A Mini Project Report On

AI-DRIVEN INTRUSION DETECTION: A FEDERATED LEARNING DATASET FOR CLOUD AND EDGE SECURITY

Submitted in partial fulfillment of the requirements

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING (AI & ML)

Submitted by

M.ABHINAY (23UJ5A6607)

K. RANJITH REDDY (23UJ5A6618)

N.PRAVEEN (23UJ5A6610)

B. GOWTHAM (22UJ1A6667)

Under the esteemed guidance of

Mr. S.Abhishek Yadav MTech.

Assistant professor

Department of computer science & engineering (AI & ML)

At

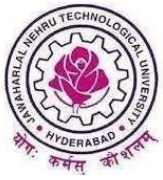
MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES

Kistapur (V), Medchal (M), Medchal Malkajgiri (Dist)-501401.

(An UGC autonomous Institution, NBA Accredited in **CSE, ECE, IT, EEE**)

2022-2026

Affiliated to



JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KUKATPALLY, HYDERABAD-85.

MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES

Kistapur (V), Medchal (M), Medchal Malkajgiri (Dist)-501401.

(An UGC autonomous Institution, NBA Accredited in CSE, ECE, IT, EEE)

CERTIFICATE

This is to certify that the Mini project entitled **" AI-DRIVEN INTRUION DETECTION:A FEDERATED LEARNING DATASET FOR CLOUD AND EDGE SECURITY"**

submitted by

M. **ABHINAY** **(23UJ5A6607)**
K. RANJITH REDDY **(23UJ5A6618)**
N.PRAVEEN **(23UJ5A6610)**
B. GOWTHAM **(22UJ1A6667)**

To the Dept. of CSE (AI & ML), **MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES**, in partial fulfilment for the requirement for the ward of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (AI & ML)** during the academic year 2024-2025

Internal Guide

Mr. S. ABHISHEK YADAV MTech

Assistant Professor

Dept. of CSE(AI&ML)

Head of the department

Mr. E.BABU MTech

Assistant Professor

Dept. of CSE(AI&ML)

EXTERNAL EXAMINER:

ACKNOWLEDGEMENT

Any attempt at any level can't be satisfied completely without the report and guidance of learned people. These words are not enough to show our gratitude towards them. We would like to express our thanks to them.

` We would like to express our immense gratitude to **Mr. V. MALLA REDDY, chairman** of "Malla Reddy Engineering College and Management Sciences" for accepting us and providing us with an opportunity to do a project in their esteemed organization.

We are thankful that **Mr. V. BHARATH SIMHA REDDY,** director of "Malla Reddy Engineering College and Management Science helped us to undergo project work as a part of the university curriculum.

We are thankful that **Dr. M. SREEDHAR REDDY,** Principal of Malla Reddy Engineering college and Management sciences helped us to undergo project work as part of the university curriculum

Our special thanks to **Mr. E. BABU** $_{\text{MTech}}$ Asst. Professor & Head of CSE(AI&ML) Department and **Mr. S. ABHISHEK YADV** $_{\text{MTech}}$ Asst. Professor of CSE(AI&ML) Department for guiding us in the right way to complete our project at the right time.

Their immense guidance and supervision have been a significant factor in the successful completion of our project. Their enthusiasm and interest in our project have been highly inspiring and has motivated us to put in our best efforts.

And we would like to express our heartfelt thanks to **Faculty Members, Lab technicians, Nonteaching staff of the CSE(AI&ML) Department** one and all who have helped us directly or indirectly in successful completion of the Mini Project.

MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES

Kistapur (V), Medchal (M), Medchal Malkajgiri (Dist)-501401.

(An UGC autonomous Institution, NBA Accredited in CSE, ECE, IT, EEE)

DECLARATION

We declare that the Mini project entitled **"AI-DRIVEN INTRUSION DETECTION:A FEDERATED LEARNING DATASET FOR COULD AND EDGE SECURITY",** is original and Bonafide work of our own for the award of the degree of **BACHELOR OF TECHNOLOGY** and submitted to the **Department of CSE(AI&ML), MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES.,** under the guidance of **Mr. S .ABHISHEK YADAV** $_{\text{MTech}}$**, Assistant Professor** and has not been copied from any other earlier reports.

