis can be performed using either network device files, application server files (database access log files, web server log files), or system log files. Analysis done using network device files involves information about activities carried out by the network infrastructure used in the system (Kara, 2021). This information is derived from network devices, including switches, routers, VPN services, firewalls, and wireless access points. These devices are responsible for recording network traffic information.

2.3.2 Network Traffic Analysis

Network Traffic Analysis is a process of tracking, detecting, and analyzing communication patterns in both encrypted and unencrypted data to monitor security anomalies (Alqudah & Yaseen, 2020). Traffic analysis is conducted to track the traffic flowing through the network, identifying the type of data and its sources within the network. There are various techniques employed in network analysis, namely (Chaudhary et al., 2023): (i) packet

capturing, involves intercepting, capturing and analysing a data packet for detection of potential security threats by making use of packet capture tools such as Wireshark and tcpdump, (ii) behavioural analysis, involves the analysis of network traffic patterns to detect anomalies that may be indicators of security threats and this is done through the use of machine learning approaches and statistical methods, (iii) data flow analysis, involves the analysis of network monitoring protocols such NetFlow or sFlow to identify anomalies and patterns in network traffic, (iv) signature-based analysis, involves comparing network traffic against known signatures in a database consisting of malicious activities. Packet filtering is another technique to analyze security threats in cybersecurity networks. This technique involves interpreting the header content of the packets and determining the course of action to drop or route the packet. The common methods of filtering packets include (Dasgupta et al., 2022): (i) packet filtering based on hop count, involves the preparation a database table in a network by a router to match a valid user with a hop count to any destination (ii) ingress/egress filtering, involves filtering network traffic based on incoming traffic to the local network (ingress) or outgoing network traffic from the local network (egress) that is consistent with the predetermined source IP, (iii) router-based packet filtering, involves filtering routing information on the source and destination IP based incoming packets.

2.3.3 Intrusion Detection Systems (IDS)

Security threats can also be analyze using intrusion detection systems (IDS) data and algorithms. Intrusion Detection Systems monitor and analyse traffic in systems or networks to detect intrusions and anomalies (Hindy et al., 2020). According to Khraisat and Alazab (2021) IDS can analyze data and identify malicious activities in systems. An IDS reacts to detected intrusions by logging information related to the intrusion, triggering alerts, taking corrective actions, and mitigating the situation. There are two types of IDS: Host Intrusion Detection System (HIDS) and Network Intrusion Detection System (NIDS) (Ashiku & Dagli, 2021). HIDS is responsible for tracking internal behavior within the system through access to data such as users' activities and log files. It can access encoded data transmitted over the network, including file attributes, registries, and

configuration databases. NIDS controls and analyzes incoming and outgoing network traffic packets at various locations. Techniques for the analysis of cyber threats include (Ashiku & Dagli, 2021):

1. Anomaly-based detection is a method for analyzing network traffic to identify irregular patterns that deviate from expected behavior. Network traffic is sampled and analyzed using statistical techniques to identify irregularities, and the administrator is notified of anomalies when a predefined threshold is exceeded. Anomaly-based detection can identify new anomalies. However, to do this, it requires significant processing power to continuously track behavioral data; the slightest change from the expected behavior may prompt a surge in false positives.
2. Signature-based detection- Knowledge-based detection is a method for analyzing patterns to pair existing or known signatures. This method examines known attacks using established signatures to identify threats. However, it still requires constant updates to identify unknown attack patterns.
3. Stateful protocol analysis- This detection method functions similarly to signature-based detection. Instead of comparing known signatures, it compares known protocol profiles to network traffic. It detects a random sequence of commands in two layers, the network and application, through pre-set patterns that are vendor-specific.

2.4 Big data

Big data is growing rapidly with the advent of the 4th Industrial Revolution (4IR) and the increasing digitization that has given rise to smart devices and the Internet of Things (IoT) (Thale & Vijayakumar, 2022). This rapidly increasing data is homogeneous in nature and derived from various sources. Big data is one of the most ubiquitous developments in the digital age (Arena & Pau, 2020). Big data analysis has been applied in various industries, including healthcare, finance, education, retail, media, and entertainment, to gain insights into users' behavior, perception, and patterns, thereby improving user experience and revenue for these industries (Al-Khasawneh, 2020).

Currently, big data file sizes are increasing rapidly to the extent that the capacity of big data platforms to process and analyze data is insufficient (Sun et al., 2023) . Analysis of Big data requires using different tools at different stages of processing to gain useful insights and outcomes. These tools are utilized to address data integration, which is challenging due to the complex and massive characteristics of big data that traditional and existing database management tools struggle to process and manage(Gupta, 2023). Figure 2.3 illustrates the difference between Big Data and Traditional Data.

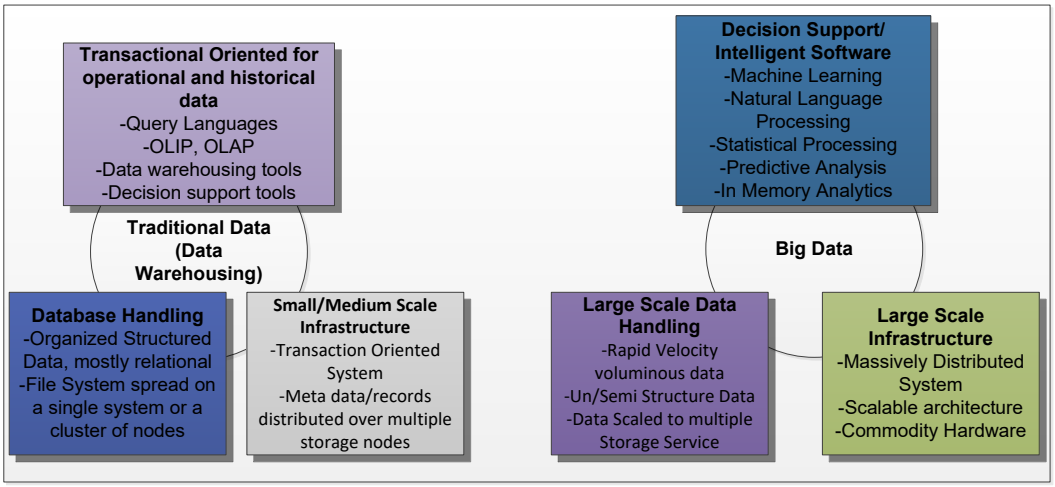


Figure 2.3: Big Data Vs Traditional Data (Data Warehousing) model (Kune et al., 2016)

2.4.1 Big Data Analysis

Big data analysis involves examining vast datasets to uncover hidden associations and relationships, valuable information, and market trends using advanced analytical tools and statistical methods (Thayyib et al., 2023). More still, big data analysis also includes potential strategies such as incremental learning, feature selection, sampling, instance selection, granular computing, parallelization, and divide-and-conquer that can convert broad challenges to insignificant challenges, which can be used to make informed decisions, decrease costs, and enable efficient processing (Hariri et al., 2019). Various methods and techniques can be employed for big data analysis, including but not limited to social media analytics, audio analytics, text analytics, and predictive analysis (Rawat & Yadav, 2021). Big data analysis ensures that the collected, stored, preprocessed,

analyzed, and visualized data can be easily translated into meaningful results (Tiwari et al., 2018).

Big data requires massive storage space due to its 5Vs characteristics; thus it makes use of cloud computing. Cloud computing provides large storage capacity and processing power, along with numerous services that include processing, computation, and storage capabilities for big data (Sandhu, 2022). The cloud computing infrastructure also provides the necessary storage requirements for big data analysis. Data processing in traditional environments differs from that of big data environments. The traditional approach to data processing involved data exploration, followed by model design and the creation of a database structure. Figure 2.4 illustrates the flow of big data. As illustrated, data processing begins with the collection of data from various sources such as the web, several files, and sensors. This collected data is warehoused in the “landing zone”, which offers capabilities of managing the volume, velocity, and variety of data. This is often a distributed file system. Once the data is stored, various alterations are made to the data to maintain its scalability and efficiency. After this stage, the data is combined into raw data extracts or databases, and analytical tasks and operational reporting are performed.

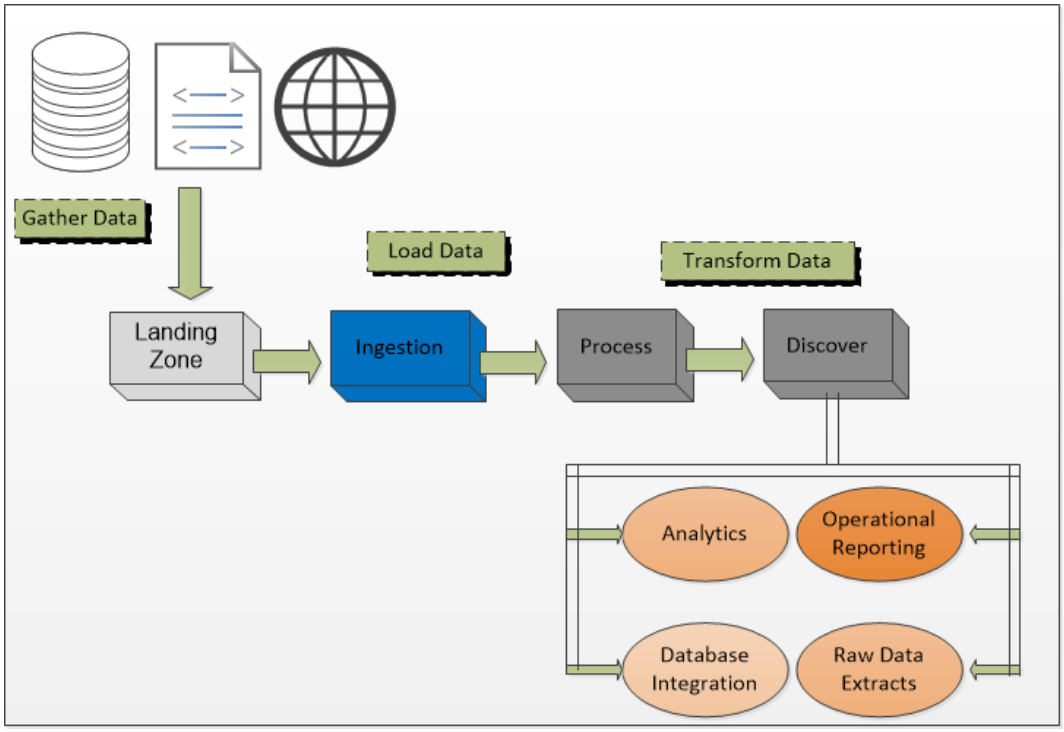


Figure 2.4: Flow in processing Big data (Abioye, 2023)

2.4.2 Types of Big Data Analysis

Data analysis is the process of transforming gathered data into meaningful information. Big data analysis consists of six main methods (Taherdoost, 2020):

1. Descriptive analysis is the first type of data analysis, encompassing various methods of analysis. It requires the least effort for analysis and is the most suitable method for large volumes of data. Hence, it is used to perform data analysis on a dataset.
2. Predictive analysis uses historical and current data to predict the likelihood of an occurrence. It can predict the value of one subject based on data from another related subject. Thus, the dataset used for prediction and the determination of the measuring variables are key to this analysis.
3. Exploratory- This method of analysis is used to determine changes in one variable by examining randomized trial data sets based on other variables to identify consequences.

4. Inferential- This analysis method uses datasets such as retrospective, observational, and cross-sectional time studies. A significant sample of data is used to draw conclusions about a larger population. This method tests the nature of a general theory based on data from a sample.
5. Mechanistic- This method of analysis requires the most effort because it uses randomized data sets to establish specific changes in variables, which can result in variations in other variables. Mechanistic analysis can be challenging to deduce; thus, it is the most suitable analysis method to utilize for situations where the highest precision in results and minimal error are required.
6. Explanatory or Causal- This analysis method is used to determine unknown relationships, establish new connections, and inform future research.

2.4.3 Security Challenges Associated with Big Data in Cybersecurity Networks

Big Data encompasses several risk areas, including the lack of security procedures, the data creation and gathering process, and the information lifecycle (derivation, classification of data, and ownership) (Moura & Serrão, 2019). However, risk areas identified as the most challenging are access control and data protection; not all data can be protected due to volume and velocity characteristics (Kim et al., 2013). Data classification and management pose a challenge due to the enormous heterogeneous sources that are structured, semi-structured, or unstructured (Rawat & Yadav, 2021). For Big Data to be leveraged to its full potential, organizations must address several issues related to Big Data, particularly those concerning policies regarding security, intellectual property, privacy, and liability (Toshniwal et al., 2015). Big data security objectives are the same as those of any data type to maintain confidentiality, integrity, and availability.

Cybersecurity networks are crucial infrastructures that generate big data (Wang & Jones, 2021). Big data in cybersecurity networks flows in real-time; this data is complex, constantly changing, and comes in different formats. This leads to veracity issues. Veracity in data analysis is cited as the biggest challenge compared to volume and velocity characteristics (Saeed & Husamaldin, 2021). Veracity refers to data quality,

encompassing inconsistencies, incompleteness, noise, and ambiguity (Hariri et al., 2019). Data veracity is normally classified as good, bad, and undefined. Veracity is typically low in big data due to the increasing diversity of sources and the variety of data (Hariri et al., 2019; Sandra, 2019). Veracity poses challenges to the analysis of big data datasets because of the uncertainty of data, provenance of data, and the noisiness of data noise (Assiri, 2020). Data uncertainty is a result of the veracity of the data collected from different sources. This data comes in different formats (structured, semi-structured, and unstructured) and high volumes that present unknown threats. Moreover, there is currently a limited body of literature comparing data collection methods and their impact on data veracity. Data provenance in data veracity is also identified as a challenge. Data provenance tracks and records data sources, usage of data, modification, the individual who accessed the data, purpose, data, and time (Reimer & Madigan, 2019; Abiodun et al., 2022). This is a complex task as data in cybersecurity networks is voluminous and transmitted in real-time, leading to high computational demands. Noisiness makes it difficult for machine learning algorithms to make accurate predictions.

Managing identities, preserving confidentiality, integrity, and availability, and detecting threats on cybersecurity networks is complex due to the vast amount of data.

2.4.4 Analysis of big data threats in cybersecurity networks

Machine learning and big data analytics have become powerful tools for alleviating ever-evolving cybersecurity threats (Alam, 2023). Machine learning is a branch of artificial intelligence that utilizes sample data to construct mathematical models that autonomously learn and improve from past experiences without requiring explicit programming. There are four main categories of machine learning, namely: supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning (Sarker, 2021).

1. Supervised learning- Learning in which a model is trained using labelled data to learn a mapping function that predicts input outputs. Supervised learning is further classified into two categories, regression and classification. Regression predicts a continuous variable, where the outcome of the prediction is a real value/

continuous value. In contrast, classification predicts a class or label, resulting in a categorical value.

2. Semi-supervised learning- Hybrid learning utilizes supervised and unsupervised learning. It utilizes both labeled and unlabeled data to train a model that makes accurate predictions.
3. Unsupervised learning- Learning in which a model is trained using unlabeled data to identify patterns and relationships without supervision to discover output. The model's output is predicted based on patterns derived from the input data. Unsupervised learning is further classified into two categories: clustering and dimension reduction. Clustering groups with similar data points and dimension reduction reduces the number of features.
4. Reinforcement learning- Learning in which an agent performs actions in an environment and learns from rewards, penalties, and errors. It uses insights drawn from the environment with which the agent interacts to increase the reward or reduce the risk.

Machine learning is an effective tool for processing and analyzing big data as well as developing models to predict and prevent cyber threats. However, anomaly detection using machine learning is challenging due to the nature of high-dimensional data that needs to be handled, the skewed datasets, and the need to identify patterns (Thwaini, 2022). The dimensionality challenge is curbed by feature dimensionality reduction and feature selection, which remove irrelevant and redundant information (Jia et al., 2022). Moreover, the level of accuracy in the analysis of big data is influenced by the type of machine learning technique chosen, the nature of the data, and the objectives to be achieved with such data.

Machine learning algorithms such as Support Vector Machine (SVM), Light Gradient Boosting (LGBM), K-Nearest Neighbors (KNN), and Random Forest can be used to enhance threat detection and prevention (Hasan et al., 2025). Table 1 illustrates the various machine learning algorithms and use cases in Intrusion Detection Systems.

Security log data has been used to detect anomalies; however, due to noise in the security log data that makes it challenging to analyze cyber threats, a paradigm shift in anomaly detection was necessary. Machine learning models have presented a solution; machine learning techniques such as regression, classification, and forecasting are key to detecting anomalies and deriving patterns to overcome cyber threats and identify advanced persistent threats (Hasan et al., 2025) . Random Forest model (LRM) is a machine learning approach that utilizes a supervised learning technique and is commonly used for classification purposes (Jurafsky & Martin, 2025) . Random Forest predicts the possibility of a target variable by finding the best suitable fitting model that predicts the possibility of an outcome based on a set of predictor variables (Salman et al., 2024). In the domain of machine learning and data science, classification has been cited as one of the best techniques for building prediction models (Alnuaimi & Albaldawi, 2024). According to Achshah and Prakash (2021), besides the easy implementation, interpretation, and training of data to quickly classify unknown data in Random Forest. There are other reasons for the selection of Random Forest, these include:

- Excels when there is a linear correlation between the input and target variables.
- Works under the condition that the number of parameters is less than the number of samples; this consideration helps it overcome the challenge of overfitting.
- Works well irrespective of the distribution of the input variable and target variable, capable of handling a wide array of data.
- Determines the relevance of the predictor and whether its impact is positive or negative.

There are three types of Random Forest models: binary Random Forest model (BLR), multinomial Random Forest model (MLR), and ordinal Random Forest model (OLR). According to Kilincer et al. (2021) machine learning models can learn from historical data to effectively identify known and unknown threats. Barbierato & Gatti (2024) believe that a model has mastered an algorithm to perform a specific task when it can learn patterns or relationships from datasets that were not part of the original dataset the model was trained on. Furthermore, determining whether a machine learning model has “mastered

a task” depends on a specific context and the model’s objective. The table below lists the general attributes that a model has learned.

Table 2.1: List attributes indicating a machine learning model has learned.

Table 2.1: Barbierato & Gatti (2024)

Benchmark	Description
Accuracy	The model should be able to make accurate predictions or classifications on unseen data. This means that the model should generalize well beyond the training data it was exposed to.
Generalizability	The model should not be overly specific to the training data and should be able to perform well on data from different sources or with different distributions.
Robustness	The model should be resistant to noise and outliers in the data and should not be easily fooled by adversarial examples.
Interpretability	The model should be understandable and explainable, allowing us to understand how it makes its decisions and identify potential biases or limitations.
Efficiency	The model should be able to train and make predictions efficiently, especially for large or complex datasets.
Relevance	The model should be relevant to the task at hand and should address the specific problem or question being posed.
Novelty	The model should provide new insights or solutions that were not previously known or available.

Impact	The model should have a positive impact on the real world, either by solving a problem, improving a process, or making a decision.
Scalability	The model's ability to maintain performance as the size of the dataset increases.
Fairness	Ensuring the model does not create or reinforce unfair bias against certain groups.
Transferability	The ability of the model to adapt to different tasks or domains with minimal adjustments.
Compliance	Adhering to legal and ethical standards, especially in sensitive areas like healthcare or finance.
Sustainability	Evaluating the environmental impact of training and deploying the model, such as energy consumption.
Security	Ensuring the model is resistant to attacks and does not expose sensitive data.
User Experience	The ease with which end-users can interact with the model and its outputs.
Reproducibility	The ability for other researchers or practitioners to recreate the model and achieve similar results.

Machine learning can potentially mitigate cyber threats in cybersecurity networks, but it is not without drawbacks. The scalability and interpretability of machine learning models are a challenge (Nassar & Kamal, 2021). As more cyber threats are introduced and evolve in the cybersecurity landscape, the volume of data processed for threat detection also grows at a rapid rate. Additionally, it ensures that big data infrastructure and machine learning models can manage continuously increasing data flows without compromising response time. Furthermore, to address this challenge, considerable investment and optimization

efforts are required to scale up infrastructure and resources. The lack of interpretability in machine learning models, their inability to fully understand the internal workings, and the determination of outcomes further exacerbate cybersecurity challenges. Transparency and explainability are core in cybersecurity; the ability to explain and substantiate why a model reached a specific decision can be challenging. According to Jurafsky and Martin (2025) and Mohamed et al. (2023), the features of Random Forest are built in a human-readable format, and a classifier's decision is understood by the role each feature plays in the decision. Random Forest can also be combined with statistical tests, which conduct investigations on characteristics such as features and magnitude, which assist with the interpretation of why a classifier reached a decision.

2.5 Related Work

This section discusses related works that have been conducted in the context of data analysis in cybersecurity networks.

Mohamed et al. (2023) conducted a study on the detection of cyber threats in big data. The authors indicate a need for sophisticated classification techniques and the latest datasets to enhance detection in intrusion detection systems. Previous studies in intrusion detection have identified limitations, including the use of obsolete datasets that may inaccurately reflect current cybersecurity threats and the disregard of various types of attacks due to a focus on binary classification alone. These limitations highlight the need for current studies to employ advanced classification techniques in determining different types of cyberattacks. This study proposes an anomaly-based intrusion detection system using the CSE-CICIDS2018 dataset (Communications Security Establishment (CSE), 2018). The system assesses machine learning algorithms, including Convolutional Neural Networks, Random Forest, Deep Neural Networks, and XGBoost, in binary and multiclass classifications. The methodology includes data preprocessing steps such as feature selection with Random Forest, hyperparameter tuning with Optuna, and ensuring dataset balance through stratified sampling and model evaluation. The main findings of this study indicate that Random Forest classifiers outperform all other classifiers, achieving an accuracy of 99%, an F-1 score of 0.97, and a recall of 93%. Thus, the performance of

intrusion detection systems is enhanced by combining big data and deep learning techniques. The authors recommend that future research focuses on assessing models on more datasets, exploring other types of classification models such as long short-term memory networks and recurrent neural networks, developing new cybersecurity datasets, and comparing performance across various data types.

Malathi & Padmaja (2023) conducted a study to examine various machine learning algorithms that identify attacks on Internet of Things (IoT) networks. The study utilized the Bot-IoT dataset to evaluate the effectiveness of various detection techniques. The Bot-IoT dataset has different cyberattacks and real and simulated Internet of Things network traffic; the attacks include denial of service, probing (social engineering, vulnerability, and port scanning), and data stealing (phishing, malware, SQL injections, zero-day attacks, and man-in-the-middle). Their study leveraged seven machine learning classifiers to evaluate the Bot-IoT dataset. The classifiers included Nave Bayes (NB), Nearest Neighbors (KNN), Multilayer Perceptron (MLP), Random Forest, Quadratic Discriminant Analysis (QDA), Iterative Dichotomiser 3 (ID3), and AdaBoost. They tested several attacks that included DDOS_HTTP, DDOS_UD, DDOS_TCP, DOS_HTTP, DOS_UDP, DOS_TCP, Data extraction, keylogging, service scan, and OS scan. Results indicated that all algorithms from the seven classifiers successfully identified all various attacks at a 90% success rate, except for the Quadratic algorithms and Naive Bayes.

Malathi and Padmaja (2023) recommended that future work analyze unsupervised learning algorithms and future detection algorithms by integrating different machine learning techniques into a multi-layered model. Although the goal of the authors varies from that of the current study, the methodology to be used in this study follows the recommendation of using machine techniques to identify security threats in big data patterns flowing through cybersecurity networks.

Pathade & Bhosale (2023) conducted a study to develop a model for predicting cyber threats using various methodologies to generate more results for cyber threat prediction. The study identified eight of the most common cyber threats: man-in-the-middle, malware, phishing, denial-of-service attacks, SQL injection, ransomware, DNS attacks, and zero-day exploits. Various prediction methods were employed, and the most effective technique was selected based on the dataset's size. The prediction methods used for this study were: multiple linear regression, k-Nearest Neighbors (kNN), regression tree, CART algorithm, k-means clustering, and Random Forest. Model selection was done to determine the best algorithm. Model selection was conducted by evaluating various factors such as the resampling method and probabilistic measure, which yielded high prediction results for model performance. The authors recommend that datasets flowing through cybersecurity networks should be organized, sanitized, and contain minimal biases for enhanced model performance and accurate prediction results. The current study is similar to that of Pathade & Bhosale (2023) in that it will use a machine learning classification technique to identify threats in big data flowing through cybersecurity networks.

Sarker (2023) conducted a literature review on machine learning algorithms and methodologies, focusing on how they can be used for intelligent data analysis and automation in the field of cybersecurity to derive meaningful insights from cyber data. Spam detection, malware detection, and intrusion detection are the common datasets available in cybersecurity. The commonly used and available datasets in cybersecurity are on the Internet: KDD'99 Cup dataset, NSL-KDD, CAIDA'07, ISCX'12, Bot-IoT, and CTU-13. These datasets include feature attributes, class identification, and types of cyberattacks. The attacks are categorized as: user-to-user intrusions (U2R), denial of service (DoS), remote-to-local (R2L) intrusions, and distributed denial of service (DDoS), to mention a few. Sarker (2023) identifies factors that affect the efficiency and effectiveness of machine learning solutions; these include the nature and quality of data, the learning algorithm, and data gathering from sources (networks, endpoints, and cloud). Furthermore, it is challenging to standardize the gathered data, especially historical data,

which presents issues such as numerous outliers, ambiguous values, missing data, and other insignificant information. The author discovered factors that make data unsuitable for learning, such as irrelevant features, non-representation, low quality, and limited data for training. The author established a need for automated methods for detecting risks and malicious user behavior as a gap that machine learning can fill. The author further proposes cyber-threat intelligent systems that can forecast critical scenarios and consequences instead of organizations relying on mitigation and defensive measures. Saker also suggests a regression model is valuable for forecasting cyberattacks or forecasting the impact of an attack, such as malicious software, viruses, worms, etc. he current study is similar to that of Sarker (2023) in such a way that the goal of the current study is to determine the factors that influence big data analysis in cybersecurity networks.

García Lozano et al. (2020) conducted a study on the veracity assessment of online data. The authors review different algorithms, approaches, methods, and tools used for the automatic assessment of the veracity of open-source data. The authors indicated that veracity assessment often utilizes indicators at the end of the assessment. Two main indicators, known as dimensions, were determined. The first indicator dimension involves the origin of data, with indicators derived from external sources, message contents, and metadata. The second indicator dimension relates to algorithm development for approval or disapproval of veracity. Furthermore, the findings of this research indicate that supervised learning on text data is the commonly used approach, and a limited number of papers use deep learning approaches. Moreover, veracity assessment is a complex challenge involving various data types, sources, methods, and indicators. However, most papers cover a limited focus, which indicates that limited research has been done in the veracity assessment area. This study employed a systematic literature review that followed a search strategy whereby specific keywords were searched across various databases. Gaps have been identified in this study, these include: limited methods and approaches for dealing with numerous data sources and type, no common definition for core terminology related to veracity and veracity assessment, challenge in the

reproduction of results due to lack of details, limited datasets suitable for benchmarking, underuse of deep learning and transfer learning, and limited scalable online methods for streaming data. The authors recommend that future research focus on improving the identified gaps mentioned above. The current study is similar to that of García Lozano et al. (2020) in that it will use a machine learning classification technique to handle large-scale data from multiple sources and types.

2.6 Summary

This chapter reviewed existing literature on big data analysis relevant and applicable to this study. The chapter begins by discussing the concepts of cybersecurity and data flow in cybersecurity networks, as well as the methods and techniques used to analyze threats. It also covers big data analysis, threat analysis in cybersecurity networks, and related work that demonstrates advancements in the field of big data analysis. The literature indicates a gap in dataset obselence and the need for models that analyse big data in real time.

CHAPTER THREE: RESEARCH DESIGN AND METHODOLOGY

The aim of this study is to develop a big data analysis model for futuristic insights into cybersecurity networks. This chapter outlines the research design and methodology adopted. The study followed the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework for model development, as it is widely adopted for data mining projects and knowledge discovery. To gain a clear and in-depth understanding of the research problem, related literature was reviewed. The positivist paradigm was adopted because it enabled the researcher to derive relationships and associations from pure data to forecast malicious data that may lead to cyberattacks, thereby achieving resilient cybersecurity. Through a quantitative approach, this knowledge was derived from secondary data that mimics the flow of data in a cybersecurity network environment. Furthermore, an experimental research strategy was adopted for this study to gain an in-depth understanding of big data in cybersecurity networks. Moreover, the positivist paradigm, quantitative approach, and experimental research underpin the CRISP-DM framework due to the empirical work involved in developing the model. Finally, the data was analyzed using machine learning techniques to draw patterns and develop a predictive model to enhance cyber threat detection.

3.1 Research Approach

A research approach is a set of research methods and strategies that span decisions from general beliefs to clear methods of data collection and reasoning (Creswell, 2009). This study employs quantitative research that utilizes a deductive approach. Quantitative research involves gathering and analyzing numerical data using mathematical methods (Kandel, 2020). Quantitative research generates explanatory relationships and associations from statistical data that forecast and control the phenomena in question. A quantitative research method was chosen for this study because it enables the researcher to develop and test models or hypotheses relating to phenomena, which provide insights into the relationships between measurable variables with the goal of explaining, forecasting, and controlling a phenomenon or situation. The quantitative approach provided this research with numerous advantages, including, but not limited to, the

following (Creswell, 2009): statistical analysis capabilities, which aid in determining variables with strong predictive power to the target variable. Moreover, outcomes in quantitative research are unbiased and fair because standardized metric calculation and feature selection functions validate libraries such as scikit-learn. metrics and sklearn.feature_selection from Python, which are free from the researcher's bias (Pedregosa et al., 2011).

3.2 Research Paradigm

A research philosophy refers to the set of principles and theories about knowledge development (Phair & Warren, 2021). On the other hand, a research paradigm refers to the shared views and beliefs among researchers on how research problems can be perceived and addressed (Kumatongo & Muzata, 2021). This study followed the positivist paradigm. This paradigm was chosen for this research because it relies heavily on quantitative methods, such as experiments and statistical tests, which enable the interpretation and generalization of findings. Additionally, to align with the positivism paradigm, the hypothetico-deductive model of science was followed, which is a "process that begins with theory from the literature to build testable hypotheses, design an experiment through operationalizing variables (i.e., identifying variables to manipulate and measure through group assignments), and conduct an empirical study based on experimentation" (Park et al.,2020).

3.3 Research strategy

A research strategy is a step-by-step guide for carrying out research. It serves as a roadmap for the researcher by addressing the research aspects of planning, implementing, and observing the study (Johannesson & Perjons, 2014). This study used an experimental research strategy. Experimental research is an empirical study that intends to examine cause-and-effect relationships between variables. The goal of the experiment is to explore big data patterns that will aid in the early detection of cyber threats, examining the capacity and limitations of a big data analysis model in cybersecurity networks by addressing specific research questions. This study utilized

secondary data, which is crucial for this experimental strategy. The secondary data was utilized because it was challenging to obtain primary data due to its sensitivity and the data protection laws and regulations that govern the use of data of this nature. Thus, secondary data was constructed to replicate real-world cybersecurity network data, providing secure and ethical conditions and settings for testing the capacity and limitations of the model. Furthermore, using secondary data was beneficial in overcoming logistics challenges related to using real network data such as data privacy and security, data anonymization, data collection, data processing and converting it into standard format, evaluating data quality and biasness, labelling the data, ensuring compliance to data protection laws while still drawing complex patterns and actual features of cyber threats (Shani et al., 2022) Moreover, generating artificial data was more inexpensive and time-saving than gathering and analysing real network traffic from a significant number of organizations (James et al., 2021). Artificially generated data can create complex scenarios and patterns of attack that may be uncommon or challenging to detect in real-time. According to Mertler (2016), the following are the appropriate steps to follow when conducting experimental research:

1. Identifying and establishing the research problem: This step involves determining a research problem and using empirical rationale to support the need for a solution. According to Abrahams et al. (2024), traditionally accepted reactive approaches are widely used for cyber threats that are detected. These approaches involve responding to malicious activities after they are detected or occur and are no longer sufficient to protect against the evolving, sophisticated threats. Thus, the primary objective of this study is to develop a big data analysis model for gaining futuristic insights into cybersecurity networks, which represents a proactive approach to cyber threat detection.