By:

M.ABHINAY (23UJ5A6607)
K. RANJITH REDDY (23UJ5A6618)
N.PRAVEEN (23UJ5A6610)
B. GOWTHAM (22UJ1A6667)

ABSTRACT

In recent years, the proliferation of Internet of Things (IoT) devices has significantly increased the volume of network traffic and the complexity of network environments. According to the Global IoT Security Market Report, the number of IoT devices surged from approximately 8.4 billion in 2017 to an estimated 30.9 billion by 2025. This exponential growth has led to a corresponding rise in sophisticated network attacks, with Mirai and BASHLITE botnets being prominent examples. The Mirai botnet, which first emerged in 2016, exploited IoT vulnerabilities to launch large-scale Distributed Denial of Service (DDoS) attacks, while BASHLITE, identified in 2014, has been known for its effective use of compromised devices in various cyber-attacks. Traditional methods for detecting anomalies in network traffic typically involve manual inspection and rule-based systems. These approaches face several challenges, including the high volume of data, the dynamic nature of network traffic, and the evolving tactics of attackers. Manual methods are often laborintensive, prone to human error, and struggle to keep pace with sophisticated attack techniques. Rule-based systems, while useful, are limited by their inability to adapt to new and previously unseen attack patterns. Machine Learning (ML) offers a promising alternative for addressing these limitations. By leveraging algorithms capable of learning from historical data and adapting to new patterns, ML models can identify anomalous traffic indicative of Mirai and BASHLITE attacks more effectively. These models can analyze vast amounts of network data in real time, detect subtle deviations from normal behavior, and improve detection accuracy over time. This approach not only enhances the ability to identify emerging threats but also reduces the dependency on manual intervention, leading to more efficient and scalable network security solutions.

TABLE OF CONTENT

ABSTRACT	I
LIST OF FIGURES	II
ABBREVIATIONS	III
CHAPTER 1: INTRODUCTION	1
1.1 OVERVIEW	2
1.2 NEED	2
1.3 RESEARCH MOTIVATION	2
1.4 PROBLEM STATEMENT	2
1.5 OBJECTIVES	3
1.6 ADVANTAGES	3
1.7 APPLICATIONS	4
CHAPTER 2: LITERATURE SURVEY	5
CHAPTER 3: EXISTING METHODOLOGY	28
CHAPTER 4: PROPOSED SYSTEM	30
4.1 OVERVIEW	30
4.2 DATA PREPROCESSING AND SPLITTING	32
4.3 ML MODEL BUILDING	33
4.3.1 NAÏVE BAYES CLASSIFIER	33
4.3.2 RANDOM FOREST CLASSIFIER	34
4.3.3 COMPARISON AND WHEN RANDOM FOREST PERFORMS BETTER	34
4.4 ADVANTAGES OF THE PROPOSED SYSTEM	35
CHAPTER 5: UML DIAGRAM	37
CHAPTER 6: SOFTWARE ENVIRONMENT	43
CHAPTER 7: SYSTEM REQUIREMENTS SPECIFICATIONS	59
CHAPTER 8: FUNCTIONAL REQUIREMENTS	60
CHAPTER 9: SOURCE CODE	65
CHAPTER 10: RESULTS AND DISSCUSSION	70
10.1 IMPLEMENTATION DESCRIPTION	70
10.2 DATASET DESCRIPTION	71

10.2.1 DATA SOURCE 72

10.2.2 FEATURES AND ATTRIBUTES 72

10.2.3 DATA PREPROCESSING 72

10.2.4 DATA SPLITTING 73

10.2.5 DATA VISUALIZATION 73

10.2.6 DATASET USAGE 73

10.3 RESULT DESCRIPTION 74

CHAPTER 11: CONCLUSION AND FUTURE SCOPE 78

11.1 CONCLUSION 78

11.2 FUTURE SCOPE 78

REFERENCES 81

LIST OF FIGURES

Figure 4.1**:** Architecture Diagram of Proposed System 32

Figure 5.1: Class Diagram 38

Figure 5.2: Data Flow Diagram 39

Figure 5.3: Sequence Diagram 40

Figure 5.4: Activity Diagram 41

Figure 5.5: Deployment Diagram 41

Figure 5.6: Use Case Diagram 42

Figure 5.7: Component Diagram 42

Figure 10.1: Overview of the Dataset 74

Figure 10.2: Dataset Description 74

Figure 10.3: Correlation Heatmap 75

Figure 10.4: Count Plot of Attack Categories 76

Figure 10.5: Confusion Matrix of Bernoulli Naive Bayes Classifier 76

Figure 10.6: Confusion Matrix of Random Forest Classifier 77

Figure 10.7: Prediction on New Test Data 77

ABBREVIATIONS

APIs	Application Programming Interface
AI	Artificial Intelligence
BSBODP	Bridge Sample Based Online Distillation Protocol
CFL	Clustered Federated Learning
CCPA	California Consumer Privacy Act
DL	Deep Learning
EECC	End-Edge-Cloud Collaboration
FL	Federated Learning
FRL	Federated Reinforcement Learning
FedAgg	Agglomerative Federated Learning
GDPR	General Data Protection Regulation
GANs	Generative Adversarial Networks
HierFAVG	Hierarchical Federated Averaging
IIoT	Industrial Internet of Things
IoT	Internet of Things
IDS	Intrusion Detection Systems
IPS	Intrusion Prevention System
IoV	Internet of Vehicles
MFA	Multi-Factor Authentication
NLP	Natural Language Processing
QoS	Quality of Service
RBAC	Role-Based Access Control
RL	Reinforcement Learning
SLR	Systematic Literature Review
UAV	Unmanned Aerial Vehicle
XAI	Explainable AI

CHAPTER 1
INTRODUCTION

1.1 Overview

In the contemporary digital landscape, the proliferation of Internet of Things (IoT) devices has revolutionized various industries, leading to unprecedented growth in network traffic and complexity. According to the Global IoT Security Market Report, the number of IoT devices surged from approximately 8.4 billion in 2017 to an estimated 30.9 billion by 2025. This exponential increase has simultaneously escalated the risk and frequency of sophisticated network attacks, with the Mirai and BASHLITE botnets being particularly notorious. The Mirai botnet, first identified in 2016, exploited vulnerabilities in IoT devices to orchestrate large-scale Distributed Denial of Service (DDoS) attacks, while the BASHLITE botnet, discovered in 2014, has been associated with various cyber-attacks leveraging compromised devices.

Traditional anomaly detection methods in network traffic, which often rely on manual inspection and rule-based systems, are increasingly inadequate in this evolving threat landscape. These conventional approaches face significant challenges due to the vast volume of data, the dynamic nature of network environments, and the continuously evolving tactics of attackers. Manual methods are labour-intensive, prone to human error, and struggle to keep pace with the sophistication of modern cyber threats. Rule-based systems, although useful, lack the flexibility to adapt to new and previously unseen attack patterns.

In response to these limitations, Machine Learning (ML) presents a promising solution. ML algorithms, with their capacity to learn from historical data and adapt to new patterns, offer enhanced capabilities for identifying anomalous traffic indicative of Mirai and BASHLITE botnet activities. By analysing extensive network data in real time, ML models can detect subtle deviations from normal behaviour, thereby improving detection accuracy and response times. This approach not only bolsters the ability to identify emerging threats but also minimizes reliance on manual intervention, paving the way for more efficient and scalable network security solutions.

1.2 Need

The rapid expansion of the Internet of Things (IoT) has revolutionized the way we interact with technology, enabling a vast network of connected devices that enhance convenience, efficiency, and automation. However, this proliferation has also introduced significant security vulnerabilities. With the number of IoT devices expected to reach 30.9 billion by 2025, the potential for cyber-attacks has grown exponentially. Botnets like Mirai and BASHLITE exploit these vulnerabilities, launching large-scale Distributed Denial of Service (DDoS) attacks and other malicious activities, threatening the stability and security of global networks. There is an urgent need for robust, intelligent, and scalable solutions to detect and mitigate these threats in real time, ensuring the protection of critical infrastructure and sensitive data.

1.3 Research Motivation

The motivation behind this research stems from the increasing frequency and sophistication of cyber-attacks targeting IoT networks. Traditional security measures, such as rule-based systems and manual inspections, are often inadequate in identifying and responding to these evolving threats. The limitations of these methods, coupled with the potential damage that botnet attacks can cause, highlight the necessity for more advanced detection techniques. Machine learning, with its ability to learn from data and identify patterns, offers a promising approach to enhancing network security. This research is driven by the desire to develop an AI framework capable of detecting anomalous network traffic associated with IoT botnets, thereby providing a proactive defense against cyber threats.

1.4 Problem Statement

The proliferation of IoT devices has led to a significant increase in network traffic, making it challenging to distinguish between legitimate and malicious activity. Botnets like Mirai and BASHLITE exploit IoT vulnerabilities to orchestrate large-scale cyber-attacks, often going undetected by traditional security measures. The problem is exacerbated by the dynamic nature of network environments and the evolving tactics of attackers. The primary problem addressed by this research is the development of an AI-driven framework that can accurately identify anomalous network traffic associated with these botnet attacks, providing real-time detection and response capabilities to mitigate potential threats.

1.5 Objective

The objective of this research is to design and implement a machine learning-based framework for the detection of anomalous network traffic related to Mirai and BASHLITE IoT botnet attacks. The framework aims to enhance the accuracy and efficiency of network security systems by leveraging algorithms that can learn from historical data and adapt to new attack patterns. Specific objectives include:

Developing a comprehensive dataset that captures the characteristics of normal and anomalous network traffic.