2. **Review related literature:** This step involves reviewing related literature to determine what has been done in the field of study, the recommendations, and the gaps that need to be addressed. These serve as guidance for identifying features that support the secondary data, as well as a baseline for selecting machine learning models and evaluating model metrics suitable for the study.

3. **Detailed description of design and procedures for data collection:** This step involves providing details on how the study was conducted and how the data were collected. In this study, secondary data were used in the development of a predictive model for cyber threat detection. The model will be developed using Python.

4. Data collection: This step involves determining sources of data, which could be primary or secondary data sources. In this study, secondary data were used for model development.

5. Analysis of data: This step involves analyzing the data collected in the previous step. Analysis of the data involves exploring the datasets and understanding their structure for preprocessing purposes. The preprocessing of the data includes: data cleaning, which involves handling missing values to ensure the data is complete without introducing bias, encoding target columns to avoid data mismatch errors, transforming skewed distributions to reduce skewness, ensuring better distribution, and performing correlation analysis to identify which features are most strongly related to the target for feature selection. Finally, classification reports with detailed evaluation metrics, such as precision, recall, and F1-score, were used to identify the best-performing algorithm based on the dataset.

6. Answering research questions through the research objectives: This step involves answering the research questions and interpreting the findings. To answer the research questions, various methods are employed as outlined in the table below.

Table 2: Research objectives and data collection methods

RO	Research Objectives	Data Collection
1	To determine the factors that influence big data analysis in cybersecurity networks.	Literature Review
2	To establish methods and techniques that have been used to analyze security threats in cybersecurity networks.	Literature Review

3	To use machine learning classification techniques to identify big data patterns in cybersecurity networks.	Statistical analysis
4	To use the identified patterns to develop a predictive model for cybersecurity networks.	Statistical analysis

The evaluation metrics of the model will provide results on its overall performance.

3.4 Research Methodology

Research methodology is an approach to explaining the research problem. It provides details of how the research is to be conducted, examining and explaining the methods (Swarooprani, 2022). Data mining research employs various research methodologies, including Knowledge Discovery in Databases (KDD), Sample, Explore, Modify, Model, Assess (SEMMA), and Cross Industry Standard Process for Data Mining (CRISP-DM). CRISP-DM integrates both KDD and SEMMA principles and ideas from other processes, making it a robust methodology in data mining (Shimaoka et al., 2023). Thus, it was adopted in this study due to its robustness and dominance in the fields of data science and data mining. While CRISP-DM originated as an industry framework, it has been widely adopted in academia for its structured, easy-to-follow methodology for conducting data mining and predictive modeling projects (Wirth & Hippo, 2000).This study adopted the CRISP-DM framework as the underlying framework because it is widely used in data mining for process modeling. It comprises six iterative phases from business understanding to deployment (Schröer et al., 2021). The six phases of CRISP-DM are illustrated in Figure 3.1.

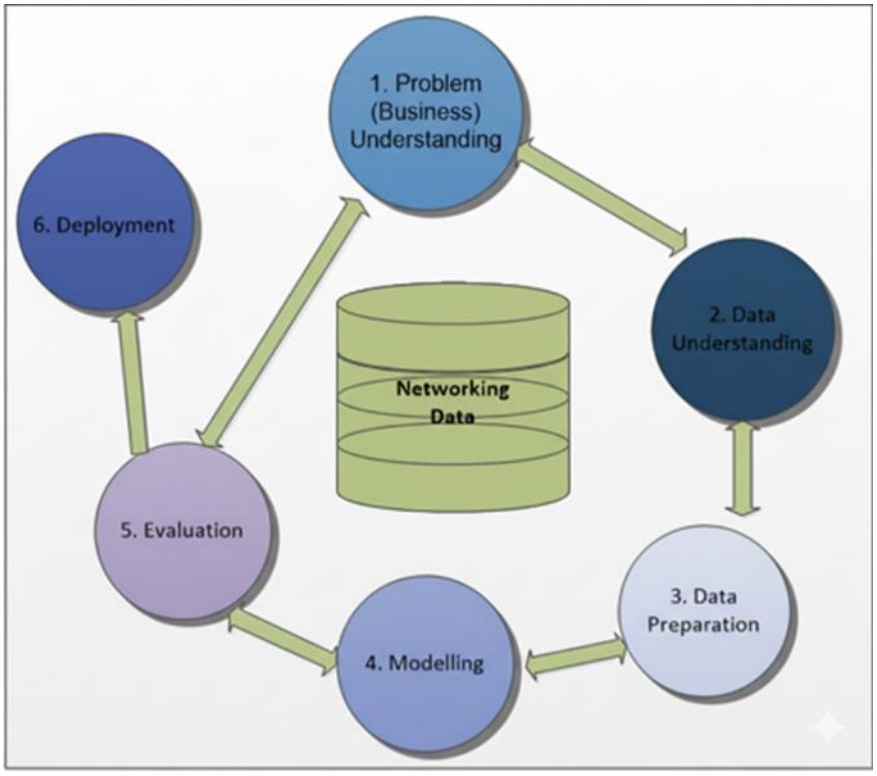


Figure 3.1: Six phases of CRISP-DM (Chapman et al., 2000)

3.4.1 Phase 1: Business Understanding

Business Understanding is a fundamental phase that focuses on understanding the study's objectives. Section 3.3 in the first step of the experimental research process clearly highlights the research problem of reactive traditional approaches in cyber threat detection and the need for proactive approaches for cyber threat detection through the development of a Big Data Analysis Model for Futuristic Insights in Cybersecurity Networks. Additionally, with the aid of the specific research objectives of this study, the challenge of cyber threat detection is better understood. The following includes the specific research objectives and their role in cyber threat detection:

- **Determining the factors that influence big data analysis in cybersecurity networks**: These factors were identified in the literature and provided an overview of challenges in big data analysis, as well as guidance on the fundamental factors in model development for threat prediction.

- **Establishing methods and techniques used to analyze security threats in cybersecurity networks**: The methods and techniques were determined through literature and served as a guideline on what has been done to detect cyber threats in cybersecurity networks; these methods were mostly reactive in nature. Proactive approaches, such as existing machine learning approaches, provided a baseline for comparison for acceptable model performance and algorithms in the specific domain.
- **Using machine learning classification techniques to identify patterns in big data cybersecurity networks**: The existing literature provides knowledge on existing algorithms, which serve as a baseline for comparison to determine the best-suited algorithm for the classification challenge identified and to better detect cyber threats through big data patterns
- **Using the identified patterns to develop a predictive model for cybersecurity networks:** Based on the chosen machine learning techniques, a model is trained and built to identify cyber threats using unseen data. This model can then identify patterns that indicate whether the data flowing through the network is malicious or benign.

In conclusion, the big data analysis model for cyber threat prediction is considered effective due to its ability to accurately identify patterns and predict cyber threats, providing futuristic insights by utilizing unseen data and the learned patterns associated with malicious activity.

3.4.2 Phase 2: Data Understanding

The data understanding phase builds upon the business understanding phase, focusing on the identification, collection, and analysis of datasets. This phase involves four tasks:

- Collecting initial data: This study used purposeful sampling to select data features. The sampling technique was chosen because there are limited and relevant datasets on the subject matter. Thus, purposeful sampling enabled the researcher to identify and select the most relevant datasets, providing appropriate and meaningful information (Campbell et al., 2020). Furthermore, the data features that

were selected were found to be most relevant to predicting cyber threats in the secondary data. This study used secondary data to develop the predictive model. These datasets were synthetically generated to mimic real-world data that consists of cyber threats in cybersecurity networks. Synthetic datasets were chosen for this study because they encompass current threats and circumvent the data protection regulations and privacy challenges associated with real-world data. Furthermore, synthetic data offers advantages in machine learning, such as data augmentation for robustness, private data release, and data de-biasing and fairness (Jordon et al., 2022). Table 3 illustrates the first ten original datasets.

Table 3: Original CSV dataset

	traffic_volume	packet_size	Protocol	duration	login_attempts	cpu_usage	is_malicious
1	7370	1480	UDP	1963	9	0.4064409133570098	0
2	960	900	FTP	2401	9	0.739466408352222	0
3	5490	634	UDP	368	0	0.3359650726739756	0
4	5291	1238	HTTPS	884	9	0.5489774865932489	0
5	5834	432	HTTPS	1731	0	0.7054910372861605	1
6	6365	1194	HTTP	1256	2	0.1444539074947394	0
7	566	1226	FTP	595	2	0.9802286725134809	0
8	4526	343	TCP	1643	0	0.1182901597801151	0
9	5678	135	HTTP	3326	2	0.502784474730247	0
10	8422	686	HTTP	77	0	0.253093197402495	0

- **Describing the data:** The secondary data was an artificially generated dataset; this dataset was generated with 7 parameters, such as traffic_volume,packet_size, protocol, duration, login_attempts,cpu_usage, and is_malicious.The traffic_volume, packet_size,duration,login_attempts,cpu_usage, and is_malicious. The traffic_volume,packet_size, duration,login_attempts, and is_malicious were integer data types, while cpu_usage was a float type, and protocol was an object

type. These features were used to generate 50,000 records of the dataset. The size was chosen to serve as a representative sample for experimental validation while maintaining computational efficiency. The dataset had no missing values, and all values were numerically encoded except the protocol, which was encoded categorically as nominal data. Label encoding protocol had to be performed on the protocol feature because machine learning models perform effectively with numeric values (Mohammed, 2023). Thus, label encoding was done to assign a unique integer to each category.

- **Exploring the data:** Exploratory Data Analysis (EDA) is the preliminary step that is taken in the data analysis process (Sahoo et al., 2019). Data exploration was done to understand and identify the characteristics and features of the secondary data. Exploratory data analysis involves using descriptive statistics to provide an overview of each feature, including the count, mean, standard deviation, minimum, and maximum values in the columns, as well as quartiles (Johnkutty et al., 2024). The class distribution of the dataset was imbalanced, with 47518 instances as benign and 2482 instances as malicious. Furthermore, visualization was performed using Python's visualization libraries, such as Seaborn and Matplotlib, to illustrate the distributional properties and relationships between features through bar charts. While explaining the distributional properties and relationships between features was done, it was also crucial that features with a strong relationship with the target variable or another feature were identified. This was accomplished using correlation analysis, a type of analysis widely employed to assess the strength of a relationship between features (Kumar & Chong, 2018).
- **Verify data quality:** Data quality refers to the qualitative and quantitative properties of data, which are characterized and measured to ensure that they are fit for the purpose of a task (Cheng & Wang, 2022). According to Zhou et al. (2024), the performance, scalability, fairness, safety, and robustness of machine learning models are significantly influenced by the quality of data used during training. This study ensured that the training data is of high quality by addressing the four most

important data dimensions to fit the purpose of the classification machine learning model. The four important data dimensions include:

- **Completeness/Missing Values Dimension:** An analysis of missing values was conducted through data exploration, examining data features/ columns, data types, and counting missing values for each record and feature using the .isnull().sum()method. Furthermore, undefined values or non-numeric features were checked, and medians were used to impute any missing values for unseen data. Finally, categorical encoding was applied for any features that were not encoded numerically.
- **Relevance/Feature Selection Dimension:** Relevance/ feature process of selecting features that are relevant to the task. This study leveraged correlation, feature engineering, feature importance using Random Forest Classifier and Gradient Boosting Classifier, Mutual Information Importance, and removal of low-variance features to select the most relevant features to the study, those with the most predictive power in relation to the target variable, while discarding irrelevant features.
- **Class imbalance:** Refers to classes where data distribution is imbalanced, which produces skewed and biased results. This study addresses this challenge by applying the Synthetic Minority Over-sampling Technique (SMOTE) by creating synthetic minority samples. Initially, the dataset was imbalanced, 95% being malicious and 5% of the instances being benign. After SMOTE, the distribution was balanced, 50% of the instances were malicious, and the remaining 50% were benign.
- **Remove duplicates:** Refers to the existence of the same file copies that create a storage inefficiency challenge. It is essential to remove duplicates from the dataset to prevent redundancy, improve the model's training and performance, and minimize the inaccuracy of results. This study removed duplicates to ensure optimal model performance by using the Pandas built-in function' df.drop_duplicates() ' to distinguish and remove duplicate rows or columns in a dataset.

3.4.3 Phase 3: Data Preparation

This phase involves preparing the final dataset for modeling and involves five tasks:

- **Selecting data:** This step involves determining which datasets to use. This study utilized secondary datasets, which were synthetically generated to circumvent data protection regulations and privacy concerns associated with real-world data. Furthermore, the choice to utilize secondary datasets was influenced by the study referenced in the related work by Mohamed et al., 2023. Mohamed et al. (2023) noted that studies in intrusion detection often utilize outdated datasets, which may inaccurately reflect current cybersecurity threats. Thus, synthetic datasets were generated to mimic real-world data that consists of current cyber threats in cybersecurity networks. Using secondary data was beneficial in overcoming logistics challenges related to using real network data, such as data privacy and security, data anonymization, data collection, data processing, and converting it into a standard format, evaluating data quality and bias, labelling the data, ensuring compliance with data protection laws, while still drawing complex patterns and actual features of cyber threats (Shani et al.,2023) . Moreover, generating artificial data was more inexpensive and time-saving than gathering and analysing real network traffic from a significant number of organizations (James et al., 2021). Artificially generated data can create complex scenarios and patterns of attack that may be uncommon or challenging to detect in real-time.
- **Cleaning data:** This step involves identifying and rectifying errors, anomalies, and inconsistencies in the datasets. Data cleaning was performed in this study by handling missing values, encoding categories, and encoding the target column, transforming skewed distributions, conducting correlation analysis, and selecting features. This was done to ensure that the data is accurate, reliable, and fits the purpose of classifying cyber threats. The following commands were used to clean the data.

- **Constructing data:** This step involves deriving new features that are strongly related to the target variable. In this study, feature engineering was performed to generate new features using existing ones. New features included:cpu_traffic_ratio,cpu_traffic_interaction,login_duration_ratio,and,login_duration_interaction.
- **Formatting data:** This step involves reformatting data. This was done by categorically encoding the data, specifically the protocol and target variable, to ensure that the model performs effectively.

3.4.4 Phase 4: Modelling

This phase involves building and assessing various models using different algorithms. This phase involves four tasks:

- **Select modeling techniques:** This step involves determining which algorithms will be used for the model. The model built addresses a classification task. Thus, the algorithms tested were Random Forest Classifier, Gradient Boosting Classifier, KNeighbors Classifier, Logistic Regression, Decision Tree Classifier, Gaussian Naive Bayes, and Multilayer Perceptron Classifier to determine the best-performing algorithm for building the model.
- **Generating test design:** This step involves splitting the data into a training, testing, and validation set. In this study, the train size was set to 80% and the test size to 20%. According to Sivakumar et al (2024), the training set typically requires a larger portion of data to learn more effectively. Finding a balance between training and test sets is key to avoiding issues of underfitting and overfitting. Underfitting occurs when a small portion of the training dataset is learned by the model, and overfitting occurs when the entire training dataset is learned well by the model but underperforms on unseen data. To overcome the challenges of overfitting and underfitting, stratified sampling was applied in the train-test split to ensure that the distribution of the original data is well-represented in both the training and test datasets, thereby reducing bias.

- **Evaluating the Model Algorithm for Model Selection:** This step involves comparing the performance of different models and selecting the best one to build the final model. This step involves deriving new features that are strongly related to the target variable. In this study, feature engineering was performed to generate new features using existing ones. New features included:cpu_traffic_ratio,cpu_traffic_interaction,login_duration_ratio,and,login_duration_interaction.
- **Building model:** This step involves training and fitting the model. The model was developed using Python's scikit-learn libraries. This study used. fit (X, y), which takes the input features and target variable to find patterns and make predictions.

3.4.5 Phase 5: Evaluation and Parameter Fine-Tuning

This phase involves assessing the model results. Assessing model results involves reviewing the performance of the algorithm employed and selecting the best-performing model that effectively addresses the classification challenge. Hyperparameters play a crucial role in determining the model's performance (Ilemobayo et al., 2024). It significantly improves the model's performance and generalization. Moreover, it helps maintain a balance between bias and variance, thereby improving the model's performance on unseen data. There are various hyperparameter tuning methods, such as Random search, Quasi-random search, and Grid search (Franceschi et al., 2024). For this study, Grid search was the chosen hyperparameter optimization technique in the Python Scikit-learn library; it performs an in-depth exploration, which entails testing all the combinations within a defined grid to determine the best hyperparameters (Belete & Huchaiah, 2022). Grid search can be described as an exhaustive exploration method, or a brute force approach, that tests all combinations of hyperparameters given to the grid configuration. Grid search is widely used in machine learning as a hyperparameter tuning method due to its reliability, low dimensionality, and ease of implementation. Furthermore, Grid Search Cross-Validation (GSCV) was used for cross-validation. Cross-validation prevented overfitting, ensuring that the model not only memorizes the training data but also performs well on real-world, unseen data (Adnan et al., 2022).

3.4.6 Phase 6: Deployment

This phase involves making the model available to carry out business applications for the end user. In this study, the model will not be deployed in real-world networks or simulated due to ethical considerations of security and privacy regarding data. Adherence to data protection laws is crucial for the protection of sensitive data. Furthermore, there was limited time to carry out this study. Deployed machine learning models require continuous monitoring, which is an iterative process that involves identifying any deviations in data patterns that could negatively affect the model's accuracy over time (Mahdi, 2024). Furthermore, it presents compatibility and scalability challenges in systems where integration may be resource-intensive, complex, expensive, and time-consuming (Sinha & Lee, 2024). Although this study is experimental, its application is explored theoretically based on developed models, and as such, it does not involve real-world deployment. Thus, for the purpose of model interpretability, an interface was designed for the model using Streamlit. Streamlit is a Python library that enables the development of user-friendly interfaces for data visualization in various formats and deployment for user access.

3.5 Reliability and Validity of Research

To ensure the integrity and validity of this study, the dataset underwent a thorough analysis, including data cleaning and validation, to remove inconsistencies, errors, and noise. On the other hand, the validity was determined using face validity, construct validity, criterion validity, and content validity.

3.5.1 Validity

Bellamy (2015) defines validity as a measurement of how well an instrument measures what it is expected to measure. The purpose of validity is to justify the accuracy of research findings (Zohrabi, 2013). According to Kubai (2019), validity is measured using both theoretical and empirical evidence. Theoretical assessment involves the operationalization or transformation of an idea into a construct, while empirical assessment utilizes statistical techniques and quantitative analysis. Generally, validity and reliability are interlinked; they are fundamental assessment qualities for determining

the validity of measurement scales (Bajpai & Bajpai, 2014). For this study, the following techniques and methods were used to ensure the validity of the measuring instrument;

a) Face validity

Face validity, also referred to as the "appearance of validity", suggests that a test which is to be used in real-world situations should not only focus on practical outcomes or functionality or statistical validity, but it should also appear practical, applicable, and align with the purpose of the test as well (Mosier, 1947). According to Allen et al. (2023), a tool's design relates to whether it is suitable and applicable to its intended domain. Face validity was ensured in this study by selecting features that are appropriate and relevant to the cybersecurity domain. The selection of parameters was informed by the literature, ensuring that the model's internal logic correlates with the real-world characteristics of malicious network traffic. Thus, the model demonstrates high face validity because its predictive framework is informed by established domain-specific characteristics, ensuring that the classification outputs are applicable and insightful to the cyber threat detection task.

b) Construct validity

Construct validity refers to the extent to which ideas, concepts, or behaviors that are constructs are transformed into functional and operational realities (Taherdoost, 2018). Construct validity involves determining the relationship between variables, the connection between the variables, with a focus on the cause-and-effect behaviors or constructs that comprise the relationship, and how one variable may change or affect another. Essentially, it is used to evaluate if the construct measured precisely corresponds to and affects the behaviours or outcomes it is expected to influence; that is, a test accurately measures a construct as expected (Taherdoost, 2018). The construct validity was ensured in this study by utilizing measurement instruments such as model evaluation metrics, correlation analysis, feature importance, and mutual information importance.

c) Criterion validity

Criterion validity, also known as instrumental validity, is used in tests to assess the relationship between scores and a highly rated existing standard instrument (Roy et al., 2023). Criterion validity statistically tests new measurement methods against a standard or independent criterion or future standards (Bellamy, 2015). Criterion validity provides an estimate that indicates the extent to which a measure aligns with an external criterion (Babbie, 2013). This study ensured conformity to criterion validity by assessing the performance of other models in research similar to the current study. Furthermore, the parameters for the artificially generated data were derived from real-world data informed by previous research similar to the current study.

d) Content validity

Content validity is defined as "the degree to which individual items capture the theoretical content domain of a construct" (Nunnally & Bernstein, 1994). In essence, a measure conforms to content validity if it comprehensively represents a construct of interest. It involves ensuring that all important items are included and the unsuitable items for a specific construct domain are excluded in the evaluation of a new survey instrument (Lewis et al., 2005). In this study, to ensure conformity to content validity, instruments that were previously used by other researchers were adopted and altered to align with this study.

3.5.2 Reliability

Drost (2011) defines reliability as "the extent to which measurements are repeatable when different people measure on different occasions, under different conditions, supposedly with alternative instruments which measure the construct or skill." It essentially involves replicating research findings under specific conditions. The most appropriate method to determine reliability is to measure the correlation between items, tests, and raters by calculating the reliability coefficient (Rosenthal & Rosnow, 2008).

3.6 Ethical Considerations

Ethics are defined as what is morally considered correct or wrong in a human's behavior or actions (Bartneck et al., 2021). According to Bos (2020), the principle of confidentiality in research ethics is fundamental; it involves the researcher ensuring that any use of information gathered from or disseminated to human participants does not violate the rights or interests of the participants, their communities, or the individuals of interest. Confidentiality was ensured in this study by using anonymized secondary data, which was anonymized for data privacy, security, and adherence to data protection laws. Moreover, this research study obtained ethical clearance from the University of Mpumalanga Research Ethics Committee (UMP REC) prior to commencing the research. All ethical guidelines, as per the UMP REC, were strictly adhered to during the data collection, analysis, and dissemination of the findings.

a) Confidentiality and anonymity

This research study ensured that the secondary data was anonymized. The study did not include personal identifiable information or human participation to avoid data privacy and security challenges, as well as to ensure adherence to the Protection of Personal Information Act (POPIA) and data regulations laws.

b) Ensuring that permission is obtained

The School of Computing and Mathematical Sciences and the University of Mpumalanga (UMP) Research Ethics Committee formally approved the proposal of the study, and all processes in the study were carried out in accordance with accepted ethical standards.

3.7 Summary

This chapter discusses the research methodology adopted to explore the identified research problem. The methodology integrated the positivist paradigm with the CRISP-DM framework. The adoption of the experimental strategy was key in addressing the research problem of cyber threats in cybersecurity networks. By following the six phases of CRISP-DM, the business objectives were established, and the features with the most positive correlation and predictive power were used for model development. The

implementation of feature engineering highlighted a key finding: modern threats contain malicious patterns that can be detected using the Random Forest algorithm, which can capture non-linear features and handle high-dimensional data. Finally, the chapter discussed the ethical considerations that the study adhered to and how related risks were mitigated.

CHAPTER FOUR: DESIGN AND DEVELOPMENT

4.1 Introduction

The objectives of this study include the application of machine learning classification techniques, such as Random Forest, to identify big data patterns related to cybersecurity networks and to develop a predictive model for these networks. To achieve these objectives, this chapter provides a comprehensive and detailed overview of the design process, including model training and testing techniques. It also outlines the tools and environmental setup used to facilitate this objective. Furthermore, this chapter elaborates on the execution of the methodology previously described in Chapter 3. Finally, it discusses various sections, including the development environment, data preparation and exploration, feature engineering and selection, predictive model implementation, initial model performance evaluation, and a summary of the developed model

4.2. The Malicious Big Data Classification Problem

The classification of malicious big data is effectively modeled as a binary classification problem, as it involves predicting observations that fall into one of two classes: malicious and benign (Uhr, 2023). Classification is a supervised machine learning technique that assigns data to discrete categories, thereby linking an unknown pattern to a known class (Alnuaimi & Albaldawi, 2024). Binary classification specifically separates data into two categories. In this study, big data is categorized as either benign or malicious. The aim is to determine, based on a big data dataset, whether the data is benign or malicious. Figure 4.1 illustrates the flow chart of the big data classification problem.

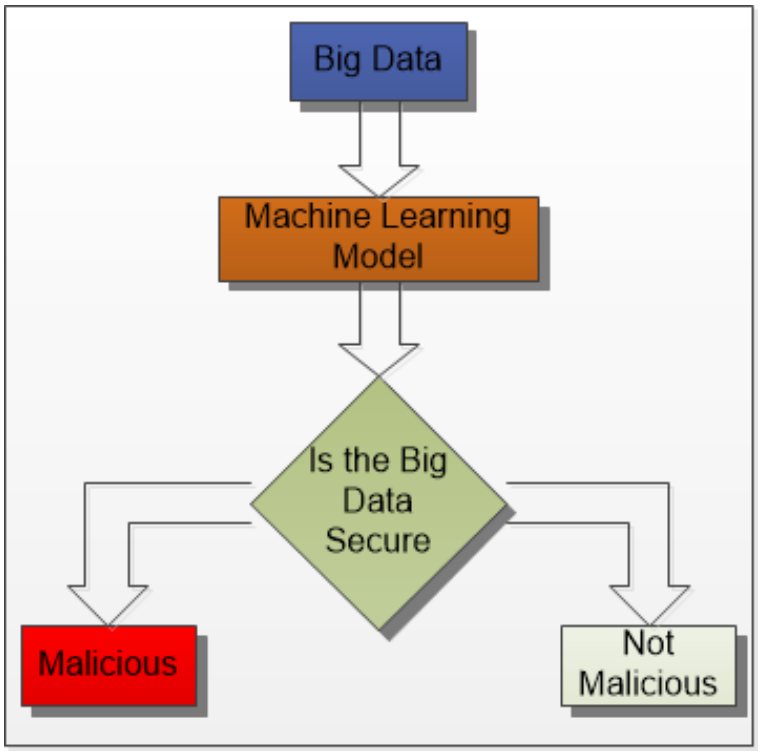


Figure 4.1: Flow chart of the malicious data classification problem

4.3 Model Design

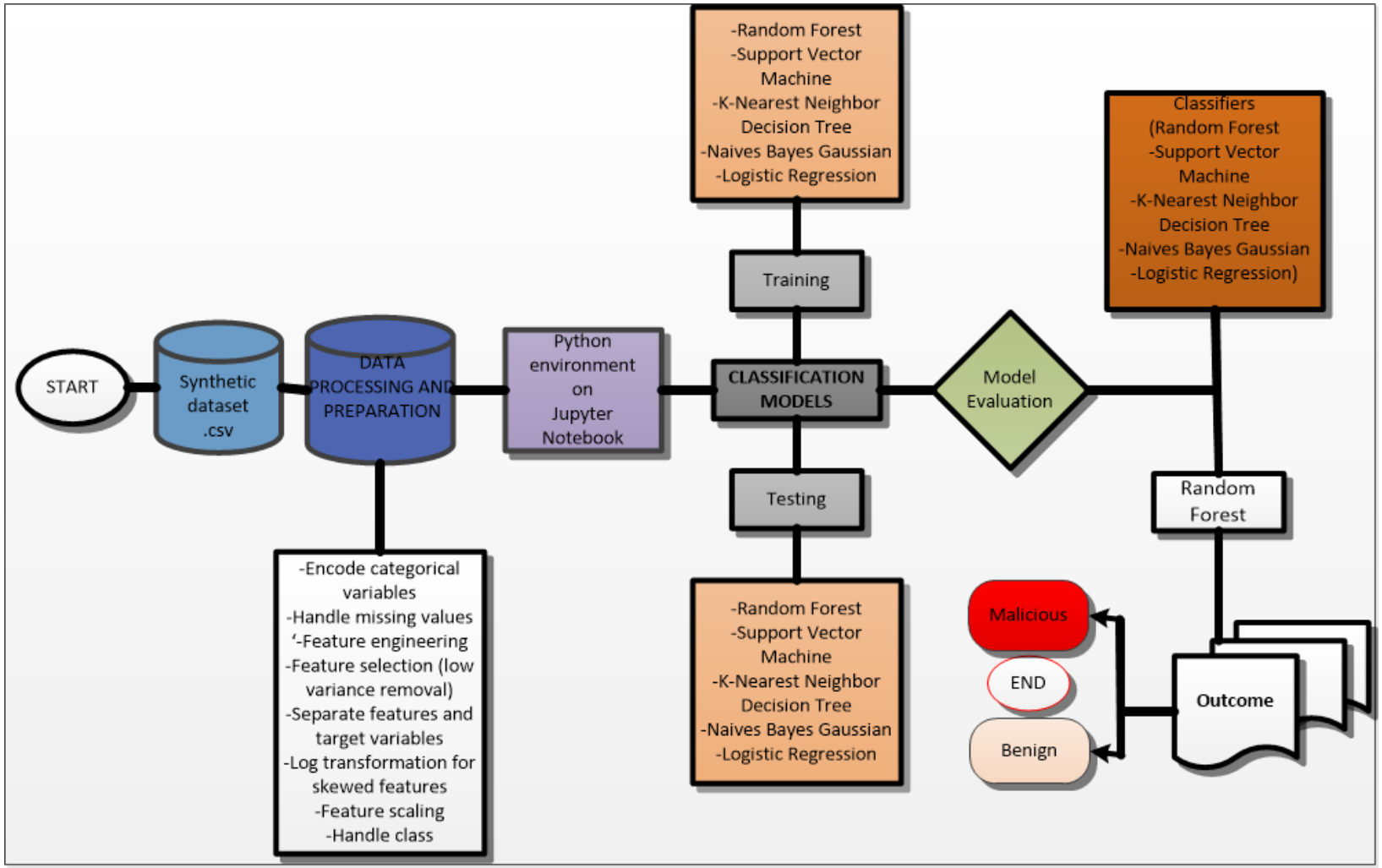


Figure 4.2: Proposed malicious detection model

Figure 4.2 shows the process followed to carry out the study experiment. The datasets were synthetically generated in CSV format, which is suitable and acceptable for the nature of the study conducted. The datasets were loaded in the experiment's Integrated Development Environment (IDE). For this research study, Jupyter Notebook was the preferred IDE for exploring the structure. This was followed by data preprocessing to ensure that the datasets were organized, sanitized, and contained minimal biases for model training and testing. Moreover, preprocessing was done to enhance model performance and accurate prediction. For model training and testing, eight machine learning classification algorithms were evaluated. These include: Random Forest Classifier, Gradient Boosting Classifier, Support Vector Machine Classifier (SVC), KNeighbors Classifier, Logistic Regression, Decision Tree Classifier, Gaussian Naive Bayes (GaussianNB), and Multi-layer Perceptron Classifier (MLPClassifier). Gradient Boosting Classifier and Random Forest Classifier were used for feature importance because they are both tree models, suitable for feature importance (Xu et al., 2019). Gradient Boosting Classifier and Random Forest Classifier were used for feature importance because they are both tree models suitable for feature selection, robustness (Xu et al., 2019). Feature importance is significant in this study because it determines which features have the most predictive power or influence for predicting whether a cyber threat is malicious or benign (Buyukkececi & Okur, 2023). Finally, the chosen algorithm was Random Forest because it provided the best evaluation metrics in terms of accuracy, precision, F1-score, and recall.