Implementing and evaluating machine learning models, such as Bernoulli Naive Bayes and Random Forest classifiers, for anomaly detection.

Integrating the framework into real-time network monitoring systems to provide continuous threat detection.

Assessing the framework's performance using metrics such as accuracy, precision, recall, and F1-score.

1.6 Advantages

The AI framework proposed in this research offers several key advantages over traditional security measures:

Real-time Detection: The framework is capable of analyzing network traffic in real-time, enabling immediate identification and response to potential threats.

Scalability: The machine learning models can handle large volumes of data, making the framework suitable for use in extensive IoT networks.

Adaptability: The framework can learn from new data and adapt to emerging threats, ensuring continuous protection against evolving attack tactics.

Reduced Manual Intervention: By automating the detection process, the framework minimizes the need for manual inspection, reducing the risk of human error and freeing up resources for other critical tasks.

Improved Accuracy: The use of advanced machine learning algorithms enhances the accuracy of anomaly detection, reducing false positives and negatives.

1.7 Applications

The proposed AI framework has broad applications across various industries and sectors that rely on IoT networks:

Smart Cities: Ensuring the security of IoT devices in smart city infrastructure, including traffic management systems, surveillance, and public services.

Healthcare: Protecting sensitive medical devices and patient data from cyber-attacks in connected healthcare environments.

Industrial IoT: Securing industrial control systems and manufacturing processes from disruptions caused by botnet attacks.

Energy Sector: Safeguarding smart grids and energy distribution networks from cyber threats.

Telecommunications: Enhancing the security of communication networks that support IoT devices and services.

Home Automation: Providing security for smart home devices, including thermostats, cameras, and voice assistants, against potential intrusions.

CHAPTER 2

LITERATURE SURVEY

1. **Soni, V.; Modi, P.; Chaudhri, V. (2013)** This paper investigates sinkhole attacks in wireless sensor networks (WSNs), which are foundational to many IoT systems. The authors explain how malicious nodes can attract traffic by advertising false routing information, leading to packet interception or dropping. They propose a method to detect sinkhole nodes based on analyzing node behavior such as routing updates and data flow patterns. A detection algorithm is designed using routing anomalies and packet loss metrics. The method is evaluated using simulation environments, demonstrating its ability to detect the presence of a sinkhole with reasonable accuracy. However, its effectiveness in large-scale networks with dynamic topologies remains a challenge. The study underscores the need for lightweight, distributed IDS suitable for resource-constrained environments. It serves as a precursor to later intrusion detection methods in IoT. Importantly, it highlights the difficulty of applying traditional cybersecurity principles directly to sensor networks. The paper remains relevant for understanding basic IoT attack detection strategies.
2. **Abomhara, M.; Køien, G.M. (2015)** This comprehensive survey discusses the security risks associated with IoT devices, covering vulnerabilities, threats, and types of attacks. The authors categorize intruders and threats at different layers of IoT, including device, network, and application layers. Key vulnerabilities include insecure communication, insufficient authentication, and lack of physical protection. The paper emphasizes the unique challenges posed by IoT's scale, heterogeneity, and lack of centralized control. Various attack types like eavesdropping, spoofing, and denial-of-service are discussed in detail. The authors call for new security architectures that go beyond traditional IT security models. Privacy preservation and trust establishment are identified as core challenges. The paper proposes high-level recommendations but does not offer implementation-level details. It laid early groundwork for IoT-specific security research. This work is widely cited for its clear framework and foundational insights into IoT threat landscapes.
3. **Andrea, I.; Chrysostomou, C.; Hadjichristofi, G. (2015)** This paper outlines the security vulnerabilities and challenges in the Internet of Things, focusing on smart devices and sensor networks. It identifies flaws in protocols and architectures used in IoT systems, such as weak encryption and unsecured firmware. A significant portion is devoted to examining threats in consumer IoT, particularly in smart homes and healthcare. The authors provide a taxonomy of attacks, including device spoofing, replay attacks, and data manipulation. They emphasize that existing network security solutions are inadequate for resource-constrained IoT devices. Suggested approaches include using lightweight cryptographic techniques and access control mechanisms. The paper advocates for proactive risk assessment before deploying IoT systems. Though it does not introduce novel detection models, its value lies in clearly defining emerging attack surfaces. The recommendations aim to influence future secure-by-

design IoT development. The work remains important for researchers studying the early stages of IoT security evolution.

4. **Sivaraman, V.; Gharakheili, H.H.; Vishwanath, A.; Boreli, R.; Mehani, O. (2015)** This study proposes a network-level security mechanism for managing and protecting smart-home IoT devices. It introduces a smart gateway that can monitor and control device behavior, effectively serving as a firewall for IoT environments. The system tracks DNS queries, connection requests, and unusual traffic patterns to detect unauthorized behavior. Unlike host-based systems, this network-level control is non-intrusive and suitable for low-power devices. A prototype was implemented and tested in a real apartment setup, showing promising results in detecting traffic anomalies. The approach emphasizes ease of deployment for end-users with minimal configuration. However, the authors acknowledge limitations in encrypted traffic inspection and centralized control. The study contributes by offering a practical architecture for IoT security at the consumer level. It bridges the gap between theoretical security frameworks and real-world smart home deployments. The paper is notable for its pragmatic focus and real-time applicability.
5. **Ronen, E.; Shamir, A. (2016)** This paper reveals a novel class of side-channel attacks targeting smart IoT devices, particularly smart lights. The authors demonstrate how attackers can exploit the over-the-air firmware upgrade mechanism to gain control of Philips Hue light bulbs. Once compromised, the devices can be used to create a covert communication channel or to disrupt entire lighting systems. The researchers further show that an attacker with a drone or nearby vantage point could inject malicious updates wirelessly. The study also explores how data could be leaked from air-gapped networks using visual light modulation. The attack raises concerns about extended functionality being exploited in IoT without proper security oversight. Mitigation strategies such as cryptographic signing of updates are suggested. The paper is pioneering in introducing physical-layer side-channel attacks in consumer IoT. It sparked awareness in the industry about non-conventional attack vectors. This work has broad implications for firmware security and over-the-air update mechanisms.
6. **Deogirikar, J.; Vidhate, A. (2017)** This paper presents a broad survey of security attacks targeting the Internet of Things (IoT), offering a classification based on the IoT stack: perception, network, and application layers. Attacks like DoS, replay, Sybil, and man-in-the-middle are thoroughly described with examples. The authors discuss how these attacks exploit limitations in bandwidth, power, and storage. They also survey defensive mechanisms such as intrusion detection, cryptography, and trust management. The study emphasizes the lack of standardized security models across vendors. Notably, it addresses how attack surfaces evolve due to the heterogeneity of devices. The paper is educational, offering concise explanations and threat-taxonomy diagrams. However, it lacks experimental validation or performance evaluations. Its strength lies in mapping threats to specific IoT components. The work is frequently referenced in later FL and ML-based security studies.
7. **Meidan, Y.; Bohadana, M.; Shabtai, A.; Ochoa, M.; Tippenhauer, N.O.; Guarnizo, J.D.; Elovici, Y. (2017)** This research introduces a machine learning-based framework for identifying unauthorized IoT devices by analyzing network traffic behavior. It utilizes features such as packet inter-arrival time, protocol usage, and byte volume to create device fingerprints. The system was tested on a smart environment testbed with

15 types of IoT devices. Using Random Forest classifiers, the authors achieved detection accuracies of over 95%, even with imbalanced classes. The framework effectively flags unauthorized devices joining the network, which is crucial for preventing rogue device exploitation. It operates passively, with no need for direct access to the device or its firmware. The model can also identify device impersonation and anomalous usage. This paper is an early example of using ML for real-time, network-level IoT intrusion detection. It inspired future research in FL for distributed device monitoring. The work demonstrates high practicality for smart homes and enterprise settings.

8. **Alaba, F.A.; Othman, M.; Hashem, I.A.T.; Alotaibi, F. (2017)** This survey provides a comprehensive overview of IoT security issues, challenges, and available countermeasures. It categorizes security concerns into device-level, network-level, and application-level threats. The authors highlight vulnerabilities such as weak encryption, lack of access control, and firmware integrity. The study discusses key security goals—confidentiality, integrity, availability—and how current systems fail to meet them. It reviews various security solutions, including blockchain, lightweight cryptography, and intrusion detection systems. However, it also identifies critical gaps in scalability and adaptability of these approaches. One of the significant contributions is the discussion on regulatory and ethical aspects of IoT security. The paper suggests a need for collaboration between academia, industry, and government to create viable security frameworks. It does not introduce new methods but provides a solid foundation for future solutions. The work is useful for understanding the security landscape of IoT before federated or AI-based systems became mainstream.
9. **McDermott, C.D.; Majdani, F.; Petrovski, A.V. (2018)** This paper presents a deep learning-based system for botnet detection in IoT networks. Using datasets like BoT-IoT and NSL-KDD, the authors trained neural networks to distinguish between normal and botnet-affected traffic. They tested different deep learning models—DNN, CNN, and RNN—and reported over 98% accuracy in several scenarios. The study highlights the scalability and efficiency of DL models for processing massive IoT network logs. One of the contributions is its architecture design that can work on edge or cloud systems depending on resource availability. The paper also discusses class imbalance and noise in datasets, which can impact the performance of classifiers. A limitation acknowledged is the lack of generalizability across different datasets without retraining. Still, the research forms a cornerstone for intrusion detection systems in federated environments. It provides insight into how FL could extend such deep learning models in decentralized networks. The work laid groundwork for future FL-based IDS designs.
10. **Doshi, R.; Apthorpe, N.; Feamster, N. (2018)** This study focuses on detecting Distributed Denial of Service (DDoS) attacks launched by compromised IoT devices using machine learning techniques. It identifies specific traffic features, such as outbound packet rate and entropy, that are indicative of botnet behavior. Using decision trees, support vector machines, and neural networks, the authors achieve high accuracy and low false positive rates. A notable strength is the use of minimal feature sets, which allows lightweight implementations suitable for consumer-grade routers or gateways. The dataset was generated using real smart devices under lab-controlled attacks, enhancing the practical relevance. The system detects attacks in under five seconds, demonstrating suitability for real-time applications. It is among the first works applying ML to detect IoT-driven DDoS at the network edge. This research indirectly motivates