4.4 The classification algorithm used

4.4.1 Random Forest Classifier

The selected algorithm for this study was the Random Forest. The Random Forest is widely used in machine learning for classification and forecasting (Salman et al., 2024). It is also used in predictive data mining, where historical data is studied to derive patterns and trends that can be utilized for future predictions. Its advantages for this task can be attributed to its high accuracy, as it yields superior outcomes even without using hyperparameter tuning compared to other approaches. It overcomes the challenge of

overfitting by ensembling multiple decision trees, which reduces bias and volatility. Furthermore, Random Forest handles missing values and provides feature importance metrics that influence the model's prediction, removing noisy or irrelevant features and positively influencing dimensionality reduction (Thomas & Kaliraj, 2024). Figure 4.3 provides a demonstration of how Random Forest works in detecting malicious and benign data. The algorithm creates several decision trees, with each decision tree trained on a set of data to determine whether the data is malicious. The data is also used as either malicious or benign based on the combined decisions of all the trees. The data containing both malicious and benign characteristics is used to train the algorithm, with each record containing various features such as traffic_volume, packet_size, protocol, duration, login_attempts, cpu_usage, and is_malicious. The traffic_volume, packet_size, duration,login_attempts, and cpu_usage. These features were used to train the decision trees, with each decision tree responsible for splitting data to extract the most relevant features for accurate cyber threat detection. In the testing phase, input is given to the algorithm as a data record, and each decision tree predicts whether the data record is malicious or benign. A final decision is given by the majority vote, which is determined by the aggregated predictions of all the trees. If the final decision indicates the data record is malicious, the data may be blocked or flagged for further investigation by the cybersecurity department.

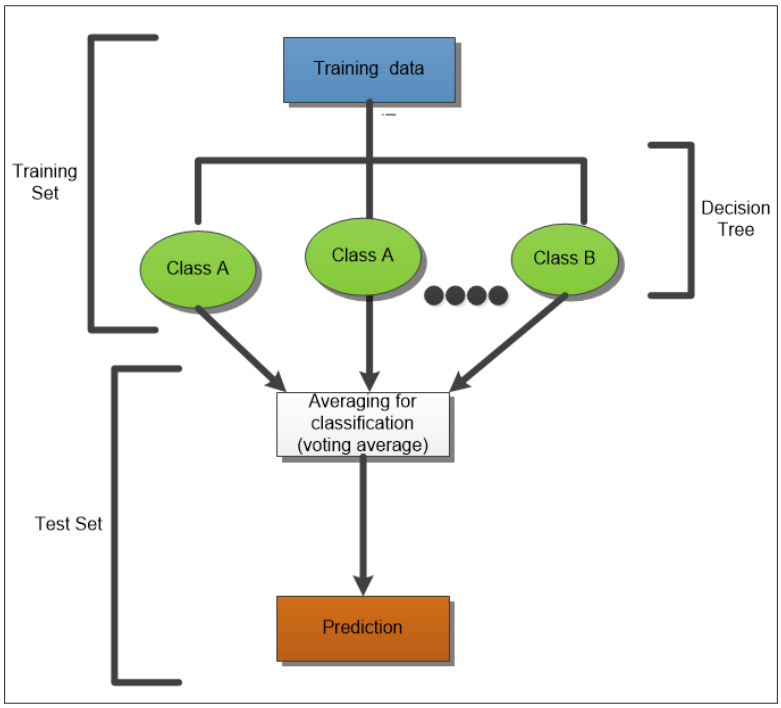


Figure 4.3: A diagram of how Random Forest works

4.5 Experiment Development Environment

4.5.1 Software

The software used in the study was Jupyter Notebook version 7.2.2, which supports the Python programming language. To implement the proposed model, importing machine learning libraries was essential. These libraries provide key tools needed for every stage of the machine learning process. Stages include preprocessing, feature selection and engineering, model training, model selection, hyperparameter tuning, model evaluation, and model deployment. The main libraries used were Pandas for data manipulation and analysis, NumPy for numerical computation, and OS for operating system-related functions, such as creating files and folders. Scikit-learn (sklearn) was used for machine learning, and imbalanced-learn (imblearn) for handling imbalanced datasets in machine learning, joblib for loading and saving the machine learning models, seaborn for data

visualization, matplotlib for plotting, and time for time intervals of the algorithms and models.

4.6 The Experimental Development Process

The process of developing the cyber threat prediction model was experimental in nature. The experiment began by generating synthetic datasets that consist of features mimicking the real-world data that flows through a network. The dataset included malicious and benign records created in CSV format. This was followed by preprocessing the data, which involved handling missing values, removing duplicates, transforming skewed distributions, performing correlation analysis to identify the features most strongly related to the target for feature selection, and removing features with low variance. The target variable was originally encoded categorically. The label encoding was implemented to indicate whether the data was malicious or benign. The protocol was also categorically encoded and assigned a label encoding that indicated its type. Consequently, the dataset was divided into subsets of training, validation, and testing sets. The Random Forest Classifier, Support Vector Classifier, KNeighbors Classifier, Logistic Regression, Decision Tree Classifier, Gaussian Naive Bayes, and Multi-layer Perceptron Classifier were trained using the training data. The performance of the models was evaluated using metrics such as accuracy, precision, F1-score, and recall determining the best-performing model. The model development process involved an iterative process of analyzing the correlation of features before and after feature engineering, testing the model with sample data, and training the model with different thresholds to ensure accuracy and improve overall model performance. Finally, the results from the evaluation metrics were analysed, interpreted, and presented. Figure 4.4 illustrates the experimental development process undertaken.

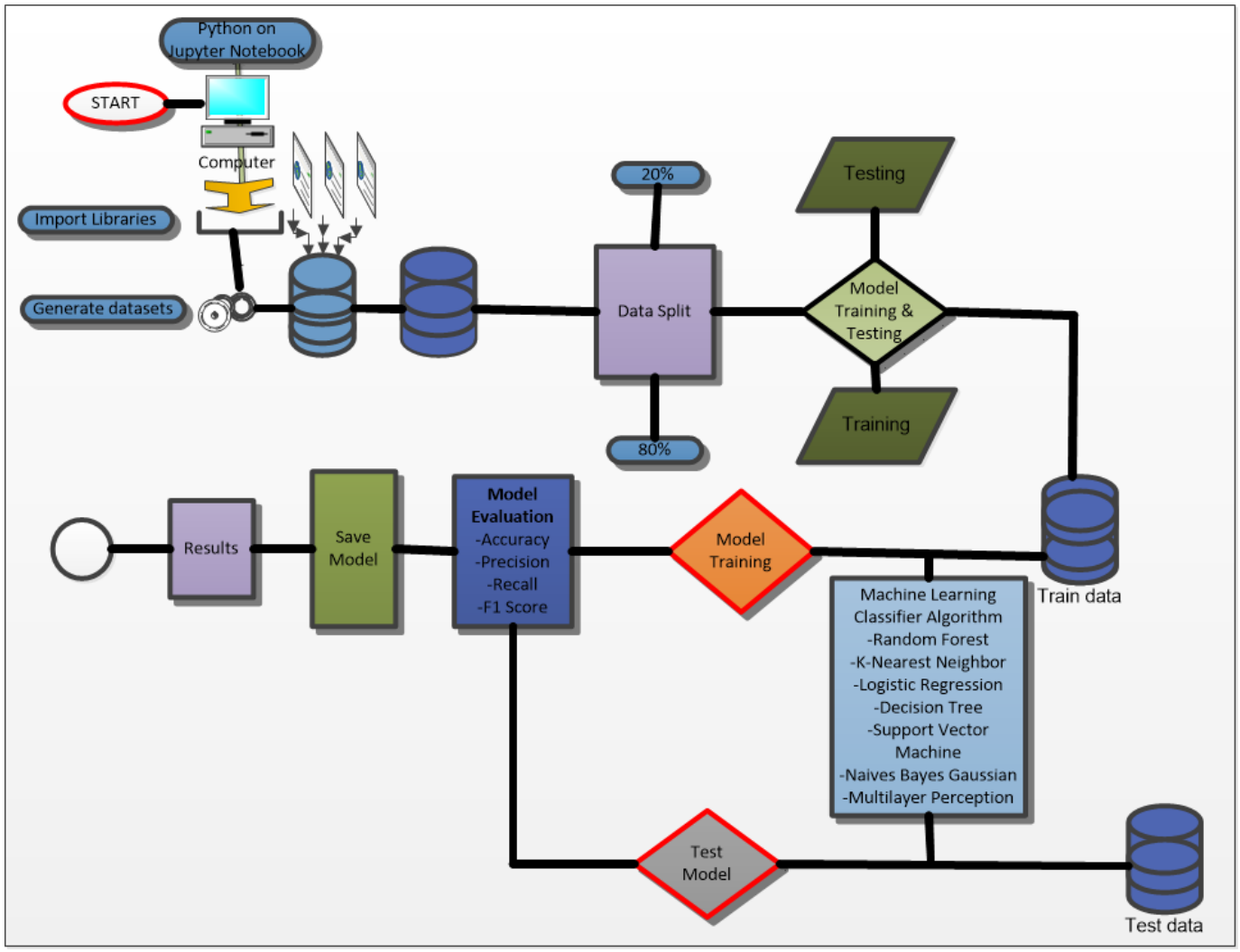


Figure 4.4: The experimental development process

4.7 Performance evaluation

Evaluation metrics were used to assess and review the quality of the trained model when tested with unseen data (Vujović, 2021). Furthermore, the evaluation metrics were used to assess the model's ability to generalize. In the current study, evaluation metrics were used to measure and summarize the quality of the trained cyber threat prediction model. The metrics used include confusion matrices, accuracy, precision, F1-score, and recall. These evaluation metrics were used to determine the best-performing model, with the outcome being either malicious or benign data.

4.7.1 Confusion matrices

The confusion matrices are generated in the train_evaluate_models function as part of the model evaluation process (Sathyanarayanan, 2024). The process started with each model being trained using training data. Once the training was complete, the model was able to make predictions on unseen test data using the method y_pred = model.predict(X_test). The confusion matrices were generated using the sklearn. metrics.confusion_matrix library, which contains the confusion_matrix () function. The formula used to achieve this was cm = confusion_matrix(y_test, y_pred), which compared the true labels y_test with the predicted labels in y_pred. The number of samples that belong to a specific class was counted. The four classes included are (Yang & Berdine, 2024):

- **True positive (TP):** Represents instances that were correctly predicted as a positive class.
- **False positive (FP):** Represents instances that were not accurately predicted as a positive class.
- **True negative (TN):** Represents instances that were accurately predicted as a negative class.
- **False negative (FN):** Represents instances that were inaccurately predicted as a negative class.

The values for true positive, false positive, true negative, and false negative were extracted and displayed in a table stored in CSV format. The confusion matrices were

visualized using seaborn. heatmap() function. The sns.heatmap(cm, annot=True, fmt= d', cmap='Blues', xticklabels=['NM', 'M'], yticklabels=['NM', 'M']) function was used to generate a graphical representation using the Seaborn library. This displays the labels in the x and y axes that indicate " malicious" and "benign," and the plot is saved to a file for display. The visualisations are further discussed and depicted in Section 5.5

4.7.2 Accuracy

Accuracy measures the number of times the model correctly predicted occurrences. It measures the model's prediction overall correctness (Rainio et al., 2024). The accuracy_score() function from Python's scikit-learn library was used in this study to determine the ratio between accurate predictions and the total predictions made. This metric assessed the accuracy of the classification models. The accuracy metric made use of the following equation to make a prediction (Owusu-Adjei et al., 2023):

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

In classification, accuracy is calculated in terms of positive and negative predictions. According to the formula, accuracy is equal to the correct predictions divided by the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Equation 4.1: Accuracy formula

Where TP = True Positives, TN= True Negatives, FP = False Positives, FN = False Negatives.

4.7.3 Precision

Precision was measured by the number of correct positive malicious data predictions made by the model (Zhou et al., 2021). In this study, precision was calculated as the ratio

of the true positive predictions to the total of true positives and false positives. The equation used to calculate the precision of the models was as follows:

$$Precision = \frac{Number\ of\ True\ Positives}{Sum\ of\ True\ Positives\ and\ False\ Positives}$$

According to the formula, precision is calculated as the number of true positives divided by the sum of true positives and false positives. In other words, precision is calculated as follows:

$$Precision = \frac{TP}{TP\ +\ FP}$$

Equation 4.2: Precision Formula

Where TP = True positives, and FP = False positives.

4.7.4 Recall

The recall metric was used to evaluate the number of positive instances in the data that the model accurately classified (Zhou et al., 2021). In this study, this metric was computed as the ratio of true positive predictions to the total number of actual instances. The equation used to calculate the precision of the models was as follows:

$$Recall = \frac{Number\ of\ True\ Positives}{Sum\ of\ True\ Positives\ and\ False\ Negatives}$$

According to the formula, recall is calculated as the number of true positives divided by the sum of true positives and false negatives. In other words, recall is calculated as follows:

$$Recall = \frac{TP}{TP\ +\ FN}$$

Equation 4.3: Recall formula

Where TP = True Positives, and FP = False Negatives.

4.7.5 F1- Score

The F1-score metric is used to measure the model's accuracy by integrating both the precision and recall as a single value (Rainio et al., 2024). Precision measures the number of positive predictions that are in class 1, and recall measures the number of instances in the positive samples that were correctly classified by the model. In the current study, this metric is computed using the harmonic mean, which assigns equal weights to both the precision and recall metrics. This means the harmonic mean is directly proportional to the harmonic mean. For example, if the F1-score has a low value, then both precision and recall will also have low values. The equation used to calculate the F1-score of the models was as follows:

$$F1 - Score = 2\,x\,\frac{Precision\ x\ Recall}{Precision\ +\ Recall}$$

Equation 4.4: F1-Score formula

4.8 Summary of Developed Model

This chapter covered the implementation process undertaken for the development of the model. It delved into the experimental process followed to overcome the classification problem of malicious data flowing in networks. The experimental process was described in detail, outlining the steps taken, from data collection, preprocessing, training, validation, and evaluation of the models. The experimental environment used was Jupyter Notebook, along with the Python programming language, to support the various functions and modules required for building the model. Furthermore, the hardware specifications used for the models were also outlined. Various classifier algorithms, such as Random Forest Classifier, Gradient Boosting Classifier, KNeighbors Classifier, Logistic Regression, Decision Tree Classifier, Gaussian Naive Bayes, and Multilayer Perceptron Classifier, were evaluated to determine the most appropriate algorithm to use to tackle the classification problem. Moreover, these algorithms were evaluated using evaluation metrics, including confusion metrics, accuracy, precision, F1-score, and recall.

CHAPTER FIVE: RESULTS AND DISCUSSION

5.1 Introduction

This chapter presents the study's results. As previously explained, the nature of the problem in this study is a classification problem. Thus, the study employed performance evaluation metrics commonly used for classification models in machine learning. The chapter, therefore, presents the results of the performance evaluation based on the metrics: accuracy, precision, recall, and F1-score. The chapter begins by explaining the experimental process that was undertaken and the results of each step. Additionally, the confusion matrix's prediction outcomes are explained. Additionally, the chapter answers the fourth objective of this study.

5.2 Results

Once the model was trained, tests were run to determine which model best classified malicious data. All the algorithms were trained and tested using equal amounts of data of the same size, which was divided into a training set and a testing set. The results from these models were compiled using the classification from the classification_report() function in scikit-learn. metrics, which consist of evaluation metrics.

5.3 Data analysis

5.3.1 Correlation Analysis

Exploratory data analysis (EDA) is a crucial step in the data analysis process. It involves investigating, evaluating, and summarizing the primary characteristics of datasets (Dhummad, 2025). Exploratory data analysis is one of the methods used to identify data anomalies, discover important relationships among dataset features, and draw insightful patterns (Assegie et al., 2021). It simplifies the data preparation process for deeper analysis. One approach to performing EDA is Pearson's correlation, a tool widely used to determine the strength and direction of a linear association with values ranging from 1, indicating a perfect negative linear correlation, to +1, indicating a perfect positive linear correlation (Das, 2023). The Pearson correlation was used in this study to analyze and

visualize the relationship between each feature and the target variable in the dataset. Furthermore, Pearson's correlation is also used for feature selection and model validation. This was crucial for determining the features that are most relevant for classifying between the two different classes in the dataset. The next section shows the results of the correlation analysis.

5.3.1.1 Correlation analysis of original features

Figure 5.1 demonstrates the correlation coefficient of the original features to determine if a linear relationship exists with the target feature(is_malicious). The top features with a positive correlation ranked according to their importance include login_attempts, protocol, duration, and cpu_usage. The other two features, traffic_volume and packet_size, were found to have a negative correlation. Overall, the correlation analysis shows that there is no strong linear relationship between the original features and the possibility of an event being malicious, as all the features are very close to zero. This suggests that linear models may struggle to identify insightful patterns. This is a limitation of Pearson's correlation, the inability to measure complex or nonlinear relationships that are common in real-world data and scenarios (Das, 2023). Thus, feature engineering was applied by creating new features to determine whether a stronger relationship exists between the features and the target feature rather than relying on the original features.

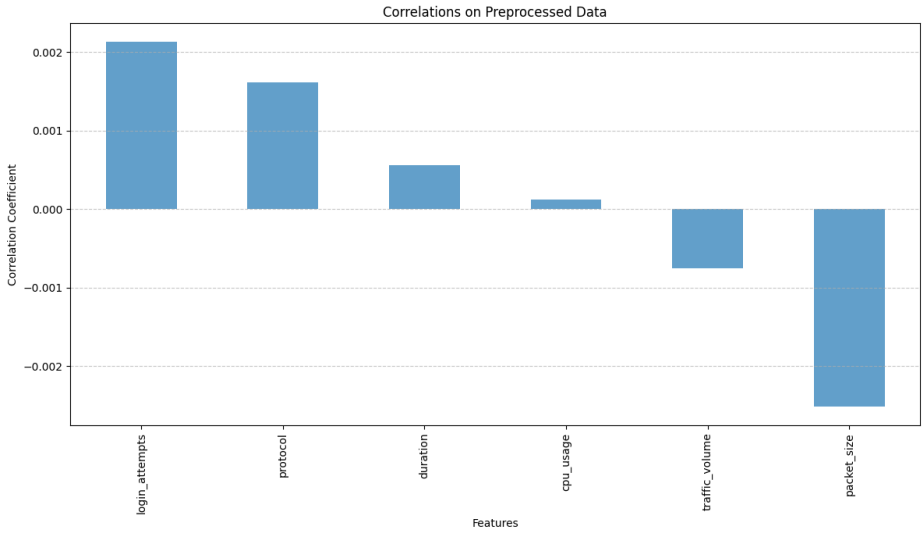


Figure 5.1: Correlation coefficient of the original features

5.3.1.2 Correlation analysis of Feature Engineered features

Figure 5.2 demonstrates the correlation coefficient of the feature-engineered features to determine if a linear relationship exists with the target feature(is_malicious). The top features with a positive correlation, ranked according to their importance, include: login_duration_interaction, login_attempts, cpu_traffic_ratio, protocol, duration, cpu_usage, and cpu_traffic_interaction. The other two features, traffic_volume, packet_size, and login_duration_ratio, were found to have a negative correlation. Overall, the correlation analysis reveals that there is no strong linear relationship between the engineered features and the likelihood of an event being malicious, although some engineered features exhibit a high correlation score. The number of features still remained very close to zero. Hence, the final selection of features with high correlation scores is to be used by the model.

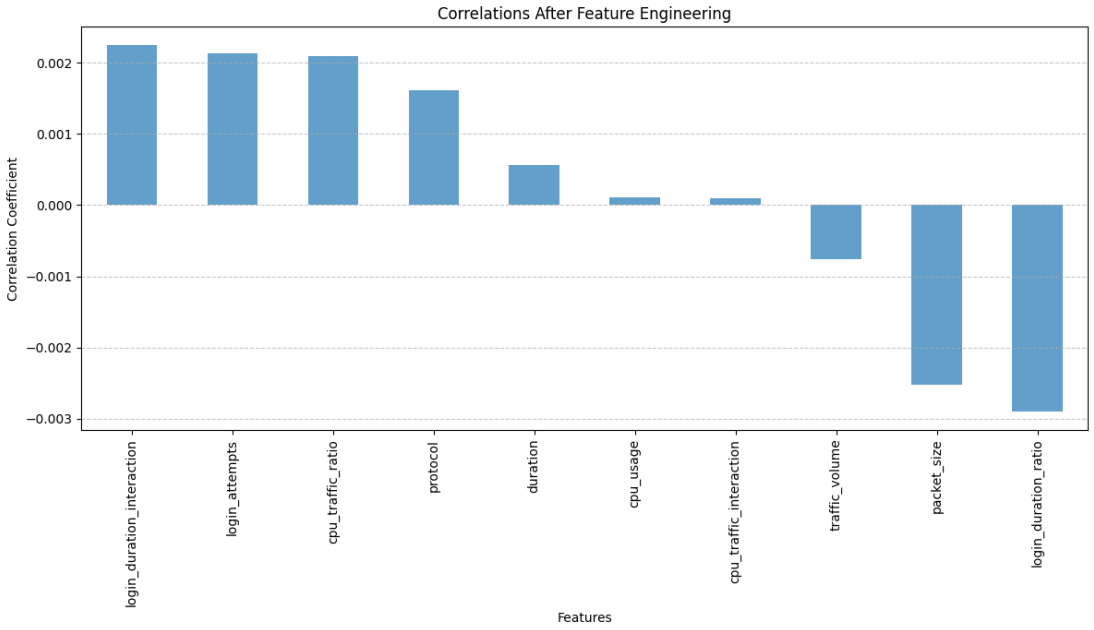


Figure 5.2: Correlation coefficient after feature engineering

5.3.1.3 Data preprocessing

Since original features and feature-engineered features do not have acceptable correlation scores, low-variance feature selection, transforming skewed features, and

separating features were done to improve the quality of data, stability, and performance of the machine learning models.

a) Low Variance Feature Selection

b) Feature selection is a method for selecting the most relevant dataset and discarding redundant or irrelevant datasets (Cheng, 2024). Low-varience feature selection is a method of discarding irrelevant dataset features with low variance from the dataset. Low-varience features were discarded from the dataset as they do not provide useful information to the models and help reduce overfitting and simplify the model (Tariq, 2024). The variance threshold hold was set to 0.01 to remove features that do not vary much and are constant, zero-variance features (Knutmejer et al., 2023). The two low-variance features removed included cpu_traffic_ratio and login_duration_ratio.

c) Transforming Skewed Features and separating features/target

Low-variance removal was followed by transforming skewed features to achieve a more symmetric distribution. The login_duration_interaction feature was found to be skewed with a value of 1.0 and was transformed using log(x + 1) to make the distribution more symmetric. Furthermore, feature and target separation was done as it is important in machine learning for ensuring optimal model performance by adhering to best practices that ensure correct training, preventing overfitting, and data leakage (Eliane Birba, 2020)

5.3.1.4 Correlation analysis of Feature Engineered features

Figure 5.3 demonstrates the correlation coefficient of the selected numeric features to determine if a linear relationship exists with the target feature(is_malicious). The top features with a positive correlation ranked according to their importance include login_duration_interaction, login_attempts, protocol, duration, cpu_usage, and cpu_traffic_interaction. Essentially, the login_duration_interaction feature showed the highest positive correlation after the implementation of the skew transformation. The other features include traffic_volume and packet_size, which were found to have a

negative correlation. Overall, the correlation analysis shows that there is no strong linear relationship between the selected numeric features and the possibility of an event being malicious, although some selected numeric features exhibit a high correlation score. The number of features still remained very close to zero. Hence, the implementation of advanced techniques is required to find nonlinear relationships between features.

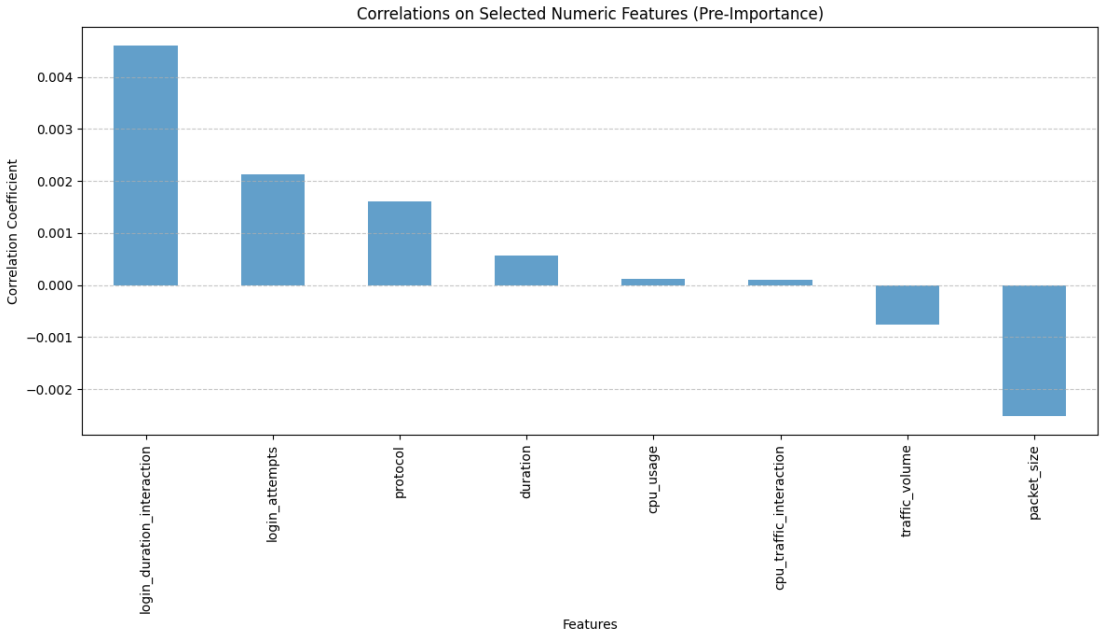


Figure 5.3: Correlation coefficient on selected features

5.3.2 Mutual information importance

Mutual information (MI) is used in machine learning to measure the information content of each feature in relation to the output class (Battiti,1994). It is used for nonlinear modelling to capture the most relevant features of a target variable in complex classification tasks. The most informative features include the protocol, which provides the most information on whether an event is malicious. In the field of cybersecurity, protocols are commonly used in the network security domain to prevent malicious programs from penetrating the network (Asaad & Saeed, 2022). Thus, it would make sense that the protocol is associated with malicious activities, as attack mechanisms are applied to weaken or bypass protocols to perform malicious activities. The protocol mutual information high score was followed by login_attempts and a moderate mutual information score, such as login_duration_interaction, cpu_usage, and traffic volume. Finally, the

least informative mutual importance features include packet_size, duration, and cpu_traffic_interaction. These features' scores were close to zero, an indication that they provide limited information and predictive power. Hence, implementing machine learning models to identify predictive relationships remains the final option.

5.3.3 Feature importance

When building machine learning models, it is crucial to rank features based on their contribution to the model output. This is especially important for safety-critical applications, where understanding how and why a model makes a decision is crucial, and for high-dimensional datasets, where challenges such as the curse of dimensionality, feature interaction, redundancy, and irrelevance are prevalent (Cheng, 2024). Moreover, understanding how a model makes decisions in these applications is crucial for demonstrating adherence to ethical concerns and regulatory compliance, verifying model performance, conducting model diagnostics, building enhanced models, and establishing trust among users (Rengasamy et al., 2022). Thus, feature importance was applied to determine the most relevant features from the dataset. Below are the results of the feature importance from the Random Forest Model and the Gradient Boosting.

5.3.3.1 Feature importance with Random Forest

Feature importance was extracted using the Random Forest model classifier, which performs feature selection using a set of variables with strong predictive power that lead to optimal performance in high-dimensional data (Menze et al., 2009). Figure 5.4 of the Random Forest Model provides a different importance score compared to the correlation analysis and mutual information importance. The model identified five features with the highest importance, with a score of around 0,15, which included packet_size, cpu_usage, duration, traffic_volume, and cpu_traffic_interaction from the feature engineering. This means the model utilized these features in its decision-making. The features protocol and login_attempts show the least importance score compared to when they were ranked by the Mutual information importance. This suggests that the

information in these features may be redundant, or that the Random Forest Model's most relevant features capture the same information.

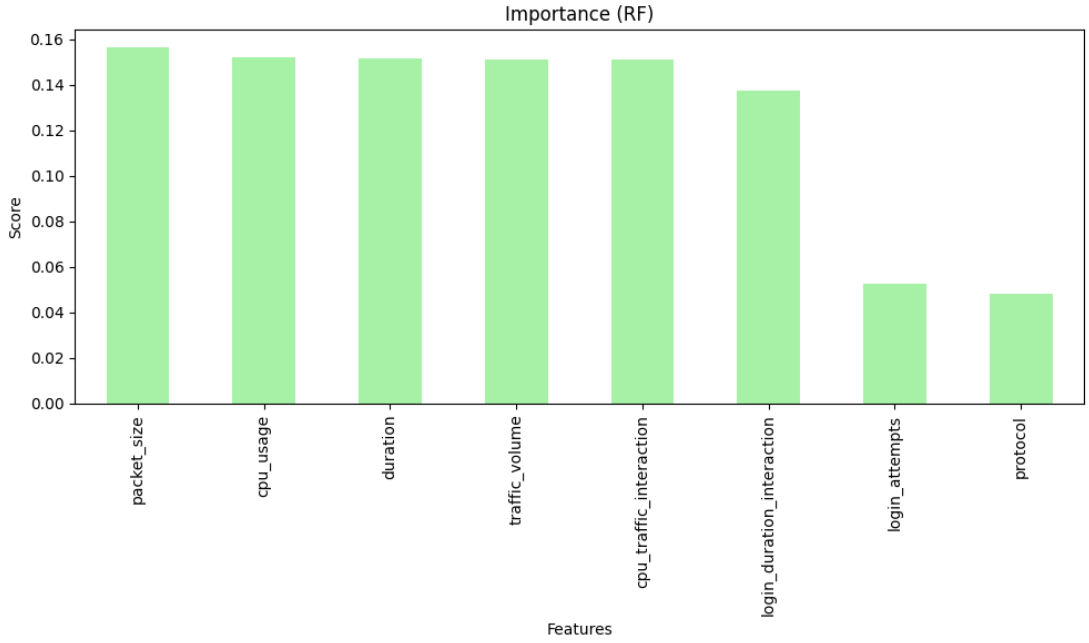


Figure 5.4: Feature importance with Random Forest Model

5.3.3.2 Feature importance with Gradient Boosting

Gradient Boosting is a widely used machine learning technique for tabular data, learning tasks, and ranking (Adler & Painsky, 2021). Furthermore, it handles categorical values, supports missing values, and constant feature scaling. Figure 5.5 of the Gradient Boosting Model ranked cpu_usage as the only and highest predictor, followed by cpu_traffic_interaction. Additionally, packet size, traffic volume, and duration were ranked as highly important in the Gradient Boosting Model, in the same way as the Random Forest Model. Similarly, the protocol and login attempts were ranked as having the least importance in this model, similar to the Random Forest model. This indicates that predictive information is captured effectively with tree-based ensemble models.