federated learning approaches by illustrating the value of localized detection. The study is widely cited for its simplicity and high effectiveness.

11. **Brun, O.; Yin, Y.; Gelenbe, E. (2018)** The authors introduce a novel deep learning architecture called the Dense Random Neural Network (dRNN) to detect cyber-attacks on smart home environments. The model is designed to process both sequential and categorical traffic features. It is tested against common IoT attacks, such as spoofing, replay, and scanning. The dRNN achieves higher efficiency than traditional RNNs in training time and memory consumption. The study also evaluates false positive rates and robustness across multiple datasets. One key contribution is the lightweight nature of the model, which is ideal for resource-constrained IoT nodes. The paper argues that smart home security should emphasize embedded, device-level intelligence. Although FL is not mentioned, this architecture aligns well with FL concepts due to its decentralized capability. The study has inspired subsequent research into efficient DL models for edge deployment. It stands out for blending neural theory with cybersecurity applications.
12. **Yavuz, F.Y.; Devrim, Ü.N.A.L.; Ensar, G.Ü.L. (2018)** This paper investigates the use of deep learning for detecting routing attacks in IoT networks, focusing specifically on blackhole and grayhole attacks. The authors simulate IoT environments using NS-3 and train deep learning models like CNN and LSTM to classify network behavior. Key features used include hop count variations, packet delivery ratios, and control message frequencies. The model achieved high detection accuracy with low false alarm rates, making it suitable for lightweight IoT environments. It addresses the challenge of dynamic topology in mobile and ad hoc networks. The authors suggest that localized, adaptive models would perform better than centralized systems. Though not implemented, they discuss the potential for decentralized approaches like federated learning. This study is among early adopters of DL in securing routing protocols. Its insights into malicious node detection continue to inform research in distributed security. The paper bridges ML with protocol-level IoT threat detection.
13. **Shiranzaei, A.; Khan, R.Z. (2018)** This research proposes a novel technique to detect two major IoT routing attacks: sinkhole and selective forwarding. The method monitors packet delivery ratios and route changes to identify inconsistencies. The authors develop a trust-based algorithm that quantifies node behavior using fuzzy logic. Anomalous behavior triggers alerts, which are then analyzed by a centralized controller. The detection mechanism is tested in simulation and found effective under moderate network loads. The study acknowledges trade-offs between detection accuracy and communication overhead. While the approach is semi-centralized, it highlights the potential for distributed trust computation. This aligns with the goals of federated learning, where local computations help detect anomalies. The paper is valuable for its dual-focus on both major routing attacks and its lightweight detection strategy. Its use of trust metrics is particularly useful in resource-constrained environments. The model is flexible enough to be integrated into larger FL systems.
14. **Palacharla, S.; Chandan, M.; GnanaSuryaTeja, K.; Varshitha, G. (2018)** The authors present a comprehensive study on wormhole attacks in IoT networks, where attackers tunnel packets through private channels to disrupt routing. The paper outlines attack models and shows how they can bypass traditional encryption methods. The authors implement a detection method based on round-trip time and packet travel paths.

They propose using multiple checkpoints to monitor packet integrity and latency. The method is simulation-based and applicable to mesh and ad hoc topologies. While not proposing a fully distributed system, the study discusses scalability issues that decentralized models like federated learning could address. The paper provides visual examples of how wormhole links skew topology mapping. It emphasizes that wormhole attacks are hard to detect with basic IDS. The work is foundational for understanding timing-based and topological attacks in IoT. It has helped inspire timing and trust-based detection in FL-integrated IDS systems.