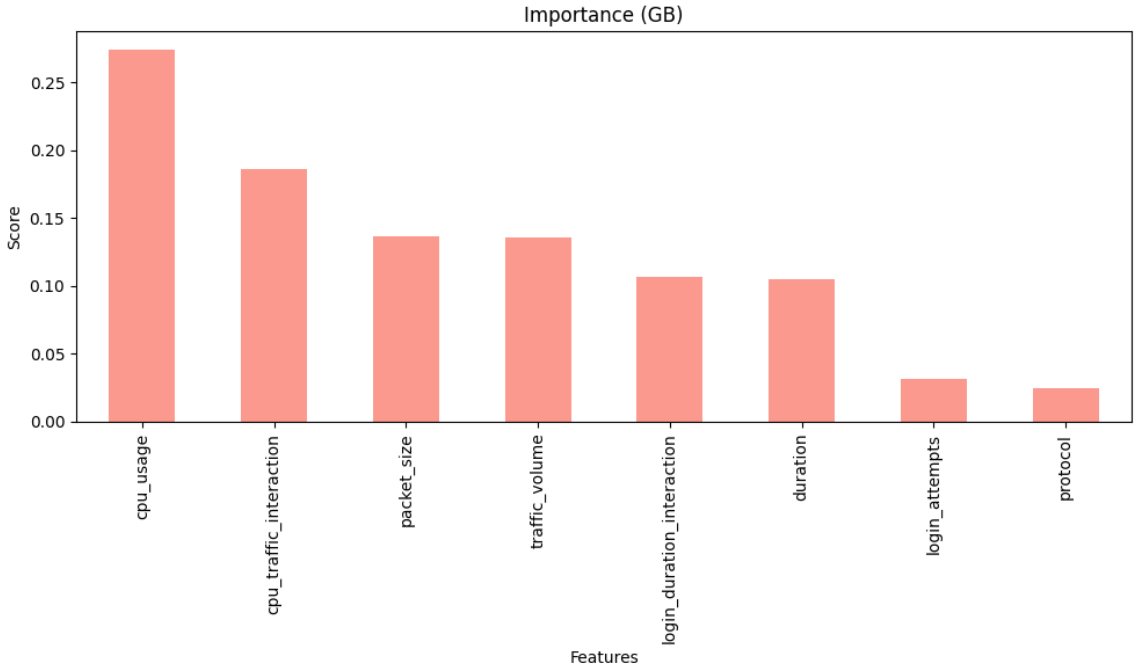


Figure 5.5: Feature importance with Gradient Boosting Model

5.3.3.3 Final feature selection

A final list of features was selected for training the final model. The final list of features was selected based on the importance analyses, and the selection criteria were based on an importance score that is greater than 0.01 from either the Random Forest or Gradient Boosting models. This selection criterion ensured that features with highly predictive power from either the Random Forest model or the Gradient Boosting model are retained. The final features with high predictive power selected for model training were:cpu_traffic_interaction,cpu_usage,duration,login_attempts,login_duration_interacti on, packet_size, protocol, and traffic_volume.

5.4. Classification Report on the final model

The classification report presents the results of each model's performance in distinguishing between malicious and benign data. The results include evaluation metrics, such as precision, recall, and F1-score, as well as support for each class, distinguishing between malicious data and benign data. Precision, also known as Positive predictive

value (PPV), refers to the proportion of the model's positively classified instances that are actually positive (Varoquaux & Colliot, 2023). In the case of binary classification of malicious and benign data, precision measures the proportion of data that was classified as malicious. Precision is influenced by false positives; as false positives decrease in the model, precision improves, and as they increase, the precision of the model decreases. A high-precision score is a good indicator that a model can effectively reduce false positives, thus ensuring that the majority of positive predictions are reliable. As such, precision is often prioritized as the model performance metric in fields such as fraud detection, where positive predictions need to be accurate and the costs of false positives are high (Schlosser et al., 2024).

Recall, also known as sensitivity or rate of true positives (TPR), is a metric that determines the fraction of all instances that are actually positive and correctly classified as such (Foody, 2023). In the case of the classification problem in this study, recall measures the portion of data that was correctly classified as malicious. Recall is a crucial metric in cases where correctly identifying a positive instance is key, meaning that a model unable to correctly identify all positive instances has negative implications (Schlosser et al.,2024). F1-score is the harmonic mean of recall and precision; it combines both metrics, precision and recall. According to Marwah et al (2024), it is a key metric for assessing classification models and cases where datasets are highly imbalanced. However, its drawback lies in treating both false positives and false negatives equally, where the implications for both depend on the application context. Support is a standard component in a classification report in the Python scikit-learn library, representing the actual number of instances in each class within a test dataset. This study represents the number of malicious and benign instances in the dataset.

5.4.1 Random Forest evaluation results

Table 4 depicts the classification report for the Random Forest Model. The precision exhibited by the model identified malicious instances correctly 98% of the time. This indicated that the model made limited false positives, eliminating the need for the cybersecurity department to spend time investigating benign cases that were incorrectly

flagged as cyber threats. Furthermore, the precision also exhibited that the model identified benign instances correctly 96% of the time. The results showed that the model correctly identified 96% of all malicious instances that were actually cyber threats. In the recall, the model further correctly identified 98% of benign instances that are not actually malicious. Additionally, the F1-score, which provides the balanced average of precision and recall, indicates that both the malicious and benign classes have a score of 0.97, showing that the model balances the classes well. Moreover, the support exhibited 9504 malicious instances and 9504 benign instances; they are equal in size. This was an indication that the test class is well-balanced. Finally, the Random Forest model's accuracy was 97%, indicating that it correctly predicted the outcomes 97% of the time from all the predictions made using the test data.

Table 4: Classification Report for Random Forest Model

	Precision	Recall	F1-Score	Support
Not malicious (0)	0.96	0.98	0.97	9504
Malicious (1)	0.98	0.96	0.97	9504
Accuracy			0.97	19008
Macro avg	0.97	0.97	0.97	19008
Weighted avg	0.97	0.97	0.97	19008

5.5 Confusion Matrices of the Models

A confusion matrix, also known as an error matrix, is a summarized table used to explain the performance of a classification model, such as binary classification, with a focus on the number of correct and incorrect predictions made by the classifier (Varoquaux & Colliot, 2023). It is an N x N matrix, where N denotes the number of classes. In the case of this binary classification study, N is 2 X2, which shows the number of :

- **True positive (TP):** The model accurately predicts “not malicious data”.

- **False positive (FP):** The model accurately predicts " not malicious data" while it is actually "malicious data" (Type I mistake).
- **True negative (TN):** The model accurately predicts "malicious data".
- **False negative (FN):** The model predicts "malicious data" when it is, in fact, "not malicious data" (Type II mistake).

5.5.1 Random Forest confusion matrix

Figure 5.6 below shows the confusion matrix of a Random Forest

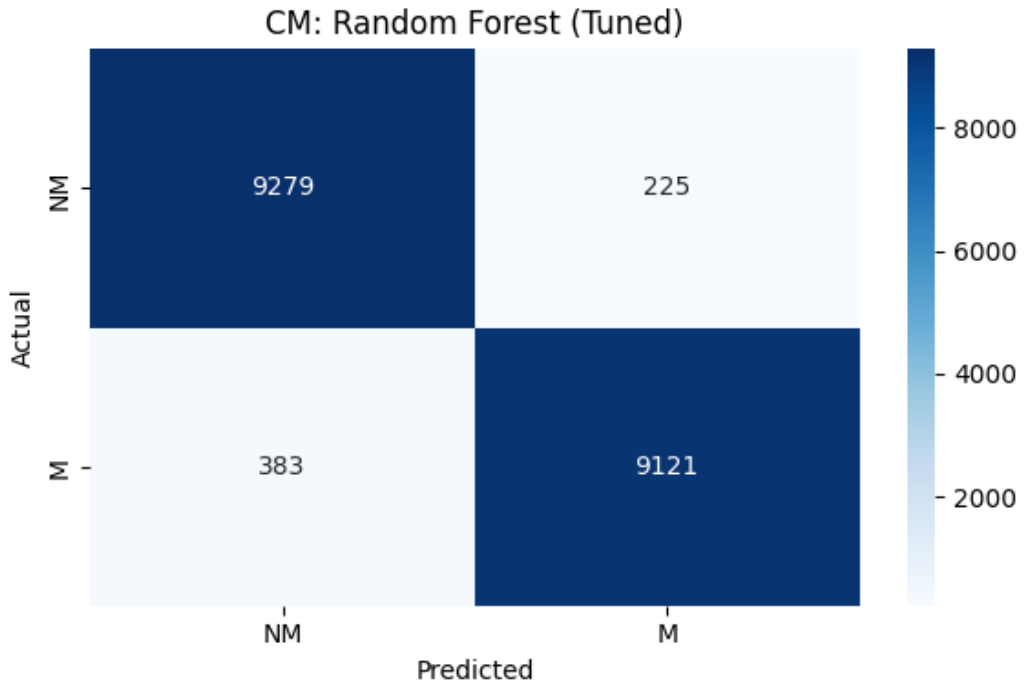


Figure 5.6: Confusion matrix of Random Forest Model

The confusion matrix above is for the Random Forest model. On the top left is the true negatives (TN), which indicates that the model correctly identified 9279 benign instances. On the top right are the false positives (FP), which show that 225 benign instances were misclassified as malicious or threats. The bottom left of the matrix depicts the false negatives (FN), the model identified 383 malicious instances as non-malicious, missing the threats. At the bottom right of the matrix, the true positives (TP)are listed, indicating that the model correctly identified 9121 instances as malicious data or threats. This

resulted in the Random Forest having an accuracy of 97%. The accuracy score is an indicator that the model is reliable in predicting malicious data or threats, achieving its primary purpose of detecting more threats while minimizing the number of false positives and missed threats. This is especially crucial for applications where false alarms, such as misclassification of real threats as benign, can have disastrous consequences.

5.6 Comparison of the Models

Figure 5.7 compares the performance of the models

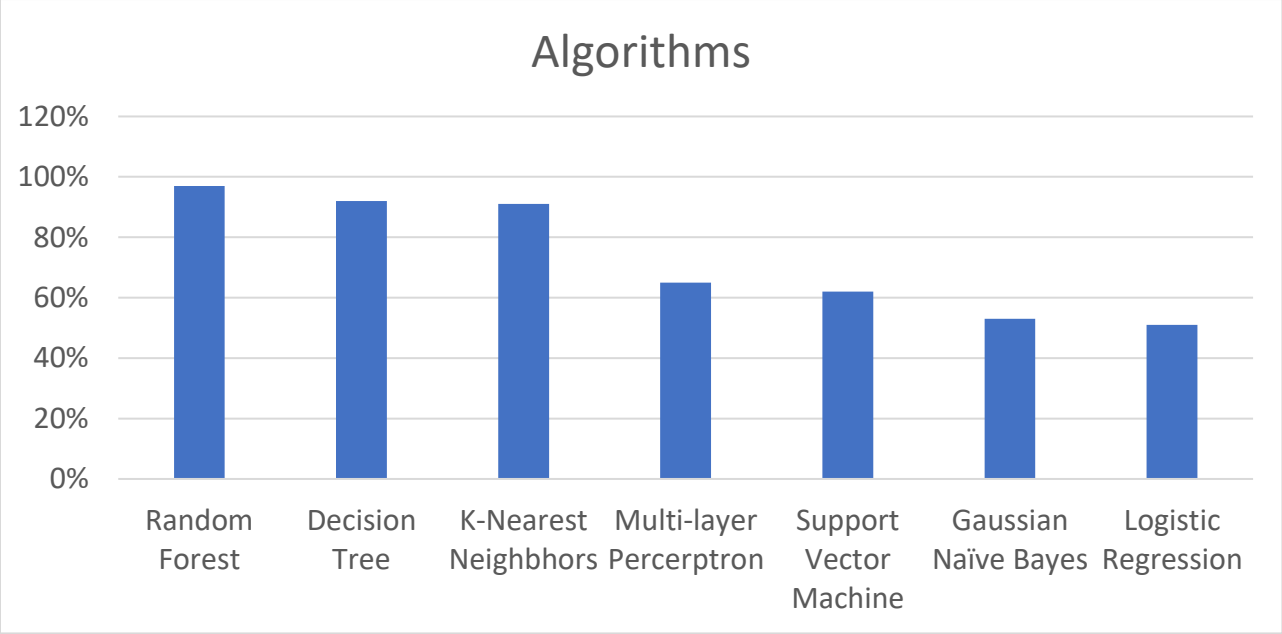


Figure 5.7: Bar graph showing the accuracy of the models

The bar graph above shows the accuracy across all seven classifier algorithms, which include: Random Forest, Decision Tree, K-Nearest Neighbors, Multi-layer Perceptron, Support Vector Machine, Gaussian Naive Bayes, and Logistic Regression. The first algorithm with the highest accuracy is Random Forest with 97% followed by Decision Tree with 92% and then K-Nearest Neighbors with 91%. The lowest accuracy is achieved by Logistic Regression

5.7 Discussion of the results in comparison to similar studies

Random Forest models are well-suited for handling large datasets and offer high accuracy compared to other models (Salman et al.,2024). Thus, this study used Random Forest to build a cyber threat detection model. Oyetoro et al. (2023) developed a Random Forest Model achieving 100% accuracy, notably higher than in several studies. They further highlight that higher accuracy in their study may be attributed to dataset and features used to train the model. According to Elhanashi et al., (2023), the IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018) dataset is known to have class imbalance (Elhanashi et al., 2023). An imbalanced dataset presents challenges of bias, feature vector size, overlap, dataset size, and performance bias (Altalhan et al., 2023). The study by Altalhan et al. (2023) recommends additional testing on new datasets to evaluate the model's effectiveness. Furthermore, a similar study was conducted by Hadi and Hadi (2024), which classifies network data using Random Forest with the NSL-KDD reference dataset. In this study, an accuracy of 99.34% was achieved. The higher accuracy may be attributed to the dataset used, which is known to have issues and is outdated. Most attacks represented in these older datasets have been addressed through patches, and updated software versions have been hardened to enhance security (Leevy & Khoshgoftaar, 2020). The slight differences in the models are due to the fact that the one chosen for this study uses synthetically generated data that represent current threats, whereas the other studies use older datasets that contain outdated threats. Moreover, the related studies do not incorporate feature importance into their machine learning preprocessing pipelines for identifying relevant features and feature selection. Feature selection helps remove redundant features, prevents overfitting, and enhances the model’s generalization (Cheng, 2024). Other factors that contributed to the differences in the model’s performance accuracy compared to other studies include the dataset's features, data preprocessing, such as feature engineering, and model hyperparameter tuning. The similarities in the performance of the models were their ability to capture data patterns. Finally, for future improvements in the model's performance, it is crucial to consider the factors mentioned in the model development process.

5.8 Model Interface

The model interface was developed using Streamlit, as highlighted in section 3.3. The model uses two modes, live simulation and batch upload. Live simulation is used for loading a single row of data to predict a single event, whereas batch upload is used to load a dataset in CSV format to make multiple predictions. Below are the interfaces for the two modes.

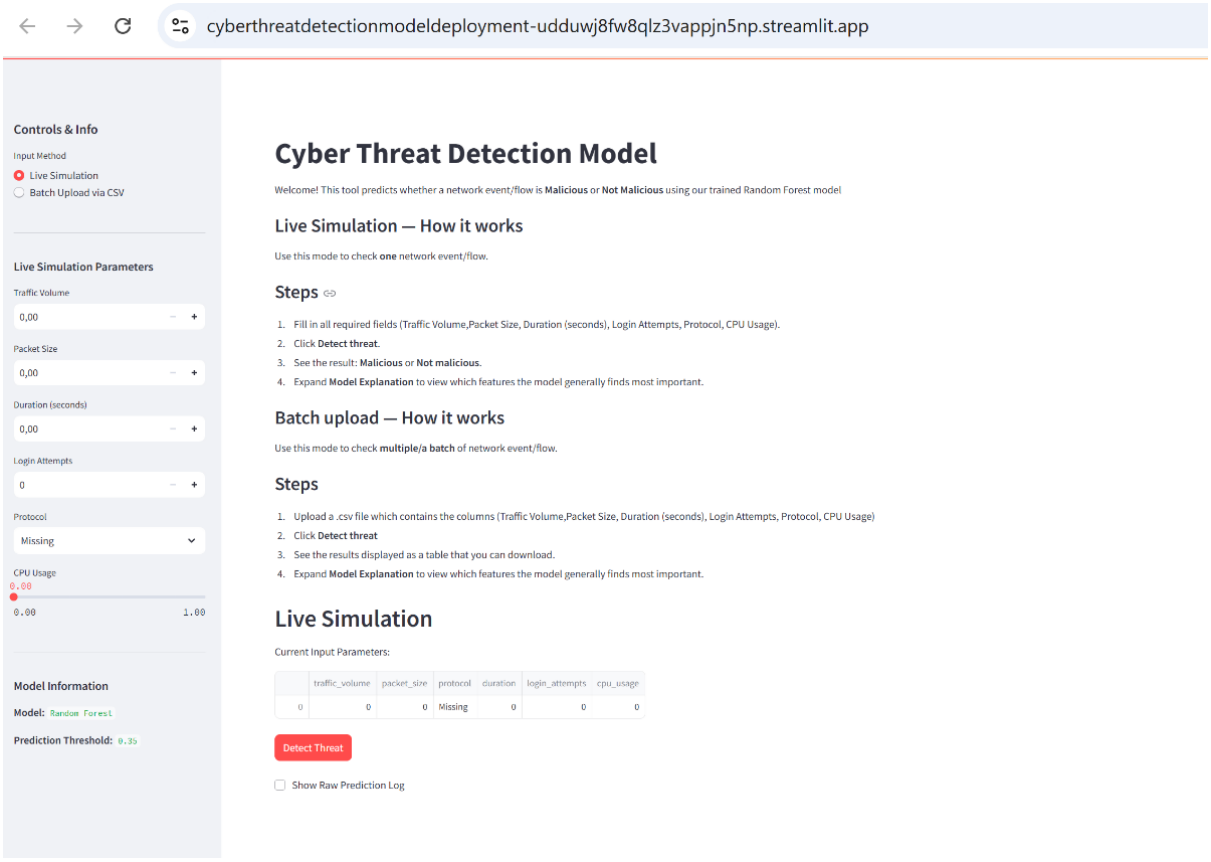


Figure 5.8: Interface for single prediction

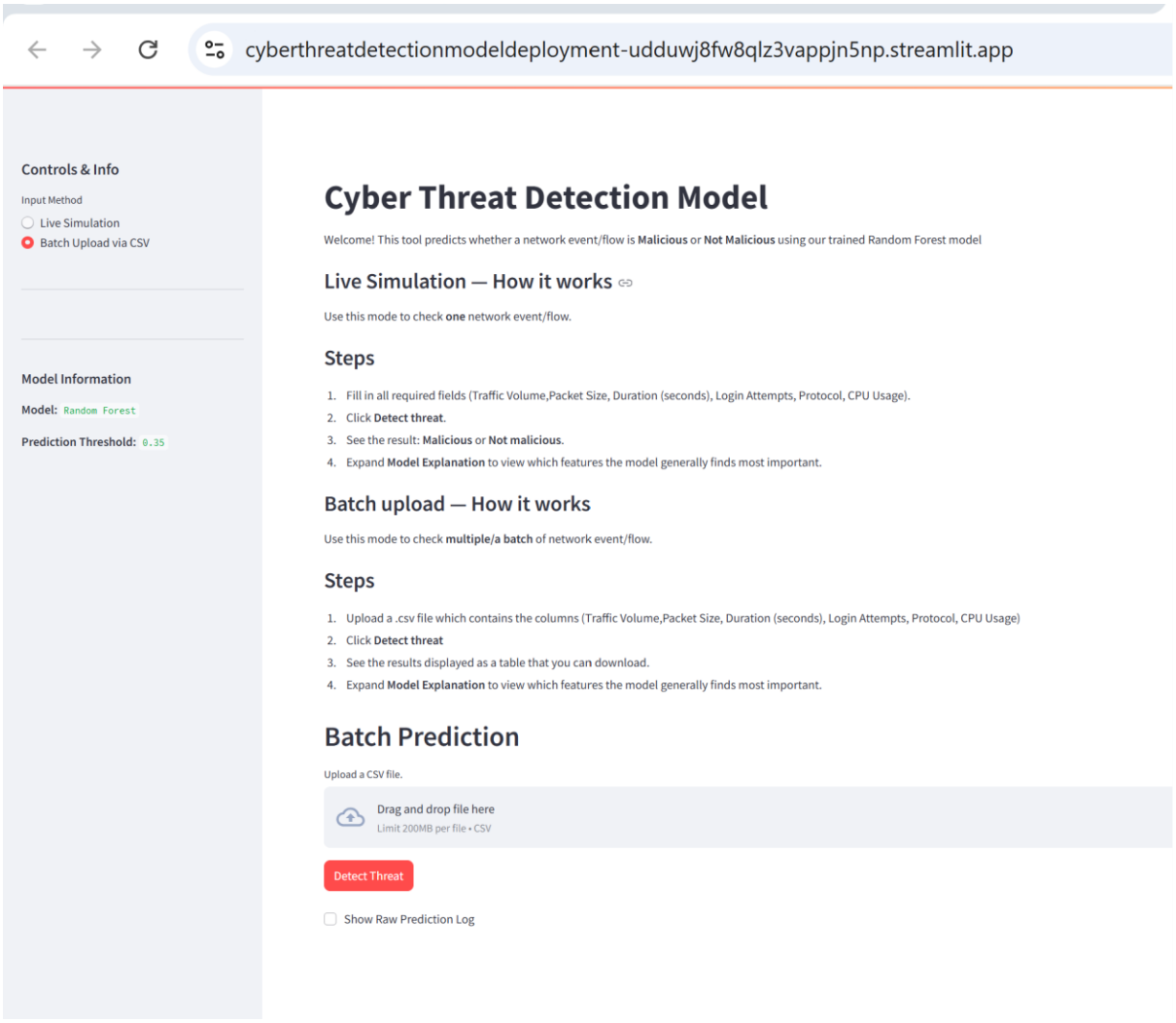


Figure 5.9: Interface for multiple predictions

5.9 Summary

The model developed demonstrated good performance in detecting malicious data with high accuracy, precision, recall, and F1-score. It performs well compared to other studies, particularly in generalizing unseen data. It is noted that there are some areas for improvement in the model's performance, such as considering the dataset's characteristics and size. Finally, the model has demonstrated potential for detection in simulated environments, highlighting the need for further validation in real-world scenarios. The next chapter explains the overall research done and provides recommendations for future work.

CHAPTER SIX: CONCLUSION AND FUTURE WORK

6.1 Introduction

This chapter presents the conclusion of this study, providing a reflection of the study from the research problem to the research findings. The chapter also explains how each research objective of the study was achieved. Furthermore, the limitations of the study are also presented in this chapter. Finally, the chapter highlights future work that can be conducted from this study.

6.2 The summary of the study

The objective of this study was to develop a big data analysis model for futuristic insights into cybersecurity networks. Various machine learning algorithms known to perform well in classification problems were used to determine whether the data was malicious or benign. These different algorithms were evaluated, and the best-performing one was based on the pattern of the selected dataset. The evaluated classifier algorithms included: Random Forest, Support Vector Machine, K-Nearest Neighbors, Decision Trees, Gaussian Naïve Bayes, and Multi-layer Perceptron. The Random Forest, which is an ensemble learning method based on decision trees, was selected as the final algorithm for building the model. Literature indicates that decision trees are widely used for classification problems, as they recognize data effectively. Their simplicity and interpretability make them more favorable than other machine learning algorithms. The simplicity and interpretability are key for users, as they enable them to easily comprehend and understand the decision-making process. Additionally, simplicity and interpretability are crucial in domains that value transparency and explainability, as these ensure improved and sustained stakeholder trust and the validity of results (Mienye & Jere, 2024). Moreover, decision trees are also used for feature selection, aiding in finding the most relevant features for differentiating between malicious and benign data. Furthermore, in the feature selection, a set of features is selected at each split to reduce overfitting and improve generalization (Srihith et al., 2023). Finally, to ensure that all the algorithms perform to their full potential, hyperparameter tuning was effected. These

parameters were applied to improve the accuracy, precision, recall, and F1-score of the model. Furthermore, the threshold value was set to a reasonable value to maintain a balance in the model's confidence in identifying false positives and false negatives. The threshold value is set to capture more real threats at the expense of generating more false positives that require further investigation. The study's experiment was conducted on a Dell Latitude 5520 laptop. The processor is an 11th Gen Intel (R) Core TM I7-1165 @ 2.80GHz 1.69 GHz, and it was running on a 64-bit operating system, x64-based processor. Python programming language was used in a Jupyter Notebook, an Integrated Development Environment that includes libraries supporting the machine learning development process. Furthermore, to make the model easily explainable and interpretable, Streamlit was used to build the interface.

First objective: To determine the factors that influence big data analysis in cybersecurity networks. This objective was achieved through Chapter 2 and the study's experiment, where feature importance was determined, providing a list of the most important and least important features. The features were determined using Random Forest and Gradient Boosting, as they are widely used for the identification of complex, important features. The top feature included : login_attempts, protocol, duration, and cpu_usage, login_duration_interaction, login_attempts, cpu_traffic_ratio, and cpu_traffic_interaction. The least important features included traffic_volume, packet_size, and login_duration_ratio. Thus, the top features were selected for training and testing all the algorithms.

Second objective: To establish methods and techniques that have been used to analyze security threats in cybersecurity networks. This objective was addressed in Chapter 2 of the literature review. Literature revealed that traditional reactive approaches were used, including log analysis, network traffic analysis, and intrusion detection systems such as anomaly-based detection, signature-based detection, and stateful protocol analysis. Machine learning model approaches include Random Forest, Neural Networks, and Logistic Regression. Literature showed that Random Forest is one of the preferred algorithms for classification problems due to its shorter training times and ability to handle

large datasets. Another preferred algorithm is logistic regression, especially for data categorization with binary class labels.

Third objective: To use machine learning classification techniques to identify big data patterns in cybersecurity networks. This objective was achieved through the study's experiment, where various algorithms, including Random Forest, Support Vector Machine, K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, and Multi-layer Perceptron, were trained and tested. Various performance evaluation metrics, such as precision, recall, F1-score, and accuracy, were used to determine the best-performing algorithm to build the final model.

Fourth objective: To use the identified patterns to develop a predictive model for cybersecurity networks. From the third objective, various machine learning classification techniques were trained and tested to identify big data patterns in cybersecurity networks. The final algorithm was selected to build the final model based on the performance evaluation metrics mentioned above. Random Forest was identified as the best-performing algorithm to build the final model. Overall, it had the highest accuracy. Precision, recall, and F1-score. Thus, it made it an excellent model for detecting malicious and benign data.

6.3 Future Work

The study presents opportunities for exploring malicious and benign data. In the future, the study can expand the dataset to include the type of attack. This will ensure researchers are updated and informed of current and evolving cyber threats. Additionally, future enhancement of this study could include the type of attack if the data is identified as malicious. Future work can explore other algorithms, such as deep learning algorithms and stacked algorithms, to improve performance. The study makes a scientific and practical contribution to the field of cybersecurity. Scientifically, the study contributes to understanding feature importance and selection by identifying the features with high predictive power for malicious data identification. In practice, the study highlights how these results can contribute to cybersecurity policy development and the implementation

of proactive defense strategies in cybersecurity networks handling big data for business operations.

REFERENCES

Aaser, M., & McElhaney, D. (2021). *Harnessing the power of external data*. 1–7.

Abiodun, I. A., Alawida, M., Omolara, A. E., & Alabdulatif, A. (2022). Data provenance for cloud forensic investigations, security, challenges, solutions and future perspectives: A survey. *Journal of King Saud University - Computer and Information Sciences,* *34*(10), 10217–10245. https://doi.org/10.1016/j.jksuci.2022.10.018

Abioye, H. (2023). *The rise of 'big data' in cloud computing. Medium*. https://helendarmi-hd.medium.com/the-rise-of-big-data-in-cloud-computing-c597910f1649

Abrahams, T. O., Ewuga, K., Dawodu, S. O., Adegbite, O., Hassan, A. O., & Author, C. (2024). A review of cybersecurity strategies in modern organizations: Examining the evolution and effectiveness of cybersecurity measures for data protection. *Computer Science & IT Research Journal*, *5*(1), 1–25. https://doi.org/10.51594/csitrj.v5i.699

Acciarini, C., Cappa, F., Boccardelli, P., & Oriani, R. (2023). How can organizations leverage big data to innovate their business models? A systematic literature review. *Technovation*, *123*, 1–18. https://doi.org/10.1016/j.technovation.2023.102713

Achshah, R. M., & Prakash, T. R. (2021). A simple approach for selecting the best machine learning algorithm. *International Journal of Scientific & Engineering Research*, *12*((9) 2229-5518). http://www.ijser.org

Adler, A. I., & Painsky, A. (2021). Feature importance in gradient boosting trees with cross-validation feature selection. *Entropy*, 24(5), 1–17. https://doi.org/10.3390/e24050687

Adnan, M., Alarood, A. A. S., Uddin, M. I., & Rehman, I. (2022). Utilizing grid search cross-validation with adaptive boosting for augmenting performance of machine learning models. PeerJ *Computer Science*, *8, 1–29.* https://doi.org/10.7717/PEERJ-CS.803

Ahmed, J. A., Mustafa Majid, A. A., Konev, A., Kosachenko, T., & Shelupanov, A. (2023). A review on security analysis of cyber physical systems using machine learning. *Materials Today: Proceedings*, *80*, 2302–2306. https://doi.org/10.1016/J.MATPR.2021.06.320

Ahmed, S. F., Alam, M. S. Bin, Hoque, M., Lameesa, A., Afrin, S., Farah, T., Kabir, M., Shafiullah, G. M., & Muyeen, S. M. (2023). Industrial internet of things enabled technologies, challenges, and future directions. *Computers and Electrical Engineering*, *110*, 1–16. https://doi.org/10.1016/J.COMPELECENG.2023.108847

Ahsan, M., Nygard, K. E., Gomes, R., Chowdhury, M. M., Rifat, N., & Connolly, J. F. (2022). Cybersecurity threats and their mitigation approaches using machine learning—A review. *Journal of Cybersecurity and Privacy, 2*(3), 527–555. https://doi.org/10.3390/jcp2030027

Alam, A. (2023). *What is Machine Learning?* https://doi.org/10.5281/zenodo.8231580

Alazab, M., & Romdhani, I. (2020). *Blockchain for cybersecurity and privacy: Architectures, challenges, and applications (pp. 1–36)*. https://www.researchgate.net/publication/338411344

Alexei, A., Alexei, A., Arina, A., & Anatolie, A. (2021). Cyber security threat analysis in higher education institutions as a result of distance learning. *International Journal of Scientific & Technology Research, (3), 128–133*. www.ijstr.org

Al-Khasawneh, M. (2020). *Big Data Applications and Tools*. https://www.researchgate.net/publication/352413480

Allen, M. S., Robson, D. A., & Iliescu, D. (2023). Face validity: A critical but ignored component of scale construction in psychological assessment. *European Journal of Psychological Assessment, 39*(3), 153–156. https://doi.org/10.1027/1015-5759/a000777

Alnuaimi, A. F. A. H., & Albaldawi, T. H. K. (2024). An overview of machine learning classification techniques. *BIO Web of Conferences*, *97,* 1-24. https://doi.org/10.1051/bioconf/20249700133

Altalhan, M., Algarni, A., & Turki-Hadj Alouane, M. (2023). Imbalanced data problem in machine learning: A review. *IEEE Access, 11,* 1–16. https://doi.org/10.1109/ACCESS.2024.0429000

Alqudah, N., & Yaseen, Q. (2020). Machine learning for traffic analysis: A review. *Procedia Computer Science, 170,* 911–916. https://doi.org/10.1016/j.procs.2020.03.111

AlSalem, T. S., Almaiah, M. A., & Lutfi, A. (2023). Cybersecurity risk analysis in the IoT: A systematic review. *Electronics, 12*(18), 1–19. https://doi.org/10.3390/electronics12183958

Alshar'e, M. (2023). Cyber security framework selection: Comparison of NIST and ISO27001. *Applied Computing Journal,* 245–255. https://doi.org/10.52098/ACJ.202364

Alzahrani, A., Alsharif, A., Almalki, A., Alhazmi, A., Alzahrani, A., & Alzahrani, A. (2020). Big Data Analytics: A Comprehensive Review. *International Journal of Advanced Computer Science and Applications, 11(2), 1-14*

Anyanwu, A., Olorunsogo, T., Abrahams, T. O., Akindote, O., & Reis, O. (2024). Data confidentiality and integrity: A review of accounting and cybersecurity controls in superannuation organizations. *Computer Science & IT Research Journal, 5*(1), 237–253. https://doi.org/10.51594/csitrj.v5i1.735

Antunes, M., Maximiano, M., Gomes, R., & Pinto, D. (2021). Information security and cybersecurity management: A case study with SMEs in Portugal. *Journal of Cybersecurity and Privacy, 1*(2), 219–238. https://doi.org/10.3390/jcp1020012

Arena, F., & Pau, G. (2020). An overview of big data analysis. *Bulletin of Electrical Engineering and Informatics, 9*(4), 1646–1653. https://doi.org/10.11591/eei.v9i4.2359

Asaad, R. R., & Saeed, V. A. (2022). Cyber security threats, vulnerability, challenges and proposed solution. *Applied Computing Journal,* 227–244. https://doi.org/10.52098/acj.202260

Ashiku, L., & Dagli, C. (2021). Network intrusion detection system using deep learning. *Procedia Computer Science, 185,* 239–247. https://doi.org/10.1016/j.procs.2021.05.025

Aslan, Ö., Aktuğ, S. S., Ozkan-Okay, M., Yilmaz, A. A., & Akin, E. (2023). A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. *Electronics, 12*(6), 1–42. https://doi.org/10.3390/electronics12061333

Assegie, T. A., Sushma, S. J., Bhavya, B. G., & Padmashree, S. (2021). Correlation analysis for determining effective data in machine learning: Detection of heart failure. *SN Computer Science, 2*(3), 2–13. https://doi.org/10.1007/s42979-021-00617-5

Assiri, F. (2020). Methods for assessing, predicting, and improving data veracity: A survey. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, 9*(4), 5–30. https://doi.org/10.14201/adcaij202094530

Babbie, E. (2013). *The practice of social research* (14th ed.). Cengage Learning.

Bajpai, R., & Bajpai, S. (2014). Goodness of measurement: Reliability and validity. *International Journal of Medical Science and Public Health, 3*(2), 1–12. https://doi.org/10.5455/IJMSPH.2013.191120133

Barbierato, E., & Gatti, A. (2024). The challenges of machine learning: A critical review. *Electronics, 13*(2), 4–16. https://doi.org/10.3390/electronics13020416

Baron, J., Contreras, J. L., Husovec, M., Larouche, P., & Thumm, N. (2019). Making the rules: The governance of standard development organizations and their policies on intellectual property rights. *JRC Science for Policy Report, 8*(3), 345–370.

Bartneck, C., Lütge, C., Wagner, A., & Welsh, S. (2021). What is ethics? In *SpringerBriefs in Ethics* (pp. 17–26). Springer Nature. https://doi.org/10.1007/978-3-030-51110-4_3

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks, 5*(4), 537–550. https://doi.org/10.1109/72.298224

Belete, D. M., & Huchaiah, M. D. (2022). Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results. *International Journal of Computers and Applications, 44*(9), 875–886. https://doi.org/10.1080/1206212X.2021.1974663

Bellamy, N. (2015). Principles of clinical outcome assessment. *Rheumatology, 1*(2), 9–19. https://doi.org/10.1016/B978-0-323-09138-1.00002-4

Bongiovanni, I. (2019). The least secure places in the universe? A systematic literature review on information security management in higher education. *Computers & Security, 82,* 68–85. https://doi.org/10.1016/j.cose.2018.12.003

Bos, J. (2020). Confidentiality. In *Research ethics for students in the social sciences* (pp. 149–173). Springer. https://doi.org/10.1007/978-3-030-48415-6_7

Büyükkekececi, M., & Okur, M. C. (2023). A comprehensive review of feature selection and feature selection stability in machine learning. *Gazi University Journal of Science, 36*(4), 1506–1520. https://doi.org/10.35378/gujs.993763

Campbell, S., Greenwood, M., Prior, S., Shearer, T., Walkem, K., Young, S., Bywaters, D., & Walker, K. (2020). Purposive sampling: Complex or simple? Research case examples. *Journal of Research in Nursing, 25*(8), 652–661. https://doi.org/10.1177/1744987120927206

Casale, G., Zhang, E. Z., & Smirni, E. (2010). Trace data characterization and fitting for Markov modeling. *Performance Evaluation, 67*(2), 61–79. https://doi.org/10.1016/j.peva.2009.09.003

Chahal, S. (2023). AI-enhanced cyber incident response and recovery. *International Journal of Science and Research, 12*(3), 1795–1801. https://doi.org/10.21275/sr231003163025

Charandura, K. (2022). Cybersecurity in the education industry. Grant Thornton. https://www.grantthornton.co.za/Newsroom/cybersecurity-in-the-education-industry/

Chaudhary, P., Kashyap, V., Sonwal, N., Panwar, P., Dadheech, M., Bhatt, M. M., & Jain, M. M. (2023). Network traffic analysis using Wireshark. *International Advanced Research Journal in Science, Engineering and Technology, 10*(2), 1–5. https://doi.org/10.17148/IARJSET.2023.10201

Cheng, E. C. K., & Wang, T. (2022). Institutional strategies for cybersecurity in higher education institutions. *Information, 13*(4), 1–14. https://doi.org/10.3390/info13040192

Cheng, X. (2024). Signals and systems ins. *Compute Signal System, 1*(1), 8–13. https://soapubs.com/index.php/ICSS

Chidukwani, A., Zander, S., & Koutsakis, P. (2022). A survey on the cybersecurity of small-to-medium businesses: Challenges, research focus, and recommendations. *IEEE Access, 10,* 85701–85719. https://doi.org/10.1109/ACCESS.2022.3197899

Chigada, J., & Madzinga, R. (2021). Cyberattacks and threats during COVID-19: A systematic literature review. *SA Journal of Information Management, 23*(1), 1–11. https://doi.org/10.4102/sajim.v23i1.1277

Creswell, J. W. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches* (3rd ed.). Sage Publications.

Curtis, J., & Oxburgh, G. (2023). Understanding cybercrime in 'real world' policing and law enforcement. *Police Journal, 96*(4), 573–592. https://doi.org/10.1177/0032258X221107584

Dagada, R. (2024). The advancement of 4IR technologies and increasing cyberattacks in South Africa. *Southern African Journal of Security, 2,* 1–27. https://doi.org/10.25159/3005-4222/15157

Das, J. (2023). Pearson's correlation in predictive analytics and machine learning: Applications and limitations. *International Journal of Multidisciplinary Research (IJFMR), 5*(3), 1–20. http://www.ijfmr.com

Das, P., Rashid, M., Asif, A., Jahan, S., Khondoker, R., Ahmed, K., & Bui, F. M. (2024). STRIDE-based cybersecurity threat modeling, risk assessment and treatment of an infotainment high performance computing (HPC) system. *Preprints,* 1–18. https://doi.org/10.20944/preprints202401.0185.v1

Dasgupta, D., Akhtar, Z., & Sen, S. (2022). Machine learning in cybersecurity: A comprehensive survey. *Journal of Defense Modeling and Simulation, 19*(1), 57–106. https://doi.org/10.1177/1548512920951275

Datta, P. (2023). The promise and challenges of the fourth industrial revolution (4IR). *Journal of Information Technology Teaching Cases, 13*(1), 2–15. https://doi.org/10.1177/20438869211056938

Dhummad, S. (2025). The imperative of exploratory data analysis in machine learning. *Scholars Journal of Engineering and Technology, 13*(1), 30–44. https://doi.org/10.36347/sjet.2025.v13i01.005

Dighriri, M., Alfoudi, A. S. D., Lee, G. M., & Baker, T. (2017). Data traffic model in machine to machine communications over 5G network slicing. In *Proceedings of the 2016 9th International Conference on Developments in E-Systems Engineering* (pp. 239–244). IEEE. https://doi.org/10.1109/DESE.2016.54

Djafri, L., Moutassem, B., & Gafour, A. K. (2022). Big data veracity: Methods and challenges. *Abdel-Kader Gafour Journal, 23*(4), 687–719. https://www.researchgate.net/publication/361723024

Drost, E. (2011). Validity and reliability in social science research. *International Perspectives on Higher Education Research, 63*(56), 823–848. https://www.researchgate.net/publication/261473819

Elhanashi, A., Gasmi, K., Begni, A., Dini, P., Zheng, Q., & Saponara, S. (2023). Machine learning techniques for anomaly-based detection system on CSE-CIC-IDS2018 dataset. In *Lecture Notes in Electrical Engineering* (Vol. 1036, pp. 131–140). Springer. https://doi.org/10.1007/978-3-031-30333-3_17

Eliane Birba, D. (2020). A comparative study of data splitting algorithms for machine learning model selection (Bachelor's thesis). University of Borås. https://www.diva-portal.org/smash/record.jsf?dswid=8537&pid=diva2%3A1506870

Familoni, B. T. (2024). Cybersecurity challenges in the age of AI: Theoretical approaches and practical solutions. *Computer Science & IT Research Journal, 5*(3), 703–724. https://doi.org/10.51594/csitrj.v5i3.930

Foody, G. M. (2023). Challenges in the real world use of classification accuracy metrics: From recall and precision to the Matthews correlation coefficient. *PLoS ONE, 18*(10), e0291908. https://doi.org/10.1371/journal.pone.0291908

Franceschi, L., Donini, M., Perrone, V., Klein, A., Archambeau, C., Seeger, M., Pontil, M., & Frasconi, P. (2024). Hyperparameter optimization in machine learning. *Machine Learning,* 1–142. http://arxiv.org/abs/2410.22854

García Lozano, M., Brynielsson, J., Franke, U., Rosell, M., Tjörnhammar, E., Varga, S., & Vlassov, V. (2020). Veracity assessment of online data. *Decision Support Systems, 129,* 113132. https://doi.org/10.1016/j.dss.2019.113132

Garoufallou, E., & Gaitanou, P. (2021). Big data: Opportunities and challenges in libraries—A systematic literature review. *College & Research Libraries, 82*(3), 410–435. https://doi.org/10.5860/crl.82.3.410

Gonaygunta, H., Nadella, G. S., Meduri, K., Pawar, P. P., & Kumar, D. (2024). The detection and prevention of cloud computing attacks using artificial intelligence technologies. *International Journal of Multidisciplinary Research and Publications (IJMRAP), 6*(8), 191–193. https://www.researchgate.net/publication/378208392

Gupta, P. M. (2023). The role of big data in smart healthcare. *International Journal of Internet of Things, 11*(1), 11–18. https://doi.org/10.5923/j.ijit.20231101.02

Hadi, A. A., & Hadi, A. M. (2024). Improving cybersecurity with random forest algorithm-based big data intrusion detection system: A performance analysis. *AIP Conference Proceedings, 30*(1), 13044–13059. https://doi.org/10.1063/5.0191707

van der Ham, J. (2021). Toward a better understanding of “cybersecurity.” *Digital Threats: Research and Practice, 2*(3), 1–3. https://doi.org/10.1145/3442445

Hariri, R. H., Fredericks, E. M., & Bowers, K. M. (2019). Uncertainty in big data analytics: Survey, opportunities, and challenges. *Journal of Big Data, 6*(1), 1–16. https://doi.org/10.1186/s40537-019-0206-3

Hasan, R., Biswas, B., Samiun, M., Saleh, M. A., Prabha, M., Akter, J., Joya, F. H., & Abdullah, M. (2025). Enhancing malware detection with feature selection and scaling techniques using machine learning models. *Scientific Reports, 15*(1), 2291. https://doi.org/10.1038/s41598-025-93447-x

Hero, A., Kar, S., Moura, J., Neil, J., Poor, H. V., Turcotte, M., & Xi, B. (2023). Statistics and data science for cybersecurity. *Harvard Data Science Review, 5*(1), 106–231. https://doi.org/10.1162/99608f92.a42024d0

Hindy, H., Brosset, D., Bayne, E., Seeam, A. K., Tachtatzis, C., Atkinson, R., & Bellekens, X. (2020). A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access, 8,* 104650–104675. https://doi.org/10.1109/ACCESS.2020.3000179

Ilemobayo, J., Durodola, O., Alade, O., Awotunde, O. J., Olanrewaju, A. T., Falana, O., Ogungbire, A., Osinuga, A., Ogunbiyi, D., Ifeanyi, A., Odezuligbo, I. E., & Edu, O. E. (2024). Hyperparameter tuning in machine learning: A comprehensive review. *Journal of Engineering Research and Reports*, *26*(6), 388–395. https://doi.org/10.9734/jerr/2024/v26i61188

Islam, R., Patamsetti, V., Gadhi, A., Gondu, R. M., Bandaru, C. M., Kesani, S. C., & Abiona, O. (2023). The future of cloud computing: Benefits and challenges. *International Journal of Communications, Network and System Sciences, 16*(4), 53–65. https://doi.org/10.4236/ijcns.2023.164004

Jaipong, P., Siripipattanakul, S., Sriboonruang, P., & Sitthipon, T. (2023). A review of metaverse and cybersecurity in the digital era. *International Journal of Computing Sciences Research, 7,* 1125–1132. https://doi.org/10.25147/ijcsr.2017.001.1.122

James, S., Harbron, C., Branson, J., & Sundler, M. (2021). Synthetic data use: Exploring use cases to optimise data utility. *Discover Artificial Intelligence, 1*(1), Article 16. https://doi.org/10.1007/s44163-021-00016-y

Jia, W., Sun, M., Lian, J., & Hou, S. (2022). Feature dimensionality reduction: A review. *Complex & Intelligent Systems, 8*(3), 2663–2693. https://doi.org/10.1007/s40747-021-00637-x

Johannesson, P., & Perjons, E. (2014). Research strategies and methods. In *An introduction to design science* (pp. 39–73). Springer. https://doi.org/10.1007/978-3-319-10632-8_3

Johnkutty, J., Haque, S., & Davis, J. (2024). Exploratory data analysis using machine learning—Behaviour based safety. *Journal of Electrical Systems, 20*(7), 744–754.

Jordon, J., Szpruch, L., Houssiau, F., Bottarelli, M., Cherubin, G., Maple, C., Cohen, S. N., & Weller, A. (2022). Synthetic data—What, why and how? *arXiv Preprint*. https://arxiv.org/pdf/2205.03257

Jurafsky, D., & Martin, J. H. (2025). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition with language models* (3rd ed., draft). Draft summary of contents (pp. 638–641).

Kabir, M. H. (2023). Data network security: Introduction. ResearchGate. https://doi.org/10.13140/RG.2.2.32941.26081

Kandel, B. (2020). Qualitative versus quantitative research. *Marsyangdi Journal, 1*(1), 1–5. https://www.researchgate.net/publication/352550744

Kara, I. (2021). Read the digital fingerprints: Log analysis for digital forensics and security. *Computer Fraud & Security, 2021*(7), 11–16. https://doi.org/10.1016/S1361-3723(21)00074-9

Kaur, R., Gabrijelčič, D., & Klobučar, T. (2023). Artificial intelligence for cybersecurity: Literature review and future research directions. *Information Fusion, 97,* 101804. https://doi.org/10.1016/j.inffus.2023.101804

Khan, N. A., Brohi, S. N., & Zaman, N. (2020). Ten deadly cyber security threats amid COVID-19 pandemic. *TechRxiv Preprint,* 1–6. https://doi.org/10.36227/techrxiv.12278792.v1

Khraisat, A., & Alazab, A. (2021). A critical review of intrusion detection systems in the Internet of Things: Techniques, deployment strategy, validation strategy, attacks, public datasets, and challenges. *Cybersecurity, 4*(1), 1–24. https://doi.org/10.1186/s42400-021-00077-7

Kilincer, I. F., Ertam, F., & Sengur, A. (2021). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks, 188,* 107840. https://doi.org/10.1016/j.comnet.2021.107840

Kim, K. H., Kim, K., & Kim, H. K. (2022). STRIDE-based threat modeling and DREAD evaluation for the distributed control system in the oil refinery. *ETRI Journal, 44*(6), 991–1003. https://doi.org/10.4218/etrij.2021-0181

Kim, S. H., Eom, J. H., & Chung, T. M. (2013). Big data security hardening methodology using attributes relationship. In *2013 International Conference on Information Science and Applications (ICISA)* (pp. 1–2). IEEE. https://doi.org/10.1109/ICISA.2013.6579427

Knutmejer, V., Leónortiz, M., Markovic, T., & Elfving, H. (2023). Feature selection for sensor failure detection in manufacturing environments [Master's thesis, Mälardalen University].

Kubai, E. (2019). Reliability and validity of research instruments. ResearchGate. https://www.researchgate.net/publication/335827941

Kumar, H., Soh, P. J., & Ismail, M. A. (2022). Big data streaming platforms: A review. *Iraqi Journal for Computer Science and Mathematics, 3*(2), 95–100. https://doi.org/10.52866/ijcsm.2022.02.01.010

Kumar, I. (2023). Emerging threats in cybersecurity: A review article. *International Journal of Applied and Natural Sciences, 1*(1), 1–8. http://bluemarkpublishers.com/index.php/IJANS

Kumar, S., & Chong, I. (2018). Correlation analysis to identify the effective data in machine learning: Prediction of depressive disorder and emotion states. *International Journal of Environmental Research and Public Health, 15*(12), 2729. https://doi.org/10.3390/ijerph15122907

Kumatongo, B., & Muzata, K. K. (2021). Research paradigms and designs with their application in education. *Journal of Lexicography and Terminology, 5*(1), 16–32. https://journals.unza.zm/index.php/jlt

Kune, R., Konugurthi, P. K., Agarwal, A., Chillarige, R. R., & Buyya, R. (2016). The anatomy of big data computing. *Software: Practice and Experience, 46*(1), 79–105. https://doi.org/10.1002/spe.2374

Lavanya, S., Mythili, K., & Kannimuthu, S. (2020). An integration of big data analytics and cyber security: A panoramic survey. *International Journal of Advanced*

Research in Engineering and Technology (IJARET), 11(9), 747–754. https://doi.org/10.34218/IJARET.11.9.2020.074

Leenen, L., & Meyer, T. (2019). Artificial intelligence and big data analytics in support of cyber defense. In *Cyber defense and situational awareness* (pp. 42–63). IGI Global. https://doi.org/10.4018/978-1-5225-8304-2.ch002

Leevy, J. L., & Khoshgoftaar, T. M. (2020). A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 big data. *Journal of Big Data, 7*(1), 1–19. https://doi.org/10.1186/s40537-020-00382-x/tables/4

Lewis, B. R., Templeton, G. F., & Byrd, T. A. (2005). A methodology for construct development in MIS research. *European Journal of Information Systems, 14*(4), 388–400. https://doi.org/10.1057/palgrave.ejis.3000552

Li, F. (2024). Application and challenges of artificial intelligence in cybersecurity. *Applied and Computational Engineering, 47*(1), 262–268. https://doi.org/10.54254/2755-2721/47/20241480

Li, Y., & Liu, Q. (2021). A comprehensive review study of cyber-attacks and cyber security: Emerging trends and recent developments. *Energy Reports, 7,* 8176–8186. https://doi.org/10.1016/j.egyr.2021.08.126

Long, K. S., & MITRE Corporation, T. (2021). Cybersecurity network monitoring challenge in commercial service provider clouds. Defense Technical Information Center. https://apps.dtic.mil/sti/html/trecms/AD1133628/

Madugula, S., Pratapagiri, S., Phridviraj, M. S. B., Rao, V. C. S., Polala, N., & Kumaraswamy, P. (2023). Big data for the comprehensive data analysis of IT organizations. *Journal of High Technology Management Research, 34*(2), 100465. https://doi.org/10.1016/j.hitech.2023.100465

Mahdi, A. A. (2024). Machine learning applications of network security enhancement: Review. *Computer Science & IT Research Journal*, *5*(10), 2283–2300. https://doi.org/10.51594/csitrj.v5i10.1635

Malathi, C., & Padmaja, I. N. (2023). Identification of cyber attacks using machine learning in smart IoT networks. *Materials Today: Proceedings, 80,* 2518–2523.

Marwah, M., Narayanan, A., Jou, S., Arlitt, M., & Pospelova, M. (2024). Is F1 score suboptimal for cybersecurity models? Introducing Cscore, a cost-aware alternative for model assessment. *Cybersecurity Analytics Journal, 1,* 1–15.

Mcanyana, W., Brindley, C., & Seedat, Y. (2020). Insight into the cyberthreat landscape in South Africa. *South African Journal of Information Security, 5*(1), 1–12.

McKane, J. (2019, September 16). South African banks hit by massive DDoS attack. *MyBroadband.* https://mybroadband.co.za/news/banking/324881-south-african-banks-hit-by-massive-ddos-attack.html

Menze, B. H., Kelm, B. M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W., & Hamprecht, F. A. (2009). A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC Bioinformatics, 10,* 213. https://doi.org/10.1186/1471-2105-10-213

Mertler, C. A. (2016). Quantitative research methods. In *Quantitative research methods* (pp. 1–37). SAGE Publications. https://us.sagepub.com/sites/default/files/upm-binaries/70019_Mertler_Chapter_7.pdf

Mitchell, O., & Osazuwa, C. (2023). Confidentiality, Integrity, and Availability in Network Systems: A Review of Related Literature. International Journal of Innovative Science and Research Technology, 8.

Mienye, I. D., & Jere, N. (2024). A survey of decision trees: Concepts, algorithms, and applications. *IEEE Access, 12,* 86716–86727. https://doi.org/10.1109/ACCESS.2024.3416838

Mishra, N., & Pandya, S. (2021). Internet of Things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access, 9,* 59353–59377. https://doi.org/10.1109/ACCESS.2021.3073408

Mohamad, N. H., Saidin, N. B., & Zaidi, M. I. H. (2023). Data security and privacy issues in cloud computing: Challenges and solutions review. *TechRxiv Preprint,* 1–21. https://doi.org/10.36227/techrxiv.170327865.59737799/v1

Mohamed, H., El Bolock, A., & Sabty, C. (2023). IntrusionHunter: Detection of cyber threats in big data. In *Proceedings of the 15th International Conference on Cybersecurity and Data Mining* (pp. 311–318). SCITEPRESS. https://doi.org/10.5220/0012081900003541

Mohammed, M. A. (2023). Effect of using numerical data scaling on supervised machine learning performance. *Journal of Computational Intelligence and Applications, 9*(2), 45–58.

Mohammed, S. A. (2016). Big data networking: Requirements, architecture and issues. *International Journal of Wireless & Mobile Networks, 8*(6), 35–49. https://doi.org/10.5121/ijwmn.2016.8604

Mosier, C. I. (1947). A critical examination of the concepts of face validity. *Educational and Psychological Measurement, 7*(2), 191–205. https://doi.org/10.1177/001316444700700201

Moura, J., & Serrão, C. (2019). Security and privacy issues of big data. In *Cyber law, privacy, and security* (pp. 375–407). IGI Global. https://doi.org/10.4018/978-1-5225-8897-9.ch019

Moya, A. (2019, February 14). City of Joburg hit by cyber attack. *ITWeb.* https://www.itweb.co.za/article/city-of-joburg-hit-by-cyber-attack/dgp45qaG8gZ7X9l8

Moyo, A. (2021, July 12). Justice Department battles to contain ransomware attack. *ITWeb.* https://www.itweb.co.za/article/justice-department-battles-to-contain-ransomware-attack/DZQ58vVPOB1MzXy2

Mpekoa, N. (2024). An analysis of cybersecurity architectures. In *Proceedings of the 19th International Conference on Cyber Warfare and Security (ICCWS 2024)* (pp. 200–207).

Mungadze, S. (2021, September 8). University of Mpumalanga thwarts R100m hack attempt. *ITWeb.* https://www.itweb.co.za/article/university-of-mpumalanga-thwarts-r100m-hack-attempt/KjlyrvW1jmmMk6am

Nassar, A., & Kamal, M. (2021). Machine learning and big data analytics for cybersecurity threat detection: A holistic review of techniques and case studies. *Journal of Artificial Intelligence and Machine Learning in Management, 5*(1), 51–63.

Natarajan, R., Ranjith, C. P., Mohideen, M. K., Gururaj, H. L., Flammini, F., & Thangarasu, N. (2024). Utilizing a machine learning algorithm to choose a significant traffic identification system. *International Journal of Information Management Data Insights, 4*(1), 100218. https://doi.org/10.1016/j.jjimei.2024.100218

Nnadozie, C. E., & Zakka, B. E. (2024). The importance of cybersecurity for a safer society. *International Journal of Research Publication and Reviews, 5*(5), 9979–9986. https://doi.org/10.55248/gengpi.5.0524.1371

Nunnally, J. C., & Bernstein, I. H. (1994). *Psychometric theory* (3rd ed.). McGraw-Hill. https://www.scirp.org/reference/referencespapers?referenceid=1017362

Owusu-Adjei, M., Hayfron-Acquah, J. B., Frimpong, T., & Abdul-Salaam, G. (2023). A systematic review of prediction accuracy as an evaluation measure for determining machine learning model performance in healthcare systems. https://doi.org/10.1101/2023.06.01.23290837

Oyetoro, A., Mart, J., & Amah, U. (2023). Using machine learning techniques (random forest and neural network) to detect cyber attacks. *Science Open Preprints,* 1–11. https://doi.org/10.14293/pr2199.000059.v1

Paper, I., Zhou, Y., Tu, F., Sha, K., & Chen, H. (2025). A survey on data quality dimensions and tools for machine learning. *GitHub Repository,* 120–131. https://github.com/haihua0913/awesome-dq4ml

Park, Y. S., Konge, L., & Artino, A. R. (2020). The positivism paradigm of research. *Academic Medicine, 95*(5), 690–694. https://doi.org/10.1097/ACM.0000000000003093

Parker, D. B. (1998). *Fighting computer crime: A new framework for protecting information.*

Parker, D. B. (2010). Our excessively simplistic information security model and how to fix it. *The ISSA Journal, 8*(7), 12–21.

Pathade, C., & Bhosale, T. (2023). Cyber threats prediction using machine learning. *International Research Journal of Engineering and Technology (IRJET).* www.irjet.net

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Cournapeau, D., Brucher, M., & Perrot, M. (2011). Scikit-learn: Machine Learning in Python Pedregosa, Varoquaux, Gramfort et al. *Journal of Machine Learning Research*, *12*, 2825–2830. http://scikit-learn.org.

Phair, D., & Warren, K. (2021). Saunders' research onion: Explained simply. 1.

Pieterse, H. (2021). The cyber threat landscape in South Africa: A 10-year review. *The African Journal of Information and Communication, 28,* 1–21. https://doi.org/10.23962/10539/32213

Qureshi, H. N., Manalastas, M., Ijaz, A., Imran, A., Liu, Y., & Al Kalaa, M. O. (2022). Communication requirements in 5G-enabled healthcare applications: Review

and considerations. *Healthcare (Switzerland), 10*(2), 201–293. https://doi.org/10.3390/healthcare10020293

Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports, 14*(1). https://doi.org/10.1038/s41598-024-56706-x

Rawat, R., & Yadav, R. (2021). Big data: Big data analysis, issues and challenges and technologies. *IOP Conference Series: Materials Science and Engineering, 1022*(1). https://doi.org/10.1088/1757-899X/1022/1/012014

Reddy, B. A. (2025). The Science of Data Splits: How Train-Test Strategies Impact Real-World Model Reliability. *International Journal of Research Publication and Reviews Journal Homepage: Www.Ijrpr.Com*, *6*, 1636–1647. www.ijrpr.com

Reimer, A. P., & Madigan, E. A. (2019). Veracity in big data: How good is good enough. *Health Informatics Journal, 25*(4), 1290–1298. https://doi.org/10.1177/1460458217744369

Rengasamy, D., Mase, J. M., Kumar, A., Rothwell, B., Torres, M. T., Alexander, M. R., Winkler, D. A., & Figueredo, G. P. (2022). Feature importance in machine learning models: A fuzzy information fusion approach. *Neurocomputing, 511*, 163–174. https://doi.org/10.1016/j.neucom.2022.09.053

Rosenthal, R., & Rosnow, R. L. (2008). *Essentials of behavioral research: Methods and data analysis.* McGraw-Hill.

Roy, R., Sukumar, G. M., Philip, M., & Gopalakrishna, G. (2023). Face, content, criterion and construct validity assessment of a newly developed tool to assess and classify work–related stress (TAWS–16). *PLoS ONE, 18*(1), 1–16. https://doi.org/10.1371/journal.pone.0280189

Saeed, N., & Husamaldin, L. (2021). Big data characteristics (V's) in industry. *Iraqi Journal of Industrial Research, 8*(1), 1–9. https://doi.org/10.53523/ijoirvol8i1id52

Saeed, S., Altamimi, S. A., Alkayyal, N. A., Alshehri, E., & Alabbad, D. A. (2023). Digital transformation and cybersecurity challenges for businesses resilience: Issues and recommendations. *Sensors, 23*(15), 123–266. https://doi.org/10.3390/s23156666

Sahoo, K., Samal, A. K., Pramanik, J., & Pani, S. K. (2019). Exploratory data analysis using Python. *International Journal of Innovative Technology and Exploring Engineering, 8*(12), 4727–4735. https://doi.org/10.35940/ijitee.L3591.1081219

Salman, H. A., Kalakech, A., & Steiti, A. (2024). Random forest algorithm overview. *Babylonian Journal of Machine Learning*, 69–79. https://doi.org/10.58496/bjml/2024/007

Sandhu, A. K. (2022). Big data with cloud computing: Discussions and challenges. *Big Data Mining and Analytics, 5*(1), 32–40. https://doi.org/10.26599/bdma.2021.9020016

Sandra, G. (2019). Big data issues from data challenges perspective. *IJCST, 10.*

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science, 2*(3). https://doi.org/10.1007/s42979-021-00592-x

Sarker, I. H. (2023). Machine learning for intelligent data analysis and automation in cybersecurity: Current and future prospects. *Annals of Data Science, 10*(6), 1473–1498. https://doi.org/10.1007/s40745-022-00444-2

Sathyanarayanan, S. (2024). Confusion Matrix-Based Performance Evaluation Metrics. *African Journal of Biomedical Research*, 4023–4031. https://doi.org/10.53555/ajbr.v27i4s.4345

Saurombe, M. D., Rayners, S. S., Mokgobu, K. A., & Manka, K. (2022). The perceived influence of remote working on specific human resource management outcomes during COVID-19 pandemic. *SA Journal of Human Resource Management.* https://doi.org/10.4102/sajhrm.v20i0.2033

Sawik, T. (2020). A linear model for optimal cybersecurity investment in Industry 4.0 supply chains. *International Journal of Production Research, 60*(4), 1368–1385. https://doi.org/10.1080/00207543.2020.1856442

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A systematic literature review on applying CRISP-DM process model. *Procedia Computer Science, 181*, 526–534. https://doi.org/10.1016/j.procs.2021.01.199

Schlosser, T., Friedrich, M., Meyer, T., Kowerko, D., & Professorship, J. (2024). A consolidated overview of evaluation and performance metrics for machine learning and computer vision. https://doi.org/10.13140/RG.2.2.14331.69928

Shaikh, F. A., & Siponen, M. (2023). Information security risk assessments following cybersecurity breaches: The mediating role of top management attention to cybersecurity. *Computers & Security, 124*, 1–18. https://doi.org/10.1016/j.cose.2022.102974

Shani, C., Zarecki, J., & Shahaf, D. (2022). The lean data scientist: Recent advances towards overcoming the data bottleneck. *Communications of the ACM, 66*(2), 92–102. https://doi.org/10.1145/3551635

Shani, C., Zarecki, J., & Shahaf, D. (2023). The lean data scientist: Recent advances toward overcoming the data bottleneck. *Communications of the ACM, 66*(2), 92–102. https://doi.org/10.1145/3551635/suppl_file/p92-shani-supp.pdf

Sharma, P., & Barua, S. (2023). From data breach to data shield: The crucial role of big data analytics in modern cybersecurity strategies. *International Journal of Information and Cybersecurity, 7*(9), 31–59. https://publications.dlpress.org/index.php/ijic/article/view/46

Shimaoka, A. M., Ferreira, R., & Goldman, A. (2023). The evolution of CRISP-DM for data science: Methods, processes and frameworks. *SBC Reviews on Computer Science.* https://doi.org/10.13140/RG.2.2.22493.42721

Singh, A. (2023). Big data technologies for cyber security in the digital era. *SSRN Electronic Journal,* 1–7. https://doi.org/10.2139/ssrn.4629044

Sinha, S., & Lee, Y. M. (2024). Challenges with developing and deploying AI models and applications in industrial systems. *Discover Artificial Intelligence, 4*(1), 93–127. https://doi.org/10.1007/s44163-024-00151-2

Sivakumar, M., Parthasarathy, S., & Padmapriya, T. (2024). Trade-off between training and testing ratio in machine learning for medical image processing. *PeerJ Computer Science.* https://doi.org/10.7717/peerj-cs.2245

Srihith, I. D., Lakshmi, P. V., Donald, A. D., Srinivas, T. A. S., & Thippanna, G. (2023). A forest of possibilities: Decision trees and beyond. *Journal of Advancement in Parallel Computing, 6*(3), 1–12. https://doi.org/10.5281/zenodo.8372196

Srivastva, R. K. (2023). Big data and cyber security: Challenges and solutions. *China Communications, 13*(3), 193–202.