15. **Neshenko, N.; Bou-Harb, E.; Crichigno, J.; Kaddoum, G.; Ghani, N. (2019)** This exhaustive survey examines the entire spectrum of IoT security challenges, compiling more than 200 sources. It maps threats across all IoT layers and discusses known large-scale exploitations. The authors present an empirical analysis of Internet-wide scanning data to show how exposed IoT devices are. A unique contribution is their inclusion of honeypot data, highlighting real-world attack trends. They underscore that most attacks exploit weak credentials, open ports, and outdated firmware. The study criticizes current industry practices that prioritize functionality over security. It argues for regulation and standardization alongside technical solutions. The authors suggest federated or distributed security approaches to handle device heterogeneity. This paper is a vital resource for researchers defining the threat landscape for FL-based IDS. It remains one of the most frequently cited surveys on practical IoT attack scenarios. The work significantly strengthens the case for proactive and collaborative detection frameworks.
16. **Demotes-Mainard, J.; Cornu, C.; Guerin, A.; et al. (2019)** This article analyzes how the General Data Protection Regulation (GDPR) impacts clinical and IoT-related research. It explains how GDPR emphasizes user consent, data minimization, and transparency—issues central to federated learning. The authors argue that centralizing sensitive health data for training AI models contradicts GDPR principles. They suggest privacy-preserving approaches like FL as a solution to maintain compliance while enabling innovation. The paper also discusses the complexity of cross-border data transfers and data subject rights. The clinical research sector, due to strict ethical standards, is viewed as a testing ground for secure AI methods. Although not technical, this work provides a legal foundation for integrating FL in regulated environments. It highlights the need for privacy-by-design in data science projects. The paper is especially relevant for medical IoT and wearable device research. It reinforces the push for FL in sensitive domains like healthcare and finance.
17. **Barrett, C. (2019)** This commentary paper discusses how GDPR and the California Consumer Privacy Act (CCPA) are shaping global data privacy standards. It argues that these regulations are indirectly forcing the tech industry to adopt user-centric data processing models. The author suggests federated learning as a viable architecture to meet compliance without compromising AI capabilities. Though not a technical paper, it links legal evolution with system design, advocating for privacy-aware infrastructure. It critiques current models that rely on vague consent or opaque data usage. Importantly, the article mentions that compliance is not just a technical task but a socio-technical challenge. This perspective supports the broader adoption of FL and privacy-preserving ML in cloud-edge ecosystems. It serves as a thought piece for policymakers and

engineers alike. The commentary encourages long-term, global thinking about AI systems' structure. Its relevance lies in policy alignment for FL deployments.

18. **Hao, M.; Li, H.; Luo, X.; Xu, G.; Yang, H.; Liu, S. (2019)** This paper introduces a privacy-enhanced federated learning framework tailored for Industrial AI applications. The authors integrate homomorphic encryption and differential privacy into the FL process. Their framework is applied to predictive maintenance and fault detection in industrial control systems. The results show that even with added privacy layers, the FL models maintain high accuracy and performance. The study presents a performance comparison between centralized, FL, and privacy-enhanced FL setups. One of the strengths is its emphasis on deployment feasibility, using real industrial datasets. The paper also discusses threats like model inversion and gradient leakage. It suggests architectural modifications to enhance resilience. This work is among the early examples of applying FL to the Industrial Internet of Things (IIoT). It demonstrates that strong privacy guarantees need not compromise model utility. The paper is a benchmark for privacy-preserving FL in industry.
19. **Bhaskar, V.V.S.R.; Etikani, P.; Shiva, K.; Choppadandi, A.; Dave, A. (2019)** This paper presents methods for building explainable AI systems by integrating federated learning (FL) on cloud platforms. It discusses how transparency and interpretability can be maintained when models are trained collaboratively across client devices. The authors highlight explainable techniques such as feature attribution, rule extraction, and model visualization tailored for FL. They experiment with prototype systems demonstrating improved user trust without compromising data privacy. The study addresses challenges like aggregating local explanations into a unified global interpretation. The work underscores the need for auditability and accountability in distributed AI. Although not focused on security, their frameworks are applicable to intrusion detection systems requiring interpretability. It serves as a reference for designing transparent FL solutions in regulated sectors. The paper bridges the gap between federated model performance and ethical AI requirements.
20. **Zhao, Z.; Feng, C.; Yang, H.H.; Luo, X. (2020)** This paper introduces the fundamental architecture and technological framework of Federated-Learning-enabled Fog Radio Access Networks (F-RANs). The authors explore how integrating FL into fog networks enhances privacy, reduces latency, and balances computational loads in smart city and IoT environments. They detail the core challenges such as resource allocation, model synchronization, and privacy preservation. Several key FL optimization techniques like asynchronous updates and model compression are discussed. The framework supports low-latency, localized learning which is ideal for time-sensitive edge applications. They also envision future expansions involving 6G and AIoT. Though theoretical, it lays a foundation for real-world FL deployment in telecommunications and smart grids. It underlines how cloud-edge synergy is strengthened via FL. The paper serves as a reference point for designing intelligent, privacy-aware F-RANs. Its insights directly contribute to the FL-based intrusion detection domain.
21. **Drainakis, G.; Katsaros, K.V.; Pantazopoulos, P.; Sourlas, V.; Amditis, A. (2020)** This paper compares centralized and federated machine learning in scenarios involving users with varying privacy preferences. The authors present a hybrid evaluation model where users decide what data to share based on privacy elasticity. They simulate both FL and centralized learning with a use case in smart transportation. The results show FL