Strippel, C. (2019). Log file analysis as a method for automated measurement of internet usage (pp. 1–25). ResearchGate. https://www.researchgate.net/publication/335883136

Sun, X., He, Y., Wu, D., & Huang, J. Z. (2023). Survey of distributed computing frameworks for supporting big data analysis. *Big Data Mining and Analytics, 6*(2), 154–169. https://doi.org/10.26599/BDMA.2022.9020014

Swarooprani, K. (2022). A study of research methodology. *International Journal of Scientific Research in Science, Engineering and Technology,* 537–543. https://doi.org/10.32628/IJSRSET2293175

Taherdoost, H. (2018). Validity and reliability of the research instrument: How to test the validation of a questionnaire/survey in a research. *SSRN Electronic Journal.* https://doi.org/10.2139/ssrn.3205040

Taherdoost, H. (2020). Different types of data analysis: Data analysis methods and techniques in research projects. *International Journal of Academic Research in Management, 9*(1). www.elvedit.com

Taherdoost, H. (2022). Understanding cybersecurity frameworks and information security standards: A review and comprehensive overview. *Electronics, 11*(14). https://doi.org/10.3390/electronics11142181

Tariq, M. A. (2024). A study on comparative analysis of feature selection algorithms for students grades prediction. *Journal of Information and Organizational Sciences, 48*(1), 133–147. https://doi.org/10.31341/jios.48.1.7

Task Force, J. (2020). *NIST special publication 800-53 revision 5: Security and privacy controls for information systems and organizations.* https://doi.org/10.6028/NIST.SP.800-53r5

Tawalbeh, L., Muheidat, F., Tawalbeh, M., & Quwaider, M. (2020). IoT privacy and security: Challenges and solutions. *Applied Sciences, 10*(12), 16–23. https://doi.org/10.3390/app10124102

Thale, B., & Vijayakumar, S. (2022). Challenge and security issues using big data: A survey. *International Journal for Research in Applied Science & Engineering Technology (IJRASET), 10,* 1–7. www.ijraset.com

Thayyib, P. V., Mamilla, R., Khan, M., Fatima, H., Asim, M., Anwar, I., Shamsudheen, M. K., & Khan, M. A. (2023). State-of-the-art of artificial intelligence and big data analytics reviews in five different domains: A bibliometric summary. *Sustainability, 15*(5), 26–40. https://doi.org/10.3390/su15054026

Thomas, N. S., & Kaliraj, · S. (2024). *An Improved and Optimized Random Forest Based Approach to Predict the Software Faults*. *5*, 530. https://doi.org/10.1007/s42979-024-02764-x

Thwaini, M. H. (2022). Anomaly detection in network traffic using machine learning for early threat detection. *Data and Metadata, 1,* 34. https://doi.org/10.56294/dm202272

Tiwari, S., Wee, H. M., & Daryanto, Y. (2018). Big data analytics in supply chain management between 2010 and 2016: Insights to industries. *Computers & Industrial Engineering, 115,* 319–330. https://doi.org/10.1016/j.cie.2017.11.017

Toshniwal, R., Dastidar, K. G., & Nath, A. (2015). Big Data Security Issues and Challenges. *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, *2*(2), 2349–2163. www.ijirae.com

Uhr, A. (2023). *Degree Project in Software Engineering for Distributed Systems Second Cycle 30.0 credits Time Series Analysis and Binary Classification in a Car-Sharing Service Application of data-driven methods for analysing trends, seasonality, residuals and prediction of user demand*, 65-77.

Vujović, Ž. (2021). Classification Model Evaluation Metrics. *International Journal of Advanced Computer Science and Applications*, *12*(6), 599–606. https://doi.org/10.14569/IJACSA.2021.0120670

Varoquaux, G., & Colliot, O. (2023). Evaluating Machine Learning Models and Their Diagnostic Value. *Neuromethods*, *197*, 601–630. https://doi.org/10.1007/978-1-0716-3195-9_20

Verma, R. (2024). Cybersecurity challenges in the era of digital transformation. *Transdisciplinary threads: crafting the future through multidisciplinary research, 1(1)*. https://doi.org/10.25215/9392917848.20

Verma, R. (2024). *The CIA Triad: Pillars of Cybersecurity Defense*. https://www.linkedin.com/pulse/cia-triad-pillars-cybersecurity-defense-rahul-verma-ng6rc/

Wan, S., Saxe, J., Gomes, C., Chennabasappa, S., Rath, A., Sun, K., & Wang, X. (2024). *Bridging the Gap: A Study of AI-based Vulnerability Management between Industry and Academia,* 80-87. http://arxiv.org/abs/2405.02435

Wang, L., & Jones, R. (2021). Big Data Analytics in Cyber Security: Network Traffic and Attacks. *Journal of Computer Information Systems*, *61*(5), 410–417. https://doi.org/10.1080/08874417.2019.1688731

Wirth, R., & Hippo, J. *(2000).* CRISP-DM Towards a Standard Process Model for Data Mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, Manchester*, 29-40.

Wook, M., Hasbullah, N. A., Zainudin, N. M., Jabar, Z. Z. A., Ramli, S., Razali, N. A. M., & Yusop, N. M. M. (2021). Exploring big data traits and data quality dimensions for big data analytics application using partial least squares structural equation modelling. Journal of Big Data, 8(1). https://doi.org/10.1186/s40537-021-00439-5

Xu, Z., Huang, G., Weinberger, K. Q., & Zheng, A. X. (2019). Gradient boosted feature selection. *In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining,* 522-531. https://doi.org/10.1145/2623330.2623635

Yang, S., & Berdine, G. (2024). Confusion matrix. *The Southwest Respiratory and Critical Care Chronicles*, *12*(53), 75–79. https://doi.org/10.12746/SWRCCC.V12I53.1391

Yang, L., Li, J., Elisa, N., Prickett, T., & Chao, F. (2019). Towards Big data Governance in Cybersecurity. *Data-Enabled Discovery and Applications*, *3*(1), 1-10. https://doi.org/10.1007/s41688-019-0034-9

Yilmaz, E., & Can, O. (2024). Unveiling Shadows: Harnessing Artificial Intelligence for Insider Threat Detection. *Engineering, Technology and Applied Science Research*, *14*(2), 13341–13346. https://doi.org/10.48084/etasr.6911

Yokoyama, R., & Arima, C. H. (2023). Application of the Threat Modeling Method In an Operating System. *Revista de Gestão e Secretariado (Management and Administrative Professional Review)*, *14*(4), 5102–5122. https://doi.org/10.7769/GESEC.V14I4.1645

Zhang, L., Taal, A., Cushing, R., de Laat, C., & Grosso, P. (2022). A risk-level assessment system based on the STRIDE/DREAD model for digital data marketplaces. *International Journal of Information Security*, *21*(3), 509–525. https://doi.org/10.1007/s10207-021-00566-3

Zhou, Y., Tu, F., Sha, K., Ding, J., & Chen, H. (2024). *A Survey on Data Quality Dimensions and Tools for Machine Learning,* 120-131. https://arxiv.org/pdf/2406.19614

Zografopoulos, I., Ospina, J., Liu, X., & Konstantinou, C. (2021). Cyber-Physical Energy Systems Security: Threat Modeling, Risk Assessment, Resources,

Metrics, and Case Studies. *IEEE Access*, *9*, 29775–29818. https://doi.org/10.1109/ACCESS.2021.3058403

Zohrabi, M. (2013). Mixed Method Research: Instruments, Validity, Reliability and Reporting Findings. *Theory and practice in language studies*, 3(2), 254-262. https://doi.org/10.4304/tpls.3.2.254-262

APPENDICES

Appendix A: Final Code

```
print("--- Loading Libraries ---")
import pandas as pd
import numpy as np
import os
import time
import joblib
import json
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import (accuracy_score, precision_score, recall_score,
                             f1_score, roc_auc_score,
confusion_matrix,classification_report)
from sklearn.feature_selection import mutual_info_classif
from imblearn.over_sampling import SMOTE
import shap
import seaborn as sns
import matplotlib.pyplot as plt
import traceback
from pandas.api.types import is_object_dtype, CategoricalDtype

try:
    from IPython.display import display
except ImportError:
    display = print

print("Libraries loaded.")

# --- Configuration Constants ---
print("\n--- Setting Configuration ---")
DATASET_PATH = 'Cyber_Threat_Detection_Dataset.csv'
OUTPUT_DIR = 'Final_Model'
```

```
TARGET_COLUMN = 'is_malicious'

PERFORM_HYPERPARAMETER_TUNING = True
TUNING_CV_FOLDS = 5
TUNING_SCORING_METRIC = 'f1'

# Path constants
DUPLICATE_REMOVED_DATA_PATH = os.path.join(OUTPUT_DIR,
'cyber_threat_deduplicated.csv')
PREPROCESSED_DATA_PATH = os.path.join(OUTPUT_DIR,
'cyber_threat_preprocessed.csv')
FEATURE_ENGINEERED_DATA_PATH = os.path.join(OUTPUT_DIR,
'cyber_threat_feature_engineered.csv')
SELECTED_FEATURES_DATA_PATH = os.path.join(OUTPUT_DIR,
'cyber_threat_selected_features.csv')
ORIGINAL_CORR_PATH = os.path.join(OUTPUT_DIR, 'original_correlation_results.csv')
ORIGINAL_CORR_PLOT_PATH = os.path.join(OUTPUT_DIR,
'original_correlation_plot.png')
FEATURE_ENG_CORR_PATH = os.path.join(OUTPUT_DIR,
'feature_engineer_correlation_results.csv')
FEATURE_ENG_CORR_PLOT_PATH = os.path.join(OUTPUT_DIR,
'feature_engineered_correlation_plot.png')
LOW_VARIANCE_FEATURES_PATH = os.path.join(OUTPUT_DIR,
'low_variance_features.csv')
SELECTED_CORR_PATH = os.path.join(OUTPUT_DIR,
'correlation_after_feature_selection.csv')
SELECTED_CORR_PLOT_PATH = os.path.join(OUTPUT_DIR,
'selected_features_correlation_plot.png')
MUTUAL_INFO_PATH = os.path.join(OUTPUT_DIR, 'feature_importance_mutual_info.csv')
MUTUAL_INFO_PLOT_PATH = os.path.join(OUTPUT_DIR,
'mutual_info_importance_plot.png')
RF_IMPORTANCE_PATH = os.path.join(OUTPUT_DIR,
'feature_importance_random_forest.csv')
RF_IMPORTANCE_PLOT_PATH = os.path.join(OUTPUT_DIR, 'rf_importance_plot.png')
GB_IMPORTANCE_PATH = os.path.join(OUTPUT_DIR,
'feature_importance_gradient_boosting.csv')
GB_IMPORTANCE_PLOT_PATH = os.path.join(OUTPUT_DIR, 'gb_importance_plot.png')
COMBINED_IMPORTANCE_PATH = os.path.join(OUTPUT_DIR,
'combined_feature_importance.csv')
INITIAL_PERFORMANCE_METRICS_PATH = os.path.join(OUTPUT_DIR,
f'initial_model_performance_metrics{"_tuned" if PERFORM_HYPERPARAMETER_TUNING
else ""}.csv')
THRESHOLD_ANALYSIS_PATH = os.path.join(OUTPUT_DIR, 'threshold_analysis.csv')
FINAL_METRICS_PATH = os.path.join(OUTPUT_DIR, 'final_chosen_model_metrics.csv')
```

```
CM_PLOT_DIR = os.path.join(OUTPUT_DIR, "confusion_matrices")
MODEL_CHOICE_INFO_PATH = os.path.join(OUTPUT_DIR, 'final_model_choice_info.json')
LABEL_ENCODERS_PATH = os.path.join(OUTPUT_DIR, 'label_encoders.joblib')
TRAINING_MEDIANS_PATH = os.path.join(OUTPUT_DIR, 'training_medians.joblib')
SCALER_PATH = os.path.join(OUTPUT_DIR, 'scaler.pkl')
FEATURE_LIST_PATH = os.path.join(OUTPUT_DIR, 'selected_feature_list.pkl')
FINAL_MODEL_PATH = os.path.join(OUTPUT_DIR, f'final_model{"_tuned" if
PERFORM_HYPERPARAMETER_TUNING else ""}.joblib')
FINAL_THRESHOLD_PATH = os.path.join(OUTPUT_DIR, 'final_threshold.json')

LOW_VARIANCE_THRESHOLD = 0.01; SKEWNESS_THRESHOLD = 1.0;
FEATURE_IMPORTANCE_THRESHOLD = 0.01
TEST_SIZE = 0.20; RANDOM_STATE = 42; SMOTE_RANDOM_STATE = 42
SMOTE_K_NEIGHBORS_CONFIG = 5; MLP_MAX_ITER = 300; FINAL_CHOSEN_THRESHOLD = 0.35

import shutil
CLEANUP_STRATEGY = "DELETE_ENTIRE_DIR"
def clear_output_directory(directory_path):
    if os.path.exists(directory_path):
        try: shutil.rmtree(directory_path); print(f"Successfully deleted old
output directory: {directory_path}")
        except OSError as e: print(f"Error deleting dir {directory_path}:
{e.strerror}. Close files/delete manually."); import sys; sys.exit("Cleanup
fail.")
    else: print(f"Output directory {directory_path} does not exist. No need to
delete.")
if CLEANUP_STRATEGY == "DELETE_ENTIRE_DIR": clear_output_directory(OUTPUT_DIR)

os.makedirs(OUTPUT_DIR, exist_ok=True)
os.makedirs(CM_PLOT_DIR, exist_ok=True)
print(f"Output directory '{OUTPUT_DIR}' ensured.")
print("Configuration complete.")

# --- Helper Function Definitions ---
print("\n--- Defining Helper Functions ---")
def save_dataframe(df, path):
    try:
        parent_dir = os.path.dirname(path)
        if parent_dir and not os.path.exists(parent_dir): os.makedirs(parent_dir,
exist_ok=True)
        print(f"Saving DataFrame to: {path}")
        df.to_csv(path, index=False); print("Save complete.")
    except Exception as e: print(f"Error saving DataFrame to {path}: {e}")
```

```
def load_data(path):
    print(f"Attempting to load dataset from: {path}")
    file_extension = os.path.splitext(path)[1].lower()
    try:
        if file_extension == '.csv': df = pd.read_csv(path)
        elif file_extension in ['.xls', '.xlsx']:
            try: df = pd.read_excel(path, engine=None)
            except ImportError: raise ImportError("Need 'openpyxl' or 'xlrd'.
Try: pip install openpyxl xlrd")
            except Exception as e_ex: print(f"Excel read failed: {e_ex}");
traceback.print_exc(); raise
        else: raise ValueError(f"Unsupported file: {file_extension}")
        print(f"{file_extension.upper()} file loaded. Shape: {df.shape}")
        return df
    except FileNotFoundError: print(f"FATAL: Dataset not found: {path}"); raise
    except Exception as e: print(f"FATAL: Failed to load data: {e}");
traceback.print_exc(); raise

def explore_data(df, stage_name="Data", target_column_name=None):
    print(f"\n--- Data Exploration: {stage_name} ---")
    if df is None: print("Data is None, skipping exploration."); return
    print(f"Shape: {df.shape}"); print("\nInfo:"); df.info(verbose=True,
show_counts=True)
    print("\nNumerical Description:\n", df.describe(include=np.number))
    print("\nObject/Categorical Description:\n", df.describe(include=['object',
'category']))
    missing = df.isnull().sum(); missing_gt_zero = missing[missing > 0]
    if not missing_gt_zero.empty: print("\nMissing Values (>0):\n",
missing_gt_zero)
    else: print("\nNo missing values found.")
    if target_column_name and target_column_name in df.columns:
        print(f"\nClass Distribution for '{target_column_name}':")
        print("Counts:\n", df[target_column_name].value_counts(normalize=False))
        print("Proportions:\n",
df[target_column_name].value_counts(normalize=True))
    elif target_column_name: print(f"\nWarn: Target '{target_column_name}' not
found for class dist.")
    print(f"--- End Data Exploration: {stage_name} ---")

def remove_and_report_duplicates(df, df_description="Dataset",
keep_policy='first'):
    print(f"\n--- Duplicates in {df_description} (keeping '{keep_policy}') ---")
    initial_shape = df.shape; print(f"Shape before duplicate removal:
{initial_shape}")
```

```
    if df.duplicated().any():
        df_no_duplicates = df.drop_duplicates(keep=keep_policy)
        print(f"Removed {initial_shape[0] - df_no_duplicates.shape[0]} duplicate
row(s). Shape after: {df_no_duplicates.shape}")
        return df_no_duplicates
    else: print("No duplicate rows found."); return df.copy()

def preprocess_data(df, target_col):
    print("\n--- Preprocessing Data (Main Pipeline - Step 2) ---")
    df_processed = df.copy(); label_encoders = {}; training_medians = None
    print("Identifying categorical columns for encoding...")
    categorical_cols_to_encode = []
    for col_name in df_processed.columns:
        col_dtype = df_processed[col_name].dtype
        is_obj = is_object_dtype(col_dtype); is_cat = isinstance(col_dtype,
CategoricalDtype)
        is_cat_nature = is_obj or is_cat
        if col_name == target_col:
            if is_cat_nature: categorical_cols_to_encode.append(col_name);
print(f"  Target '{target_col}' ({col_dtype}) will be encoded.")
            else: print(f"  Target '{target_col}' is numeric ({col_dtype}),
skipping encoding.")
        elif is_cat_nature: categorical_cols_to_encode.append(col_name)
    if categorical_cols_to_encode:
        print(f"Encoding categorical columns: {categorical_cols_to_encode}")
        for col_to_enc in categorical_cols_to_encode:
            print(f"  Encoding column: '{col_to_enc}'")
            le = LabelEncoder(); series_to_enc = df_processed[col_to_enc].copy()
            if col_to_enc in ['protocol']:
                print(f"    Normalizing (lowercase, strip) strings in
'{col_to_enc}'.")
                series_to_enc =
series_to_enc.astype(str).str.lower().str.strip().replace('nan', np.nan)
            series_to_enc_str = series_to_enc.astype(str)
            series_to_enc_str.loc[series_to_enc.isnull()] = 'Missing'
            fit_series = series_to_enc_str.copy()
            if 'Missing' not in fit_series.unique():
                print(f"    INFO: Temporarily adding 'Missing' to '{col_to_enc}'
for robust encoder fitting.")
                fit_series = pd.concat([fit_series, pd.Series(['Missing'])],
ignore_index=True)
            # print(f"    DEBUG: Unique for fit in '{col_to_enc}':
{np.sort(fit_series.unique()).tolist()}")
            le.fit(fit_series)
```

```
            df_processed[col_to_enc] = le.transform(series_to_enc_str)
            label_encoders[col_to_enc] = le
            print(f"    '{col_to_enc}' encoded. Classes learned:
{list(le.classes_)}")
        print("Categorical encoding complete.")
    else: print("No categorical columns for encoding.")
    print("\nStarting numerical NaN imputation...")
    num_cols = df_processed.select_dtypes(include=np.number).columns
    num_cols_consider = num_cols.drop(target_col, errors='ignore')
    if not num_cols_consider.empty:
        print(f"Calculating training medians for: {list(num_cols_consider)}")
        training_medians = df_processed[num_cols_consider].median(skipna=True)
        nan_in_num = df_processed[num_cols_consider].isnull().sum()
        cols_impute = nan_in_num[nan_in_num > 0].index
        if not cols_impute.empty:
            print(f"Applying median imputation for: {list(cols_impute)}")
            fill_vals = training_medians.reindex(cols_impute).dropna()
            if not fill_vals.empty:
                for c_fill, med_val in fill_vals.items(): df_processed[c_fill] =
df_processed[c_fill].fillna(med_val)
                rem_nans = df_processed[cols_impute].isnull().sum()
                if rem_nans.any() and rem_nans.sum() > 0: print(f"Warn: NaNs
remain in: {rem_nans[rem_nans > 0].index.tolist()}")
                else: print("    Median imputation complete.")
            else: print("Warn: No valid medians for columns with NaNs.")
        else: print("No numerical NaNs for imputation.")
    else: print("No numerical columns (excl. target) for medians/NaNs.")
    if df_processed.isnull().sum().any(): print(f"WARNING:
{df_processed.isnull().sum().sum()} NaNs remain!");
print(df_processed.isnull().sum()[df_processed.isnull().sum() > 0])
    else: print("Preprocessing complete. No missing values detected.")
    return df_processed, label_encoders, training_medians

def analyze_correlation(df, target_col, title, save_path, plot_path=None):
    print(f"\n--- Correlation Analysis: {title} ---")
    if target_col not in df.columns: print(f"Target '{target_col}' not in
DataFrame."); return None
    try:
        num_df = df.select_dtypes(include=np.number)
        if target_col not in num_df.columns: print(f"Target '{target_col}' not
numeric."); return None
        if num_df.shape[1] < 2 : print("Not enough numeric columns for
correlation."); return None
        corr_matrix = num_df.corr();
```

```
        if target_col not in corr_matrix.columns: print(f"Corr for '{target_col}'
not computed."); return None
        corr_target =
corr_matrix[target_col].sort_values(ascending=False).drop(target_col,
errors='ignore')
        if corr_target.empty: print("No other numeric features to correlate with
target."); return None
        print(f"Top 5 correlations with '{target_col}':\n", corr_target.head())
        save_dataframe(pd.DataFrame({'Feature': corr_target.index, 'Correlation':
corr_target.values}), save_path)
        plt.figure(figsize=(12, max(7, len(corr_target) // 3 + 1)));
corr_target.plot(kind='bar', alpha=0.7)
        plt.title(title); plt.ylabel("Correlation Coefficient");
plt.xlabel("Features"); plt.xticks(rotation=90)
        plt.grid(axis='y', linestyle='--', alpha=0.7); plt.tight_layout()
        if plot_path: plt.savefig(plot_path); print(f"Correlation plot saved:
{plot_path}")
        plt.show(); print(f"--- End Correlation Analysis: {title} ---"); return
corr_target
    except Exception as e: print(f"Error during correlation '{title}': {e}");
traceback.print_exc(); return None

def engineer_features(df):
    print("\n--- Feature Engineering ---"); df_eng = df.copy()
    req = ['cpu_usage','traffic_volume','login_attempts','duration']
    for col in req:
        if col not in df_eng.columns: df_eng[col]=0
        else: df_eng[col]=pd.to_numeric(df_eng[col],errors='coerce').fillna(0)
    try:

df_eng['cpu_traffic_ratio']=df_eng['cpu_usage']/(df_eng['traffic_volume']+1e-6)

df_eng['cpu_traffic_interaction']=df_eng['cpu_usage']*df_eng['traffic_volume']

df_eng['login_duration_ratio']=df_eng['login_attempts']/(df_eng['duration']+1e-6)

df_eng['login_duration_interaction']=df_eng['login_attempts']*df_eng['duration']
        print("Engineered features created.")
    except Exception as e: print(f"Error FE: {e}"); traceback.print_exc()
    print("--- End Feature Engineering ---"); return df_eng

def select_features_low_variance(df, target_col, threshold, save_path):
    print("\n--- Low Variance Feature Selection ---"); df_sel = df.copy()
    feats_check = df_sel.select_dtypes(include=np.number)
```

```
    if target_col in feats_check.columns:
feats_check=feats_check.drop(columns=[target_col])
    if feats_check.empty: print("No num features (excl. target) for low var
check."); return df_sel
    try:
        variances = feats_check.var(ddof=0).dropna()
        low_var = variances[variances < threshold].index.tolist()
        if low_var:
            print(f"Low var features (<{threshold}): {low_var}")

save_dataframe(pd.DataFrame({'Removed_Low_Var_Feature':low_var}),save_path)
            df_sel=df_sel.drop(columns=low_var); print(f"Removed {len(low_var)}
low var features.")
        else: print("No low variance features found.")
    except Exception as e: print(f"Error low var select: {e}");
traceback.print_exc(); return df.copy()
    print("--- End Low Variance Selection ---"); return df_sel

def transform_skewed_features(df, target_col, skew_threshold):
    print("\n--- Transforming Skewed Features ---"); df_trans = df.copy();
transformed = []
    cols_transform =
df_trans.select_dtypes(include=np.number).columns.drop(target_col,errors='ignore'
)
    for col in cols_transform:
        try:
            if df_trans[col].nunique(dropna=False)<=1: continue
            skew = df_trans[col].skew()
            if pd.isna(skew) or abs(skew)<=skew_threshold: continue
            min_v = df_trans[col].min()
            if min_v>=0: df_trans[col]=np.log1p(df_trans[col]); print(f"Log1p:
'{col}' (Skew:{skew:.2f})")
            else:
                shifted = df_trans[col]-min_v
                if shifted.nunique(dropna=False)>1:
df_trans[col]=np.log1p(shifted); print(f"Log1p (shifted by {min_v:.2f}): '{col}'
(Skew:{skew:.2f})")
                else: print(f"Skipped '{col}' (shifted): no variance.")
            transformed.append(col)
        except Exception as e: print(f"Warn transform '{col}': {e}");
traceback.print_exc()
    if not transformed: print("No features transformed for skewness.")
    else: print(f"Skew transform applied to: {transformed}")
    print("--- End Skew Transformation ---"); return df_trans
```

```
def calculate_feature_importance(X,y,method,save_path,plot_path=None):
    print(f"\n--- Feature Importance: {method} ---");
X_num=X.select_dtypes(include=np.number)
    if X_num.empty: print("No numeric X for importance."); return None
    if X_num.shape[1]<X.shape[1]: print(f"Warn: Non-num cols excluded for
{method} imp.")
    if y.shape[0]!=X_num.shape[0]: print(f"Warn: X_num/y length mismatch!");
return None
    scores=None
    try:
        if method=='MI':
            if X_num.shape[0]<2 or len(np.unique(y))<2: print("Skip MI:
Insufficient samples/classes."); return None

scores=pd.Series(mutual_info_classif(X_num,y,random_state=RANDOM_STATE),index=X_n
um.columns).sort_values(ascending=False)
            col,title,color='MI_Score','Importance (MI)','skyblue'
        elif method=='RF':

m=RandomForestClassifier(random_state=RANDOM_STATE,n_estimators=100,class_weight=
'balanced').fit(X_num,y)

scores=pd.Series(m.feature_importances_,index=X_num.columns).sort_values(ascendin
g=False)
            col,title,color='RF_Importance','Importance (RF)','lightgreen'
        elif method=='GB':

m=GradientBoostingClassifier(random_state=RANDOM_STATE,n_estimators=100).fit(X_nu
m,y)

scores=pd.Series(m.feature_importances_,index=X_num.columns).sort_values(ascendin
g=False)
            col,title,color='GB_Importance','Importance (GB)','salmon'
        else: print(f"Unknown imp method: {method}"); return None
        if scores is None or scores.empty: print(f"No scores for {method}.");
return None
        print(f"Top 5 {method} importances:\n",scores.head())

save_dataframe(pd.DataFrame({'Feature':scores.index,col:scores.values}),save_path
)
        plt.figure(figsize=(10,max(6,len(scores)//4+2)));
scores.plot(kind='bar',color=color,alpha=0.8)
```

```
        plt.title(title); plt.ylabel('Score'); plt.xlabel('Features');
plt.xticks(rotation=90); plt.tight_layout()
        if plot_path: plt.savefig(plot_path); print(f"Plot saved: {plot_path}")
        plt.show(); print(f"--- End Feature Importance: {method} ---"); return
scores
    except Exception as e: print(f"Error calc {method} imp: {e}");
traceback.print_exc(); return None

def select_features_from_importance(X_in,rf_scores,gb_scores,imp_thresh):
    print("\n--- Importance-Based Feature Selection ---")
    if (rf_scores is None or rf_scores.empty) and (gb_scores is None or
gb_scores.empty):
        print("Warn: RF&GB scores missing/empty. Using all numeric X_in
features.")
        num_cols_fb=X_in.select_dtypes(include=np.number).columns.tolist();
return num_cols_fb if num_cols_fb else []
    sel_rf=set(rf_scores[rf_scores>imp_thresh].index) if rf_scores is not None
and not rf_scores.empty else set()
    sel_gb=set(gb_scores[gb_scores>imp_thresh].index) if gb_scores is not None
and not gb_scores.empty else set()
    final_sel=sorted(list(sel_rf.union(sel_gb)))
    if not final_sel:
        print(f"Warn: No features > thresh {imp_thresh}. Using all numeric X_in
features.")
        final_sel=X_in.select_dtypes(include=np.number).columns.tolist()
        if not final_sel: print("Warn: X_in no numeric features for fallback.");
return []
    else: print(f"Selected {len(final_sel)} features by importance >
{imp_thresh}.")
    print("Final selected features by importance:",final_sel); print("--- End
Imp-Based Selection ---"); return final_sel

def scale_and_resample(X_features, y_target, scaler_output_path,
k_neighbors_smote=5):
    print("\n--- Scaling and Resampling ---")
    if X_features.empty: raise ValueError("X_features is empty before scaling.")
    scaler_obj = StandardScaler()
    X_numeric_features = X_features.select_dtypes(include=np.number)
    if X_numeric_features.empty: raise ValueError("No numeric features in
X_features to scale.")
    if X_numeric_features.shape[1] < X_features.shape[1]: print(f"Warn: Non-
numeric cols ignored by scaling:
{list(X_features.select_dtypes(exclude=np.number).columns)}")
    try:
```

```
        X_scaled_array = scaler_obj.fit_transform(X_numeric_features)
        X_scaled_dataframe = pd.DataFrame(X_scaled_array,
index=X_numeric_features.index, columns=X_numeric_features.columns)
        print("Scaling complete.")
    except Exception as e_scale: print(f"Error scaling: {e_scale}");
traceback.print_exc(); raise RuntimeError(f"Scaling failed: {e_scale}")
    try: joblib.dump(scaler_obj, scaler_output_path); print(f"Scaler saved:
{scaler_output_path}")
    except Exception as e_save_scaler: print(f"Error saving scaler:
{e_save_scaler}"); traceback.print_exc()
    y_aligned_for_smote = y_target.loc[X_numeric_features.index] if not
X_numeric_features.index.equals(y_target.index) else y_target.copy()
    print(f"\nClass dist before SMOTE ({len(y_aligned_for_smote)}
samples):\nProp:\n{y_aligned_for_smote.value_counts(normalize=True)}\nCounts:\n{y
_aligned_for_smote.value_counts(normalize=False)}")
    n_minority_samples = y_aligned_for_smote.value_counts().min()
    num_unique_classes = len(y_aligned_for_smote.unique())
    X_final_resampled_np = X_scaled_dataframe.values; y_final_resampled_np =
y_aligned_for_smote.values
    if num_unique_classes < 2: print(f"Skip SMOTE: Only {num_unique_classes}
class.")
    elif n_minority_samples <= k_neighbors_smote:
        print(f"INFO: Minority ({n_minority_samples}) <= k_neighbors
({k_neighbors_smote}).")
        if n_minority_samples > 1 and (n_minority_samples - 1) > 0:
            adj_k_val = n_minority_samples - 1
            print(f"Attempting SMOTE with adj k = {adj_k_val}")
            try:
                smote_adj_instance = SMOTE(random_state=SMOTE_RANDOM_STATE,
k_neighbors=adj_k_val)
                X_final_resampled_np, y_final_resampled_np =
smote_adj_instance.fit_resample(X_scaled_dataframe.values,
y_aligned_for_smote.values)
                # print(f"DEBUG (adj k={adj_k_val}): SMOTE output shapes:
X={X_final_resampled_np.shape}, y={y_final_resampled_np.shape}")
                print(f"SMOTE (adj k) done. New
dist:\nProp:\n{pd.Series(y_final_resampled_np).value_counts(normalize=True)}\nCou
nts:\n{pd.Series(y_final_resampled_np).value_counts(normalize=False)}")
            except Exception as e_adj: print(f"Error SMOTE (adj k): {e_adj}");
traceback.print_exc()
        else: print(f"Minority ({n_minority_samples}) too small for SMOTE. No
oversampling.")
    else:
        print(f"Attempting SMOTE with k_neighbors={k_neighbors_smote}...")
```

```
        try:
            smote_main_instance = SMOTE(random_state=SMOTE_RANDOM_STATE,
k_neighbors=k_neighbors_smote)
            X_final_resampled_np, y_final_resampled_np =
smote_main_instance.fit_resample(X_scaled_dataframe.values,
y_aligned_for_smote.values)
            # print(f"DEBUG (k={k_neighbors_smote}): SMOTE output shapes:
X={X_final_resampled_np.shape}, y={y_final_resampled_np.shape}")
            print(f"SMOTE done. New
dist:\nProp:\n{pd.Series(y_final_resampled_np).value_counts(normalize=True)}\nCou
nts:\n{pd.Series(y_final_resampled_np).value_counts(normalize=False)}")
        except Exception as e_smote_main: print(f"Error SMOTE
(k={k_neighbors_smote}): {e_smote_main}"); traceback.print_exc()
    print(f"\nShapes after scale/resample: X={X_final_resampled_np.shape},
y={y_final_resampled_np.shape}")
    print("--- End Scale/Resample ---"); return X_final_resampled_np,
y_final_resampled_np, scaler_obj

def _evaluate_and_log_model(model_obj, model_key_short, model_name_disp, X_test,
y_test, params_str, time_val, perf_list_ref, gscv_scoring_metric,
cm_plots_output_dir_param, is_tuned_ver):
    version_str = 'Tuned' if is_tuned_ver else 'Default'
    print(f"  Evaluating {version_str} version of {model_name_disp} on test
set...")

    if X_test.shape[0] == 0:
        print(f"  Warn: Test data empty for {model_name_disp}. Skip eval.")
        metric_entry = {'Model_Key': model_key_short, 'Model_Display':
model_name_disp, 'Tuned_Params': params_str, 'Version': version_str, 'Error':
"Test data empty", **{k: np.nan for k in ['Accuracy', 'Precision', 'Recall', 'F1
Score', 'ROC AUC', gscv_scoring_metric, 'Time(s)', 'TN', 'FP', 'FN', 'TP']}}
        perf_list_ref.append(metric_entry)
        return

    try:
        y_p_eval = model_obj.predict(X_test)
        y_pr_eval = model_obj.predict_proba(X_test)[:, 1] if hasattr(model_obj,
'predict_proba') else None
    except Exception as e_pr:
        print(f"  ERR Pred {model_name_disp} ({version_str}): {e_pr}")
        traceback.print_exc()
        perf_list_ref.append({'Model_Key': model_key_short, 'Model_Display':
model_name_disp, 'Tuned_Params': params_str, 'Version': version_str, 'Error':
f"Pred Err:{e_pr}"})
```

```
        return

    # --- Generate and Print Classification Report ---
    print(f"\n--- Classification Report for {model_name_disp} ({version_str}) ---
")
    try:

        report_str = classification_report(
            y_test,
            y_p_eval,
            target_names=['Not Malicious (0)', 'Malicious (1)']
        )
        print(report_str)
    except Exception as e_report:
        print(f"Could not generate classification report: {e_report}")
    print("------------------------------------------------------------\n")


    # Calculate individual metrics, positive class
    acc_e = accuracy_score(y_test, y_p_eval)
    prc_e = precision_score(y_test, y_p_eval, zero_division=0)
    rec_e = recall_score(y_test, y_p_eval, zero_division=0)
    f1_e = f1_score(y_test, y_p_eval, zero_division=0)
    auc_e = roc_auc_score(y_test, y_pr_eval) if y_pr_eval is not None and
len(np.unique(y_test)) > 1 else None
    auc_s_e = f"{auc_e:.4f}" if auc_e is not None else "N/A"

    print(f"  Overall Metrics ({version_str}): Acc={acc_e:.4f}, Prc={prc_e:.4f},
Rec={rec_e:.4f}, F1={f1_e:.4f}, AUC={auc_s_e}")

    # Confusion Matrix calculation and plotting
    tn_e, fp_e, fn_e, tp_e = None, None, None, None
    cm_e_plot = np.array([[0, 0], [0, 0]])
    try:
        cm_e = confusion_matrix(y_test, y_p_eval, labels=[0, 1])
        if cm_e.shape == (2, 2):
            tn_e, fp_e, fn_e, tp_e = cm_e.ravel()
            cm_e_plot = cm_e
        else:
            print(f"    Warn: CM for {model_name_disp} ({version_str}) not 2x2
(shape: {cm_e.shape}). Manual calc.")
            tp_e = np.sum((y_test == 1) & (y_p_eval == 1))
            tn_e = np.sum((y_test == 0) & (y_p_eval == 0))
            fp_e = np.sum((y_test == 0) & (y_p_eval == 1))
```

```
            fn_e = np.sum((y_test == 1) & (y_p_eval == 0))
            cm_e_plot = np.array([[tn_e, fp_e], [fn_e, tp_e]])
    except Exception as e_cm_calc:
        print(f"    Error CM calc for {model_name_disp} ({version_str}):
{e_cm_calc}")

    print(f"    CM ({version_str}): TN={tn_e}, FP={fp_e}, FN={fn_e}, TP={tp_e}")
    plt.figure(figsize=(6, 4))
    if isinstance(cm_e_plot, np.ndarray) and cm_e_plot.shape == (2, 2):
        sns.heatmap(cm_e_plot, annot=True, fmt='d', cmap='Blues',
xticklabels=['NM', 'M'], yticklabels=['NM', 'M'])
    else:
        annot_data = [[str(tn_e) if tn_e is not None else 'N/A', str(fp_e) if
fp_e is not None else 'N/A'],
                      [str(fn_e) if fn_e is not None else 'N/A', str(tp_e) if
tp_e is not None else 'N/A']]
        sns.heatmap(np.array([[0, 0], [0, 0]]), annot=annot_data, fmt='s',
cmap='Blues', xticklabels=['NM', 'M'], yticklabels=['NM', 'M'])
        print(f"    Plotting approximated CM for {model_name_disp}
({version_str}).")

    plt.title(f'CM: {model_name_disp} ({version_str})')
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.tight_layout()
    cm_path_s = os.path.join(cm_plots_output_dir_param,
f'cm_{model_key_short.replace(" ", "_").lower()}_{version_str.lower()}.png')
    try:
        plt.savefig(cm_path_s)
        print(f"    CM plot saved: {cm_path_s}")
    except Exception as e_plt:
        print(f"    Err saving CM plot: {e_plt}")
        traceback.print_exc()
    plt.show()

    # Logging metrics to the performance list
    metric_entry = {
        'Model_Key': model_key_short, 'Model_Display': model_name_disp,
'Tuned_Params': params_str, 'Version': version_str,
        'Accuracy': acc_e, 'Precision': prc_e, 'Recall': rec_e, 'F1 Score': f1_e,
'ROC AUC': auc_e, 'Time(s)': time_val,
        'TN': tn_e, 'FP': fp_e, 'FN': fn_e, 'TP': tp_e, 'Error': None
    }
    score_val_for_df = None
```

```
    if gscv_scoring_metric == 'f1': score_val_for_df = f1_e
    elif gscv_scoring_metric == 'roc_auc': score_val_for_df = auc_e
    elif gscv_scoring_metric == 'accuracy': score_val_for_df = acc_e
    elif gscv_scoring_metric == 'precision': score_val_for_df = prc_e
    elif gscv_scoring_metric == 'recall': score_val_for_df = rec_e

    if score_val_for_df is not None:
        metric_entry[gscv_scoring_metric] = score_val_for_df
    else:
        metric_entry[gscv_scoring_metric] = f1_e if f1_e is not None else np.nan
        print(f"Warn: Score for '{gscv_scoring_metric}' not mapped for
{model_name_disp}, using F1 for df col '{gscv_scoring_metric}'.")

    perf_list_ref.append(metric_entry)

def train_evaluate_models(X_train_data, y_train_data, X_test_data, y_test_data,
cm_plots_output_dir,
                          perform_tuning=True, tuning_cv_folds=3,
tuning_scoring='f1'):
    print(f"\n--- Training & Evaluating Models (Tuning: {perform_tuning}, GSCV
Score: '{tuning_scoring}') ---")
    if not all(isinstance(a,np.ndarray) for a in
[X_train_data,y_train_data,X_test_data,y_test_data]):
X_train_data,y_train_data,X_test_data,y_test_data=map(np.array,[X_train_data,y_tr
ain_data,X_test_data,y_test_data])
    if X_train_data.shape[0]==0: print("ERR: Train data empty."); return
pd.DataFrame(),{}


models_config={'RF':{'estimator':RandomForestClassifier(random_state=RANDOM_STATE
,class_weight='balanced_subsample'),'params':{'n_estimators':[50,100],'max_depth'
:[None,10,20],'min_samples_split':[2,5],'min_samples_leaf':[1,2]} if
perform_tuning else None,'display_name':'Random Forest'},

'SVM':{'estimator':SVC(probability=True,random_state=RANDOM_STATE,class_weight='b
alanced'),'params':{'C':[0.1,1],'gamma':['scale',.01],'kernel':['rbf']} if
perform_tuning else None,'display_name':'Support Vector Machine'},

'LogReg':{'estimator':LogisticRegression(random_state=RANDOM_STATE,max_iter=1000,
solver='liblinear',class_weight='balanced'),'params':{'C':[.1,1,10],'penalty':['l
1','l2']} if perform_tuning else None,'display_name':'Logistic Regression'},

'KNN':{'estimator':KNeighborsClassifier(),'params':{'n_neighbors':[3,5,7],'weight
```

```
s':['uniform','distance'],'metric':['minkowski','manhattan']} if perform_tuning
else None,'display_name':'K-Nearest Neighbors'},

'DTree':{'estimator':DecisionTreeClassifier(random_state=RANDOM_STATE,class_weigh
t='balanced'),'params':{'max_depth':[None,5,10,15],'min_samples_split':[2,5,10],'
min_samples_leaf':[1,2,5],'criterion':['gini','entropy']} if perform_tuning else
None,'display_name':'Decision Tree'},

'NB':{'estimator':GaussianNB(),'params':None,'display_name':'Gaussian Naive
Bayes'},

'MLP':{'estimator':MLPClassifier(max_iter=MLP_MAX_ITER,random_state=RANDOM_STATE,
early_stopping=True,n_iter_no_change=10,learning_rate_init=.001),'params':{'hidde
n_layer_sizes':[(50,),(100,)],'activation':['relu','tanh'],'alpha':[.0001,.001]}
if perform_tuning else None,'display_name':'Multi-layer Perceptron'}}

    p_list=[]; best_est_dict={}
    for mk,cfg in models_config.items():
        mdn=cfg['display_name']; print(f"\n----- Eval: {mdn} (Key:{mk}) -----");
base_e=cfg['estimator']; p_grid=cfg['params']
        tuned_p_str="N/A (Default/Fixed)"; cur_best_e_for_report=None;
        print(f"Fitting {mdn} with default params..."); s_def_fit=time.time()
        default_model_instance = base_e.__class__(**base_e.get_params())
        try:
            default_model_instance.fit(X_train_data,y_train_data)
            def_fit_t=time.time()-s_def_fit; print(f"Default fit time:
{def_fit_t:.2f}s.")

_evaluate_and_log_model(default_model_instance,mk,mdn,X_test_data,y_test_data,"N/
A (Default/Fixed)",def_fit_t,p_list,tuning_scoring,cm_plots_output_dir,False)
            cur_best_e_for_report = default_model_instance
        except Exception as e_d_fit: print(f"ERR Fit Default {mdn}:{e_d_fit}");
traceback.print_exc();
p_list.append({'Model_Key':mk,'Model_Display':mdn,'Tuned_Params':"N/A
(Default/Fixed)",'Version':'Default','Error':f"Fit Err:{e_d_fit}"});

        if perform_tuning and p_grid:
            print(f"GridSearchCV for {mdn} (opt
'{tuning_scoring}',CV={tuning_cv_folds})...");s_tune_t=time.time()

n_s,u_c,c_c_min=X_train_data.shape[0],len(np.unique(y_train_data)),np.min(np.uniq
ue(y_train_data,return_counts=True)[1]) if
len(np.unique(y_train_data,return_counts=True)[1])>0 else 0
```

```
            if n_s>0 and tuning_cv_folds>0 and tuning_cv_folds<=n_s: print(f"
GSCV on {n_s} samples. Each of {tuning_cv_folds} folds: ~{n_s-
(n_s/tuning_cv_folds):.0f} train, ~{n_s/tuning_cv_folds:.0f} val.")
            if n_s<tuning_cv_folds or u_c<2 or c_c_min<tuning_cv_folds:
print(f"Skip GSCV {mdn}: Insufficient data/classes for {tuning_cv_folds}-fold CV
(Samples:{n_s}, Unique Classes:{u_c}, Min Class Samples:{c_c_min}).")
            else:
                gscv_e_instance=base_e.__class__(**base_e.get_params())

gs=GridSearchCV(gscv_e_instance,p_grid,cv=tuning_cv_folds,scoring=tuning_scoring,
verbose=1,n_jobs=-1,error_score='raise')
                try:
                    gs.fit(X_train_data,y_train_data)
                    tune_t=time.time()-s_tune_t
                    cur_best_e_for_report=gs.best_estimator_;
tuned_p_str=str(gs.best_params_)
                    print(f"Best params {mdn}:{tuned_p_str}"); print(f"Tune
time:{tune_t:.2f}s.")

_evaluate_and_log_model(cur_best_e_for_report,mk,mdn,X_test_data,y_test_data,tune
d_p_str,tune_t,p_list,tuning_scoring,cm_plots_output_dir,True)
                except Exception as e_gs_fit: print(f"ERR GSCV
{mdn}:{e_gs_fit}"); traceback.print_exc();
p_list.append({'Model_Key':mk,'Model_Display':mdn,'Tuned_Params':'Tuning
Failed','Version':'Tuned','Error':f"GSCV Err:{e_gs_fit}"})

        best_est_dict[mk]=cur_best_e_for_report if cur_best_e_for_report is not
None else base_e

    performance_df=pd.DataFrame(p_list)
    if 'INITIAL_PERFORMANCE_METRICS_PATH' in globals() and 'save_dataframe' in
globals() and callable(save_dataframe):
        save_dataframe(performance_df,INITIAL_PERFORMANCE_METRICS_PATH)
    else: print("Warn: INITIAL_PERFORMANCE_METRICS_PATH or save_dataframe not
defined. Metrics DF not saved.")
    print(f"--- Model Eval Complete (Tuning:{perform_tuning}) ---"); return
performance_df,best_est_dict

print("Helper functions defined.")


print("\n--- Starting Main Pipeline Execution ---")
# Step 1: Load, Clean, Explore
print("\n--- Step 1: Load, Clean (Duplicates), Explore ---")
```

```
raw_data = None; data_for_pipeline = None
try:
    raw_data = load_data(DATASET_PATH)
    if raw_data is None: raise ValueError("Data loading failed.")
    explore_data(raw_data, "Raw Loaded Data", TARGET_COLUMN)
    data_no_duplicates = remove_and_report_duplicates(raw_data, "Loaded Dataset")
    save_dataframe(data_no_duplicates, DUPLICATE_REMOVED_DATA_PATH)
    explore_data(data_no_duplicates, "Data After Duplicate Removal",
TARGET_COLUMN)
    data_for_pipeline = data_no_duplicates.copy()
    print(f"--- End Step 1 (data_for_pipeline shape: {data_for_pipeline.shape}) -
--")
except Exception as e_step1: print(f"CRITICAL ERROR in Step 1: {e_step1}");
traceback.print_exc(); raise

# Step 2: Preprocess
print("\n--- Step 2: Preprocess ---")
data_processed, label_encoders_dict, training_medians_series = None, None, None
try:
    data_processed, label_encoders_dict, training_medians_series =
preprocess_data(data_for_pipeline, TARGET_COLUMN)
    if TARGET_COLUMN not in data_processed.columns: raise KeyError(f"Target
column '{TARGET_COLUMN}' lost after preprocessing!")
    if label_encoders_dict: joblib.dump(label_encoders_dict,
LABEL_ENCODERS_PATH); print(f"Label encoders saved to {LABEL_ENCODERS_PATH}.")
    if training_medians_series is not None and not training_medians_series.empty:
joblib.dump(training_medians_series, TRAINING_MEDIANS_PATH); print(f"Training
medians saved to {TRAINING_MEDIANS_PATH}.")
    save_dataframe(data_processed, PREPROCESSED_DATA_PATH)
    print("Preprocessed Data Head:\n"); display(data_processed.head())
    print(f"--- End Step 2 (data_processed shape: {data_processed.shape}) ---")
except Exception as e_step2: print(f"CRITICAL ERROR in Step 2 (Preprocessing):
{e_step2}"); traceback.print_exc(); raise

# Step 3: Correlation on Preprocessed
analyze_correlation(data_processed, TARGET_COLUMN, "Correlations on Preprocessed
Data", ORIGINAL_CORR_PATH, ORIGINAL_CORR_PLOT_PATH)

# Step 4: Feature Engineering
data_engineered = engineer_features(data_processed)
save_dataframe(data_engineered, FEATURE_ENGINEERED_DATA_PATH)

# Step 5: Correlation After FE
```

```
analyze_correlation(data_engineered, TARGET_COLUMN, "Correlations After Feature
Engineering", FEATURE_ENG_CORR_PATH, FEATURE_ENG_CORR_PLOT_PATH)

# Step 6: Low Variance Selection
data_low_var_removed = select_features_low_variance(data_engineered,
TARGET_COLUMN, LOW_VARIANCE_THRESHOLD, LOW_VARIANCE_FEATURES_PATH)

# Step 7: Skew Transformation
data_skew_transformed = transform_skewed_features(data_low_var_removed,
TARGET_COLUMN, SKEWNESS_THRESHOLD)

# Step 8: X/y Split, Numeric X
print("\n--- Step 8: Separate Features/Target, Ensure Numeric X ---")
data_current_for_split = data_skew_transformed.copy()
if TARGET_COLUMN not in data_current_for_split.columns: raise KeyError(f"Target
column '{TARGET_COLUMN}' is missing before X/y split!")
X_all_features_pre_selection =
data_current_for_split.drop(columns=[TARGET_COLUMN])
y_all_labels_pre_selection = data_current_for_split[TARGET_COLUMN]
X_numeric_for_importance =
X_all_features_pre_selection.select_dtypes(include=np.number)
if X_numeric_for_importance.empty: raise ValueError("No numeric features
available after all preprocessing steps.")
print(f"X numeric features selected. Original columns:
{len(X_all_features_pre_selection.columns)}, Numeric columns:
{len(X_numeric_for_importance.columns)}")
y_aligned_for_importance =
y_all_labels_pre_selection.loc[X_numeric_for_importance.index] if not
X_numeric_for_importance.index.equals(y_all_labels_pre_selection.index) else
y_all_labels_pre_selection.copy()
data_numeric_to_save = X_numeric_for_importance.copy();
data_numeric_to_save[TARGET_COLUMN] = y_aligned_for_importance
save_dataframe(data_numeric_to_save, SELECTED_FEATURES_DATA_PATH)
print(f"Shapes of data for importance calculation:
X_numeric={X_numeric_for_importance.shape},
y_aligned={y_aligned_for_importance.shape}")

# Step 9: Correlation on Selected Numeric
analyze_correlation(data_numeric_to_save, TARGET_COLUMN, "Correlations on
Selected Numeric Features (Pre-Importance)", SELECTED_CORR_PATH,
SELECTED_CORR_PLOT_PATH)

# Step 10-12: Feature Importance
```

```
mi_importance_scores = calculate_feature_importance(X_numeric_for_importance,
y_aligned_for_importance, 'MI', MUTUAL_INFO_PATH, MUTUAL_INFO_PLOT_PATH)
rf_importance_scores = calculate_feature_importance(X_numeric_for_importance,
y_aligned_for_importance, 'RF', RF_IMPORTANCE_PATH, RF_IMPORTANCE_PLOT_PATH)
gb_importance_scores = calculate_feature_importance(X_numeric_for_importance,
y_aligned_for_importance, 'GB', GB_IMPORTANCE_PATH, GB_IMPORTANCE_PLOT_PATH)

# Step 13: Final Feature Selection from Importance Scores
final_selected_feature_names =
select_features_from_importance(X_numeric_for_importance, rf_importance_scores,
gb_importance_scores, FEATURE_IMPORTANCE_THRESHOLD)
if not final_selected_feature_names:
    print("CRITICAL: No features selected by importance. Using all numeric as
fallback.")
    final_selected_feature_names = X_numeric_for_importance.columns.tolist()
    if not final_selected_feature_names: raise ValueError("No numeric features
for fallback.")
X_final_features_for_model =
X_numeric_for_importance[final_selected_feature_names] if
final_selected_feature_names else
pd.DataFrame(index=X_numeric_for_importance.index)
y_final_labels_for_model =
y_all_labels_pre_selection.loc[X_final_features_for_model.index]
if X_final_features_for_model.empty : raise
ValueError("X_final_features_for_model empty. Cannot proceed.")
print(f"\nShapes after final selection: X={X_final_features_for_model.shape},
y={y_final_labels_for_model.shape}")
try:
    if rf_importance_scores is not None and gb_importance_scores is not None and
final_selected_feature_names:
        rf_final_imp =
rf_importance_scores.reindex(final_selected_feature_names).fillna(0) if not
rf_importance_scores.empty else pd.Series(0, index=final_selected_feature_names,
dtype=float)
        gb_final_imp =
gb_importance_scores.reindex(final_selected_feature_names).fillna(0) if not
gb_importance_scores.empty else pd.Series(0, index=final_selected_feature_names,
dtype=float)
        combined_df = pd.DataFrame({'Feature': final_selected_feature_names,
'RF_Importance': rf_final_imp.values, 'GB_Importance': gb_final_imp.values})
        save_dataframe(combined_df.sort_values(by=['RF_Importance',
'GB_Importance'], ascending=[False, False]), COMBINED_IMPORTANCE_PATH)
        print("Combined importance saved.")
```

```
    else: print("Skipping combined importance save (missing scores or
features).")
    joblib.dump(final_selected_feature_names, FEATURE_LIST_PATH); print(f"Final
feature list saved: {FEATURE_LIST_PATH}")
except Exception as e_save_imp: print(f"Error saving importance/list:
{e_save_imp}"); traceback.print_exc()

# Step 14: Scale & Resample
print(f"\n--- Step 14: Scale Features and Resample Data ---")
print(f"Data shapes entering scale_and_resample:
X_final={X_final_features_for_model.shape},
y_final={y_final_labels_for_model.shape}")
X_resampled_data, y_resampled_data, final_scaler_object = None, None, None
try:
    X_resampled_data, y_resampled_data, final_scaler_object = scale_and_resample(
        X_final_features_for_model, y_final_labels_for_model, SCALER_PATH,
k_neighbors_smote=SMOTE_K_NEIGHBORS_CONFIG
    )
except Exception as e_step14: print(f"CRITICAL ERROR Step 14: {e_step14}");
traceback.print_exc(); raise
print(f"\nVERIF (Main): Shapes after scale_and_resample:
X={X_resampled_data.shape}, y={y_resampled_data.shape}")
print(f"VERIF (Main): Class dist in
y_resampled_data:\nProp:\n{pd.Series(y_resampled_data).value_counts(normalize=Tru
e)}\nCounts:\n{pd.Series(y_resampled_data).value_counts(normalize=False)}")
print(f"--- End Step 14 ---")

# Step 15: Split Data
print("\n--- Step 15: Split Data ---")
X_train, X_test, y_train, y_test = None, None, None, None
try:
    if y_resampled_data is None or len(y_resampled_data) == 0: raise
ValueError("Resampled y empty for split.")
    unique_cls, counts_cls = np.unique(y_resampled_data, return_counts=True)
    min_samp = counts_cls.min() if len(counts_cls) > 0 else 0
    stratify_opt = y_resampled_data if min_samp >= 2 and len(unique_cls) > 1 else
None
    if stratify_opt is None and len(y_resampled_data) > 0 : print("Warn:
Stratification disabled for split.")
    X_train, X_test, y_train, y_test = train_test_split(
        X_resampled_data, y_resampled_data, test_size=TEST_SIZE,
random_state=RANDOM_STATE, stratify=stratify_opt
    )
```

```
    print(f"Data split. Train: {X_train.shape}, y_train: {y_train.shape} (Dist:
{np.unique(y_train, return_counts=True)})")
    print(f"Test: {X_test.shape}, y_test: {y_test.shape} (Dist:
{np.unique(y_test, return_counts=True)})")
    print(f"--- End Step 15 ---")
except ValueError as ve_s: print(f"ValueError during split: {ve_s}.");
traceback.print_exc(); raise
except Exception as e_s: print(f"UNEXPECTED ERROR during split: {e_s}");
traceback.print_exc(); raise


# Step 16: Train & Evaluate Models using Hyperparameter Tuning
print(f"\n--- Step 16: Train & Evaluate Models (Hyperparameter Tuning Enabled:
{PERFORM_HYPERPARAMETER_TUNING}) ---")
initial_performance_df, all_best_fitted_models_dict = train_evaluate_models(
    X_train, y_train, X_test, y_test,
    CM_PLOT_DIR,
    perform_tuning=PERFORM_HYPERPARAMETER_TUNING,
    tuning_cv_folds=TUNING_CV_FOLDS,
    tuning_scoring=TUNING_SCORING_METRIC
)
print("\n--- Initial Model Performance Summary (Potentially after tuning) ---")
display(initial_performance_df)

# Step 17: Final Model Choice (Corrected Filtering)
print("\n--- Step 17: Final Model Choice ---")
chosen_model_short_name_key = 'RF'
chosen_model_display_name = 'Random Forest'
chosen_model_was_actually_tuned = False
chosen_model_final_params_info = "N/A (Default/Fixed)"

if initial_performance_df is not None and not initial_performance_df.empty and
TUNING_SCORING_METRIC in initial_performance_df.columns:
    temp_df_for_selection = initial_performance_df.copy()
    if 'Error' not in temp_df_for_selection.columns:
        print(f"Warning: 'Error' column missing in performance_df, adding testing
for filtering.")
        temp_df_for_selection['Error'] = None
    valid_perf_df = temp_df_for_selection.dropna(subset=[TUNING_SCORING_METRIC])
    valid_perf_df = valid_perf_df[valid_perf_df['Error'].isnull()]
    if not valid_perf_df.empty:
        best_model_series_info = None
        if PERFORM_HYPERPARAMETER_TUNING and 'Version' in valid_perf_df.columns:
```

```
            tuned_versions_df = valid_perf_df[valid_perf_df['Version'] ==
'Tuned']
            if not tuned_versions_df.empty:
                try: best_model_series_info =
tuned_versions_df.loc[tuned_versions_df[TUNING_SCORING_METRIC].idxmax()];
print(f"Selected best 'Tuned' model based on '{TUNING_SCORING_METRIC}'.")
                except Exception as e_sel_tuned: print(f"Error selecting best
tuned model: {e_sel_tuned}. Will consider default models.");
traceback.print_exc()
            else: print(f"No successfully 'Tuned' model versions found.
Considering 'Default' versions.")
        if best_model_series_info is None:
            target_df_sel = valid_perf_df
            if 'Version' in valid_perf_df.columns:
                default_df = valid_perf_df[valid_perf_df['Version'] == 'Default']
                if not default_df.empty: target_df_sel = default_df;
print(f"Selecting best model from 'Default' versions using
'{TUNING_SCORING_METRIC}'.")
                else: print(f"No 'Default' model versions found. Selecting from
all valid entries.")
            if not target_df_sel.empty:
                try: best_model_series_info =
target_df_sel.loc[target_df_sel[TUNING_SCORING_METRIC].idxmax()]
                except Exception as e_sel_default: print(f"Error selecting best
default model: {e_sel_default}. Script defaults used."); traceback.print_exc()
            else: print(f"Warn: No valid models to select from. Script defaults
used.")
        if best_model_series_info is not None:
            try:
                chosen_model_short_name_key = best_model_series_info['Model_Key']
                chosen_model_display_name =
best_model_series_info['Model_Display']
                chosen_model_final_params_info =
best_model_series_info['Tuned_Params']
                if best_model_series_info.get('Version') == 'Tuned' and
chosen_model_final_params_info not in ["N/A (Default/Fixed)", "Tuning Failed"]:
chosen_model_was_actually_tuned = True
            except KeyError as e_key: print(f"KeyError extracting info from
best_model_series: {e_key}. Series: {best_model_series_info}");
traceback.print_exc()
    else: print(f"Warning: No valid model performances after filtering.
Defaulting to '{chosen_model_display_name}'.")
else: print(f"Warning: Perf DF empty or '{TUNING_SCORING_METRIC}' missing.
Defaulting to '{chosen_model_display_name}'.")
```

```
print(f"Final chosen model info: Name='{chosen_model_display_name}',
Key='{chosen_model_short_name_key}', Params='{chosen_model_final_params_info}',
Was Tuned='{chosen_model_was_actually_tuned}'")
print(f"--- End Step 17 ---")


# Step 18: Threshold Tuning
print(f"\n--- Step 18: Threshold Tuning for Chosen Model:
{chosen_model_display_name} ---")
final_model_object_for_tuning_and_saving = None
if chosen_model_short_name_key in all_best_fitted_models_dict and
all_best_fitted_models_dict[chosen_model_short_name_key] is not None:
    final_model_object_for_tuning_and_saving =
all_best_fitted_models_dict[chosen_model_short_name_key]
    status_msg = "tuned (best from GSCV)" if chosen_model_was_actually_tuned else
"default/fixed param"
    print(f"Using {status_msg} version of {chosen_model_display_name} for
subsequent steps.")
else:
    print(f"CRIT WARN: Chosen key '{chosen_model_short_name_key}' not in best
models dict or model is None. Re-fitting RF.")
    final_model_object_for_tuning_and_saving =
RandomForestClassifier(random_state=RANDOM_STATE,
class_weight='balanced_subsample').fit(X_train, y_train)
    chosen_model_display_name = f"Random Forest (Fallback)";
chosen_model_was_actually_tuned = False; chosen_model_final_params_info = "N/A
(Fallback)"

threshold_analysis_list = []
if final_model_object_for_tuning_and_saving is None: print(f"Error: Final model
for '{chosen_model_display_name}' is None. Skipping threshold analysis.")
elif not hasattr(final_model_object_for_tuning_and_saving, "predict_proba"):
print(f"Error: Model '{chosen_model_display_name}' no predict_proba for threshold
analysis.")
else:
    y_prob_thresh_tune =
final_model_object_for_tuning_and_saving.predict_proba(X_test)[:, 1]
    thresh_vals_eval = sorted(list(set(t for t in
np.arange(0.05,0.96,0.05).tolist()+[0.01,0.99] if 0<t<1)))
    print(f"Analyzing thresholds for {chosen_model_display_name}:
{thresh_vals_eval}")
    for current_thr in thresh_vals_eval:
        y_pred_thr_eval = (y_prob_thresh_tune >= current_thr).astype(int)
        recall_at_thr = recall_score(y_test, y_pred_thr_eval, zero_division=0)
```

```
        precision_at_thr = precision_score(y_test, y_pred_thr_eval,
zero_division=0)
        f1_at_thr = f1_score(y_test, y_pred_thr_eval, zero_division=0)
        cm_thr_eval = confusion_matrix(y_test, y_pred_thr_eval, labels=[0,1])
        tn_t, fp_t, fn_t, tp_t = cm_thr_eval.ravel() if cm_thr_eval.shape==(2,2)
else (None,)*4

threshold_analysis_list.append({'Threshold':current_thr,'Recall':recall_at_thr,'P
recision':precision_at_thr,'F1':f1_at_thr,'FN':fn_t,'FP':fp_t,'TP':tp_t,'TN':tn_t
})
    if threshold_analysis_list:
        threshold_results_df = pd.DataFrame(threshold_analysis_list);
print("Threshold Analysis Results:\n"); display(threshold_results_df)
        save_dataframe(threshold_results_df,THRESHOLD_ANALYSIS_PATH)
    else: print("No threshold results generated.")
print("--- Threshold Analysis Complete ---")


# Step 19: Final Threshold
print(f"\n--- Step 19: Final Threshold Selection (from Configuration) ---")
print(f"Final Model for deployment: '{chosen_model_display_name}'. Configured
Threshold: {FINAL_CHOSEN_THRESHOLD}")


# Step 20: Final Metrics
print(f"\n--- Step 20: Final Metrics for '{chosen_model_display_name}' @
Configured Threshold {FINAL_CHOSEN_THRESHOLD} ---")
try:
    if final_model_object_for_tuning_and_saving is None: raise ValueError("Final
model object is None.")
    if not hasattr(final_model_object_for_tuning_and_saving,"predict_proba"):
raise AttributeError("Final model no predict_proba.")
    y_probs_final_eval =
final_model_object_for_tuning_and_saving.predict_proba(X_test)[:,1]
    y_preds_final_eval = (y_probs_final_eval >=
FINAL_CHOSEN_THRESHOLD).astype(int)
    accuracy_final = accuracy_score(y_test, y_preds_final_eval)
    precision_final = precision_score(y_test, y_preds_final_eval,
zero_division=0)
    recall_final = recall_score(y_test, y_preds_final_eval, zero_division=0)
    f1_final = f1_score(y_test, y_preds_final_eval, zero_division=0)
    roc_auc_final_val = roc_auc_score(y_test,y_probs_final_eval) if
len(np.unique(y_test))>1 and y_probs_final_eval is not None else None
    cm_final_result = confusion_matrix(y_test,y_preds_final_eval,labels=[0,1])
```

```
    tn_f_res, fp_f_res, fn_f_res, tp_f_res = cm_final_result.ravel() if
cm_final_result.shape==(2,2) else (None,)*4
    roc_auc_str_final = f"{roc_auc_final_val:.5f}" if roc_auc_final_val is not
None else "N/A"
    print(f"  Final Metrics: Acc={accuracy_final:.5f}, Prc={precision_final:.5f},
Rec={recall_final:.5f}, F1={f1_final:.5f}, AUC={roc_auc_str_final}")
    if tn_f_res is not None: print(f"  Final CM: TN={tn_f_res}, FP={fp_f_res},
FN={fn_f_res}, TP={tp_f_res}")
    else: print(f"  Final CM undetermined (Shape: {cm_final_result.shape}).")
    final_metrics_dict = {'Model':chosen_model_display_name,
'Tuned_Params_Used':chosen_model_final_params_info,
                          'Threshold':FINAL_CHOSEN_THRESHOLD,
'Accuracy':accuracy_final, 'Precision':precision_final,
                          'Recall':recall_final, 'F1 Score':f1_final, 'ROC
AUC':roc_auc_final_val,

'TN':tn_f_res,'FP':fp_f_res,'FN':fn_f_res,'TP':tp_f_res}
    save_dataframe(pd.DataFrame([final_metrics_dict]),FINAL_METRICS_PATH)
except Exception as e_fm: print(f"Error final metrics for
{chosen_model_display_name}: {e_fm}"); traceback.print_exc()
print("--- Final Metric Calc Complete ---")

# Step 21: Save Final Artifacts
print("\n--- Step 21: Saving Final Model and Artifacts ---")
print(f"Saving final model ({chosen_model_display_name}) to: {FINAL_MODEL_PATH}")
print(f"Saving threshold ({FINAL_CHOSEN_THRESHOLD}) to: {FINAL_THRESHOLD_PATH}")
try:
    if final_model_object_for_tuning_and_saving:
joblib.dump(final_model_object_for_tuning_and_saving, FINAL_MODEL_PATH);
print("Final model saved.")
    else: print("Error: Final model object is None. Not saved.")
    with open(FINAL_THRESHOLD_PATH,'w') as f: json.dump({'threshold':
float(FINAL_CHOSEN_THRESHOLD)}, f); print("Final threshold saved.")
    final_model_choice_data = {
        "chosen_model_short_name": chosen_model_short_name_key,
"chosen_model_display_name": chosen_model_display_name,
        "pipeline_attempted_tuning": PERFORM_HYPERPARAMETER_TUNING,
        "chosen_model_was_actually_tuned": chosen_model_was_actually_tuned,
        "tuned_parameters_string": chosen_model_final_params_info
    }
    with open(MODEL_CHOICE_INFO_PATH, 'w') as f_mci:
json.dump(final_model_choice_data, f_mci, indent=4); print(f"Model choice info
saved: {MODEL_CHOICE_INFO_PATH}")
    print(f"\nAssociated artifacts in {OUTPUT_DIR}:")
```

```
    for p_name, p_val in zip(['Model', 'ModelChoiceInfo', 'Scaler',
'FeatureList', 'ThresholdCfg', 'Encoders', 'Medians'],
                             [FINAL_MODEL_PATH, MODEL_CHOICE_INFO_PATH,
SCALER_PATH, FEATURE_LIST_PATH, FINAL_THRESHOLD_PATH, LABEL_ENCODERS_PATH,
TRAINING_MEDIANS_PATH]):
        print(f" - {p_name}: {p_val} (exists: {os.path.exists(p_val)})")
except Exception as e_save_final: print(f"Error saving final artifacts:
{e_save_final}"); traceback.print_exc()
print("--- Saving Final Artifacts Complete ---")

print("\n--- Creating and Saving SHAP Explainer ---")
import shap
SHAP_EXPLAINER_PATH = os.path.join(OUTPUT_DIR, 'shap_explainer.joblib')
try:
    if final_model_object_for_tuning_and_saving and X_train is not None:
        # TreeExplainer is efficient
        explainer = shap.TreeExplainer(final_model_object_for_tuning_and_saving,
X_train)
        joblib.dump(explainer, SHAP_EXPLAINER_PATH)
        print(f"SHAP explainer saved successfully to: {SHAP_EXPLAINER_PATH}")
    else:
        print("Skipping SHAP explainer creation: final model or training data not
available.")
except Exception as e_shap:
    print(f"Error creating or saving SHAP explainer: {e_shap}")
    traceback.print_exc()

# Step 22: Define & Test Local Prediction Function
print("\n--- Step 22: Define & Test Local Prediction Function ---")
loaded_model_for_test = None; loaded_scaler_for_test = None;
loaded_feature_list_for_test = None
loaded_encoders_for_test = {}; loaded_medians_for_test = None
loaded_threshold_for_test = FINAL_CHOSEN_THRESHOLD
all_artifacts_loaded_ok_for_step22 = True
try:
    if os.path.exists(FINAL_MODEL_PATH): loaded_model_for_test =
joblib.load(FINAL_MODEL_PATH); print(f"...Model loaded: {FINAL_MODEL_PATH}")
    else: all_artifacts_loaded_ok_for_step22 = False; print(f"...Model NOT FOUND:
{FINAL_MODEL_PATH}")
    if os.path.exists(SCALER_PATH): loaded_scaler_for_test =
joblib.load(SCALER_PATH); print(f"...Scaler loaded: {SCALER_PATH}")
    else: all_artifacts_loaded_ok_for_step22 = False; print(f"...Scaler NOT
FOUND: {SCALER_PATH}")
```

```
    if os.path.exists(FEATURE_LIST_PATH): loaded_feature_list_for_test =
joblib.load(FEATURE_LIST_PATH); print(f"...Feature List loaded:
{FEATURE_LIST_PATH}")
    else: all_artifacts_loaded_ok_for_step22 = False; print(f"...Feature List NOT
FOUND: {FEATURE_LIST_PATH}")
    if os.path.exists(LABEL_ENCODERS_PATH):
        temp_le = joblib.load(LABEL_ENCODERS_PATH)
        if isinstance(temp_le, dict): loaded_encoders_for_test = temp_le;
print(f"...Encoders loaded: {LABEL_ENCODERS_PATH}")
        else: print(f"...Encoders file at {LABEL_ENCODERS_PATH} not a dict.")
    else: print(f"...Encoders file NOT FOUND: {LABEL_ENCODERS_PATH}.")
    if os.path.exists(TRAINING_MEDIANS_PATH):
        temp_med = joblib.load(TRAINING_MEDIANS_PATH)
        if isinstance(temp_med, pd.Series): loaded_medians_for_test = temp_med;
print(f"...Medians loaded: {TRAINING_MEDIANS_PATH}")
        else: print(f"...Medians file at {TRAINING_MEDIANS_PATH} not a Series.")
    else: print(f"...Medians file NOT FOUND: {TRAINING_MEDIANS_PATH}.")
    if os.path.exists(FINAL_THRESHOLD_PATH):
        with open(FINAL_THRESHOLD_PATH, 'r') as f: loaded_threshold_for_test =
float(json.load(f).get('threshold', loaded_threshold_for_test))
        print(f"...Threshold loaded from file: {loaded_threshold_for_test}")
    else: print(f"...Threshold file NOT FOUND. Using default:
{loaded_threshold_for_test}")
    if not all_artifacts_loaded_ok_for_step22: print("WARNING: Not all essential
artifacts loaded for local test.")
except Exception as e: all_artifacts_loaded_ok_for_step22 = False; print(f"Error
loading artifacts for local test: {e}"); traceback.print_exc()

def local_predict_pipeline(input_df, model_to_use=None, scaler_to_use=None,
feature_list_to_use=None,
                           encoders_to_use=None, medians_to_use=None,
threshold_to_apply=None):
    fn_name = "local_predict_pipeline"
    current_threshold = threshold_to_apply if threshold_to_apply is not None else
(FINAL_CHOSEN_THRESHOLD if 'FINAL_CHOSEN_THRESHOLD' in globals() else 0.5)
    print(f"\n[{fn_name}] Processing {len(input_df)} records. Using threshold:
{current_threshold:.3f}.")
    if not isinstance(input_df, pd.DataFrame): print(f"[{fn_name}] Error: Input
not DataFrame."); return None, None
    if model_to_use is None or scaler_to_use is None or feature_list_to_use is
None: print(f"[{fn_name}] Error: Critical artifacts missing."); return None, None
    effective_encoders = encoders_to_use if encoders_to_use is not None else {}
    if input_df.empty: print(f"[{fn_name}] Input DataFrame empty."); return
np.array([]), np.array([])
```

```
    data_to_predict = input_df.copy(); processing_start_time = time.time()
    try:
        print(f"[{fn_name}] Applying preprocessing steps for local
prediction...")
        categorical_cols_predict =
data_to_predict.select_dtypes(include=['object','category']).columns
        if not categorical_cols_predict.empty:
            for col_cat_pred in categorical_cols_predict:
                if col_cat_pred in effective_encoders:
                    label_enc = effective_encoders[col_cat_pred]
                    series_to_transform_pred =
data_to_predict[col_cat_pred].copy()
                    if col_cat_pred in ['protocol']:
                        # print(f"    Normalizing strings for '{col_cat_pred}'
before transform.")
                        series_to_transform_pred =
series_to_transform_pred.astype(str).str.lower().str.strip()
                        series_to_transform_pred.replace(['nan', ''], np.nan,
inplace=True)
                    series_to_transform_pred =
series_to_transform_pred.astype(str)
                    original_nan_mask_pred =
data_to_predict[col_cat_pred].isnull() | (series_to_transform_pred.str.lower() ==
'nan')
                    series_to_transform_pred.loc[original_nan_mask_pred] =
'Missing'
                    default_encoded_value_for_unknown = 0
                    if 'Missing' in label_enc.classes_:
default_encoded_value_for_unknown = label_enc.transform(['Missing'])[0]
                    # else: print(f"    Warn: 'Missing' not in classes for
encoder '{col_cat_pred}'. Unknowns map to 0.")
                    processed_values_list = []; unknown_cat_warned = False
                    for item_idx, item_val_norm in
enumerate(series_to_transform_pred):
                        try:
processed_values_list.append(label_enc.transform([item_val_norm])[0])
                        except ValueError:

processed_values_list.append(default_encoded_value_for_unknown)
                            if not unknown_cat_warned: print(f"    [{fn_name}]
Warn: Post-norm category '{item_val_norm}' in '{col_cat_pred}' unknown. Mapped to
default ({default_encoded_value_for_unknown})."); unknown_cat_warned=True
                    data_to_predict[col_cat_pred] = processed_values_list
```

```
                    data_to_predict[col_cat_pred] =
pd.to_numeric(data_to_predict[col_cat_pred],
errors='coerce').fillna(default_encoded_value_for_unknown)
                else: data_to_predict[col_cat_pred] =
pd.to_numeric(data_to_predict[col_cat_pred], errors='coerce').fillna(0)

        for col_check_num in data_to_predict.columns:
            if data_to_predict[col_check_num].dtype == 'object':
data_to_predict[col_check_num] = pd.to_numeric(data_to_predict[col_check_num],
errors='ignore')
        numerical_cols_imp =
data_to_predict.select_dtypes(include=np.number).columns
        if medians_to_use is not None:
            for col_num_imp in numerical_cols_imp:
                if data_to_predict[col_num_imp].isnull().any():
                    if col_num_imp in medians_to_use:
data_to_predict[col_num_imp] =
data_to_predict[col_num_imp].fillna(medians_to_use[col_num_imp])
                    else: local_med = data_to_predict[col_num_imp].median();
data_to_predict[col_num_imp] = data_to_predict[col_num_imp].fillna(local_med if
pd.notna(local_med) else 0)
        elif data_to_predict[numerical_cols_imp].isnull().sum().any():
            cols_zf = data_to_predict[numerical_cols_imp].isnull().sum(); cols_zf
= cols_zf[cols_zf > 0].index
            for col_zf_l in cols_zf: data_to_predict[col_zf_l] =
data_to_predict[col_zf_l].fillna(0)

        req_fe_cols_local =
['cpu_usage','traffic_volume','login_attempts','duration']
        for col_fe in req_fe_cols_local:
            if col_fe not in data_to_predict.columns: data_to_predict[col_fe] = 0
            else: data_to_predict[col_fe] =
pd.to_numeric(data_to_predict[col_fe], errors='coerce').fillna(0)

data_to_predict['cpu_traffic_ratio']=data_to_predict['cpu_usage']/(data_to_predic
t['traffic_volume']+1e-6)

data_to_predict['cpu_traffic_interaction']=data_to_predict['cpu_usage']*data_to_p
redict['traffic_volume']

data_to_predict['login_duration_ratio']=data_to_predict['login_attempts']/(data_t
o_predict['duration']+1e-6)
```

```
data_to_predict['login_duration_interaction']=data_to_predict['login_attempts']*d
ata_to_predict['duration']

        X_for_scale = pd.DataFrame(index=data_to_predict.index)
        for f_exp in feature_list_to_use:
            if f_exp in data_to_predict.columns: X_for_scale[f_exp] =
data_to_predict[f_exp]
            else: X_for_scale[f_exp] = 0
        for col_fc in X_for_scale.columns: X_for_scale[col_fc] =
pd.to_numeric(X_for_scale[col_fc], errors='coerce')
        if X_for_scale.isnull().values.any(): X_for_scale = X_for_scale.fillna(0)
        non_num_fin_loc = X_for_scale.select_dtypes(exclude=np.number).columns
        if not non_num_fin_loc.empty: raise ValueError(f"[{fn_name}] Non-num data
in final feats: {list(non_num_fin_loc)}.")

        scaled_feats_loc = scaler_to_use.transform(X_for_scale)
        probs_loc = model_to_use.predict_proba(scaled_feats_loc)[:,1]
        preds_loc = (probs_loc >= current_threshold).astype(int)
        print(f"[{fn_name}] Local predict success ({(time.time()-
processing_start_time):.2f}s).")
        return preds_loc, probs_loc
    except Exception as e: print(f"[{fn_name}] ERR local predict pipe: {e}");
traceback.print_exc(); return None,None

if not all_artifacts_loaded_ok_for_step22:
    def local_predict_pipeline(input_df, **kwargs): print("Local predict test
value: artifacts failed."); return None,None

print("\n--- Testing Local Prediction Pipeline ---")
try:
    sample_df_for_local_test = pd.DataFrame({
        'traffic_volume':[2500, 450, np.nan, 10000, 0],
        'packet_size':   [128,  64,  32,   1500, 512],
        'protocol':      ['tcp','udp','new_protocol_test','TCP ', None],
        'duration':      [45.5, np.nan, 1.0, 300.0, 0.0],
        'login_attempts':[1,    0,   0,    5,    2],
        'cpu_usage':     [0.65,0.15,0.05,  0.90, 0.75]
    })
    print("Sample Raw Data for Local Prediction Test:\n");
display(sample_df_for_local_test)
    if all_artifacts_loaded_ok_for_step22 and callable(local_predict_pipeline):
        preds_l, probs_l = local_predict_pipeline(sample_df_for_local_test,
```

```
            model_to_use=loaded_model_for_test,
scaler_to_use=loaded_scaler_for_test,
            feature_list_to_use=loaded_feature_list_for_test,
encoders_to_use=loaded_encoders_for_test,
            medians_to_use=loaded_medians_for_test,
threshold_to_apply=loaded_threshold_for_test
        )
        if preds_l is not None: display(pd.DataFrame({'Prediction':preds_l,
'Malicious_Probability':probs_l}))
        else: print("Local prediction test failed (returned None).")
    else: print("Local prediction test skipped (artifacts not loaded or function
not callable).")
except Exception as e_loc_test: print(f"Error during local prediction test:
{e_loc_test}"); traceback.print_exc()

print("\n--- Step 22: Local Prediction Function Definition and Test Complete ---
")
print("\n--- ENTIRE PIPELINE SCRIPT COMPLETE ---")
```

Appendix B: Streamlit Interface- User Interface Code

```
import streamlit as st
import pandas as pd
import numpy as np
import joblib
import json
import os
import traceback
import altair as alt
import sqlite3
from datetime import datetime
import shap
import matplotlib.pyplot as plt

# --- Page Configuration ---
st.set_page_config(
    page_title="Cyber Threat Detector",
    layout="wide",
    initial_sidebar_state="expanded"
)

# --- Configuration & Paths ---
ARTIFACTS_DIR = 'Final_Model'
DB_PATH = 'prediction_log.db'
PERFORMED_TUNING = True

MODEL_FILENAME = f'final_model{"_tuned" if PERFORMED_TUNING else ""}.joblib'
MODEL_PATH = os.path.join(ARTIFACTS_DIR, MODEL_FILENAME)
SCALER_PATH = os.path.join(ARTIFACTS_DIR, 'scaler.pkl')
FEATURE_LIST_PATH = os.path.join(ARTIFACTS_DIR, 'selected_feature_list.pkl')
LABEL_ENCODERS_PATH = os.path.join(ARTIFACTS_DIR, 'label_encoders.joblib')
TRAINING_MEDIANS_PATH = os.path.join(ARTIFACTS_DIR, 'training_medians.joblib')
THRESHOLD_PATH = os.path.join(ARTIFACTS_DIR, 'final_threshold.json')
MODEL_CHOICE_INFO_PATH = os.path.join(ARTIFACTS_DIR,
'final_model_choice_info.json')
IMPORTANCE_PATH = os.path.join(ARTIFACTS_DIR, 'combined_feature_importance.csv')
SHAP_EXPLAINER_PATH = os.path.join(ARTIFACTS_DIR, 'shap_explainer.joblib')

# --- Database Functions ---
def init_db():
    with sqlite3.connect(DB_PATH) as conn:
        conn.execute('''
            CREATE TABLE IF NOT EXISTS predictions
```

```
            (timestamp TEXT, prediction TEXT, probability REAL, cpu_usage REAL,
traffic_volume REAL, login_attempts INTEGER)
        ''')
        conn.commit()

def log_prediction(inputs, prediction, probability):
    with sqlite3.connect(DB_PATH) as conn:
        conn.execute("INSERT INTO predictions VALUES (?,?,?,?,?,?)",
                     (datetime.now().isoformat(timespec='seconds'), prediction,
probability,
                      inputs['cpu_usage'][0], inputs['traffic_volume'][0],
inputs['login_attempts'][0]))
        conn.commit()

def get_log_data():
    with sqlite3.connect(DB_PATH) as conn:
        return pd.read_sql('SELECT * FROM predictions', conn,
parse_dates=['timestamp'])

# --- Core Application Functions ---
@st.cache_resource
def load_artifacts():
    required_paths = {
        'Model': MODEL_PATH, 'Scaler': SCALER_PATH, 'Feature List':
FEATURE_LIST_PATH,
        'Encoders': LABEL_ENCODERS_PATH, 'Medians': TRAINING_MEDIANS_PATH,
        'Threshold': THRESHOLD_PATH, 'Model Info': MODEL_CHOICE_INFO_PATH,
        'Importance Data': IMPORTANCE_PATH, 'SHAP Explainer': SHAP_EXPLAINER_PATH
    }
    missing = [name for name, path in required_paths.items() if not
os.path.exists(path)]
    if missing:
        st.error(f"Error: Missing artifact(s): {', '.join(missing)}. Please re-
run `Cyberapp.py`.")
        return None
    try:
        artifacts = {
            'model': joblib.load(MODEL_PATH), 'scaler': joblib.load(SCALER_PATH),
            'feature_list': joblib.load(FEATURE_LIST_PATH), 'label_encoders':
joblib.load(LABEL_ENCODERS_PATH),
            'training_medians': joblib.load(TRAINING_MEDIANS_PATH),
'importance_df': pd.read_csv(IMPORTANCE_PATH),
            'explainer': joblib.load(SHAP_EXPLAINER_PATH)
        }
```

```
        with open(THRESHOLD_PATH, 'r') as f: artifacts['threshold'] =
json.load(f)['threshold']
        with open(MODEL_CHOICE_INFO_PATH, 'r') as f: artifacts['model_info'] =
json.load(f)
        return artifacts
    except Exception as e:
        st.error(f"Error loading artifacts: {e}"); traceback.print_exc(); return
None

def make_prediction(input_df, artifacts):
    if not artifacts: return None, None, None, None
    model, scaler, feature_list = artifacts['model'], artifacts['scaler'],
artifacts['feature_list']
    encoders, medians, threshold = artifacts['label_encoders'],
artifacts['training_medians'], artifacts['threshold']
    data = input_df.copy()
    try:
        for col, le in encoders.items():
            if col in data.columns:
                series =
data[col].astype(str).str.lower().str.strip().replace('nan', 'Missing')
                series.loc[data[col].isnull()] = 'Missing'
                known_classes = list(le.classes_)
                series[~series.isin(known_classes)] = 'Missing'
                data[col] = le.transform(series)
        for col in data.select_dtypes(include=np.number).columns:
            if data[col].isnull().any():
                impute_value = medians.get(col, 0)
                data[col].fillna(impute_value, inplace=True)
        for col in ['cpu_usage', 'traffic_volume', 'login_attempts', 'duration']:
            if col not in data: data[col] = 0
            else: data[col] = pd.to_numeric(data[col], errors='coerce').fillna(0)
        data['cpu_traffic_ratio'] = data['cpu_usage'] / (data['traffic_volume'] +
1e-6)
        data['cpu_traffic_interaction'] = data['cpu_usage'] *
data['traffic_volume']
        data['login_duration_ratio'] = data['login_attempts'] / (data['duration']
+ 1e-6)
        data['login_duration_interaction'] = data['login_attempts'] *
data['duration']
        X_final = pd.DataFrame(columns=feature_list, index=data.index)
        for col in feature_list: X_final[col] = data[col] if col in data else 0
        X_final = X_final.fillna(0)
        scaled_features = scaler.transform(X_final)
```

```
        probs = model.predict_proba(scaled_features)[:, 1]
        preds = (probs >= threshold).astype(int)
        return preds, probs, X_final, scaled_features
    except Exception as e:
        st.error(f"Prediction error: {e}"); traceback.print_exc(); return None,
None, None, None

# --- Main App ---
init_db()
st.title("Cyber Threat Detection Model")
st.markdown("An interactive interface to predict malicious activity using a pre-
trained Random Forest machine learning model.")

artifacts = load_artifacts()
if not artifacts:
    st.stop()

# --- Sidebar ---
with st.sidebar:
    st.header("Controls & Info")
    input_method = st.radio("Input Method", ("Live Simulation", "Batch Upload via
CSV"))
    st.markdown("---")

    if input_method == "Live Simulation":
        st.subheader("Live Simulation Parameters")
        traffic_volume = st.number_input("Traffic Volume", min_value=0.0,
format="%.2f", help="Total network traffic in bytes.")
        packet_size = st.number_input("Packet Size", min_value=0.0,
format="%.2f", help="Average size of network packets.")
        duration = st.number_input("Duration (seconds)", min_value=0.0,
format="%.2f", help="Duration of the connection or event.")
        login_attempts = st.number_input("Login Attempts", min_value=0, step=1,
help="Number of login attempts.")
        protocol_options = list(artifacts['label_encoders']['protocol'].classes_)
        protocol = st.selectbox("Protocol", options=protocol_options,
help="Network protocol used.")
        cpu_usage = st.slider("CPU Usage", 0.0, 1.0, help="System CPU usage (0.0
to 1.0) during the event.")

    st.markdown("---")
    st.subheader("Model Information")
    model_info = artifacts.get('model_info', {})
```

```
    st.write(f"**Model:** `{model_info.get('chosen_model_display_name',
'N/A')}`")
    st.write(f"**Prediction Threshold:** `{artifacts.get('threshold'):.2f}`")


# --- Main Page ---
input_df = None
if input_method == "Live Simulation":
    st.header("Live Simulation")
    input_data = {
        'traffic_volume': [traffic_volume], 'packet_size': [packet_size],
'protocol': [protocol],
        'duration': [duration], 'login_attempts': [login_attempts], 'cpu_usage':
[cpu_usage]
    }
    input_df = pd.DataFrame(input_data)
    st.write("Current Input Parameters:")
    st.dataframe(input_df)
else: # Batch Upload
    st.header("Batch Prediction")
    uploaded_file = st.file_uploader("Upload a CSV file.", type="csv")
    if uploaded_file:
        input_df = pd.read_csv(uploaded_file)
        st.write("Uploaded Data Preview:")
        st.dataframe(input_df.head())

if st.button("Detect Threat", type="primary"):
    if input_df is not None and not input_df.empty:
        with st.spinner("Analyzing..."):
            predictions, probabilities, preprocessed_df, scaled_features =
make_prediction(input_df, artifacts)

            if predictions is not None:
                st.success("Analysis Complete!")

                # Setup tabs for results
                tab1, tab2 = st.tabs(["Prediction Results", "Model Explanation"])

                with tab1:
                    if input_method == "Live Simulation":
                        prediction_label = "Malicious" if predictions[0] == 1
else "Not Malicious"
                        probability_text = f"{probabilities[0]:.2%}"
                        threshold = artifacts.get('threshold', 0.5)
```

```
                    st.subheader("Prediction Summary")
                    col1, col2, col3 = st.columns(3)
                    col1.metric(label="Prediction Outcome",
value=prediction_label)
                    col2.metric(label="Malicious Probability",
value=probability_text)
                    col3.metric(label="Decision Threshold",
value=f"{threshold:.2f}")

                    if predictions[0] == 1:
                        st.warning(f"**Threat Detected:** The event's
probability ({probability_text}) is above the {threshold:.2f} threshold.")
                    else:
                        st.success(f"**No Threat Detected:** The event's
probability ({probability_text}) is below the {threshold:.2f} threshold.")

                    log_prediction(input_df.to_dict('list'), "Malicious" if
predictions[0] == 1 else "Not Malicious", probabilities[0])

                st.subheader("Prediction Details")
                results_df = input_df.copy()
                results_df['Malicious Probability'] = [f"{p:.2%}" for p in
probabilities]
                results_df['Prediction'] = ['Malicious ' if p == 1 else 'Not
Malicious' for p in predictions]
                st.dataframe(results_df)

                if input_method == "Batch Upload via CSV":
                    csv = results_df.to_csv(index=False).encode('utf-8')
                    st.download_button("Download Results as CSV", csv,
"prediction_results.csv", "text/csv")

            with tab2:
                st.subheader("Global Feature Importance")
                st.markdown("This chart shows which features the model
*generally* finds most important across all data.")
                importance_df =
artifacts['importance_df'].sort_values(by='RF_Importance', ascending=False)
                chart = alt.Chart(importance_df.head(10)).mark_bar().encode(
                    x=alt.X('RF_Importance:Q', title='Importance Score'),
y=alt.Y('Feature:N', sort='-x', title='Feature'),
                    tooltip=['Feature',
'RF_Importance']).properties(title='Top 10 Most Important Features')
```

```
                    st.altair_chart(chart, use_container_width=True)

                    if input_method == "Live Simulation":
                        st.markdown("---")
                        st.subheader("Local Prediction Explanation")
                        st.markdown("This waterfall plot shows how each feature
pushed the prediction from the model's base value to its final output. Red
features increase the score (more malicious), while those in blue decrease it.")

                        explainer = artifacts['explainer']
                        shap_explanation = explainer(scaled_features)


                        shap_explanation_obj = shap.Explanation(
                            values=shap_explanation.values[0, :, 1],
                            base_values=shap_explanation.base_values[0, 1],
                            data=preprocessed_df.iloc[0].values,
                            feature_names=preprocessed_df.columns.tolist()
                        )

                        # 2. Calling the plot function
                        shap.waterfall_plot(shap_explanation_obj, show=False)

                        # 3. Getting the current matplotlib figure and passing to
Streamlit.
                        fig = plt.gcf()
                        st.pyplot(fig)

    else:
        st.warning("Please provide input data before predicting.")

if st.checkbox("Show Raw Prediction Log"):
    st.subheader("Full Prediction Log")
    st.dataframe(get_log_data())
```

Appendix C: Dataset

Cyber_Threat_Detect
ion_Dataset_2024.csv