offers near-centralized accuracy while satisfying privacy demands. This work uniquely introduces "privacy elasticity" as a quantifiable concept. The study addresses the communication vs. performance trade-off in heterogeneous edge networks. It suggests adaptive aggregation and local differential privacy to meet user-specific constraints. Though not focused on security, the methodology is applicable to FL-based IDS systems. It promotes fairness, personalization, and user control in FL systems. This work aligns with ethical AI principles and supports human-centric intrusion detection models.

22. **Sunyaev, A. (2020)** In this book chapter, Sunyaev provides a comprehensive analysis of cloud computing fundamentals, covering service models (IaaS, PaaS, SaaS), deployment models, and enabling technologies. The chapter explores core components such as virtualization, elasticity, and multitenancy. A special emphasis is placed on cloud security risks, data sovereignty, and regulatory compliance. Although FL is not the primary focus, the author suggests decentralized architectures like edge computing and federated learning as future directions for privacy preservation. The chapter explains how trust management and secure APIs are critical in distributed environments. It also touches upon the role of AI in cloud-based analytics, paving the way for secure, intelligent cloud-edge systems. The work is foundational for understanding the infrastructure supporting FL-based intrusion detection. It provides the architectural and operational context within which FL operates. A useful theoretical base for students and practitioners in cloud security.
23. **Li, L.; Fan, Y.; Tse, M.; Lin, K.Y. (2020)** This paper presents a comprehensive review of federated learning applications across multiple sectors, including healthcare, finance, and IoT. The authors categorize FL frameworks based on architecture (cross-device vs. cross-silo), aggregation strategy, and privacy mechanisms. Use cases demonstrate how FL enables collaborative model training without raw data exchange. Challenges such as data heterogeneity, communication overhead, and adversarial attacks are discussed in depth. The review identifies research gaps in FL model validation and lifecycle management. Although not specific to intrusion detection, the insights can be adapted to IDS model training across organizations. It also presents benchmark FL frameworks like TensorFlow Federated and PySyft. The paper underscores the necessity of interpretability and accountability in FL systems. This work is a key reference for understanding the versatility and technical landscape of FL. It’s especially valuable for aligning security-focused FL implementations with general trends.
24. **Kholod, I.; Yanaki, E.; Fomichev, D.; Shalugin, E.; Novikova, E.; Filippov, E.; Nordlund, M. (2020)** This comparative review evaluates several open-source federated learning frameworks tailored for IoT environments. The authors assess platforms like TensorFlow Federated, PySyft, FATE, and PaddleFL based on performance, scalability, and privacy features. They also measure resource efficiency, making the study especially relevant for constrained IoT devices. The paper includes experimental benchmarks simulating IoT training tasks on lightweight sensors. A critical finding is that most current frameworks require significant customization for real-time or embedded systems. The review emphasizes the need for lightweight encryption, efficient aggregation, and failure recovery in FL systems. By focusing on IoT use cases, the authors provide a roadmap for adapting FL frameworks to cloud-edge security

needs. This work is instrumental for selecting appropriate development tools for FL-based IDS. It also highlights gaps in current frameworks that future research can address.

25. **Golosova, J.; Romanovs, A. (2020)** This study analyzes the advantages and disadvantages of blockchain technology, particularly in relation to security and trust. It categorizes use cases into data integrity, authentication, and access control. While not specific to FL, the paper proposes blockchain as a viable solution for FL participant verification and auditability. It warns, however, about scalability, energy consumption, and latency issues. A unique contribution is its evaluation of blockchain as a mechanism for enforcing smart contracts in distributed environments. This could be leveraged to govern FL model updates or penalize malicious contributors. The authors suggest that combining blockchain with FL enhances trustworthiness in untrusted networks. The paper is theoretical but serves as a launchpad for blockchain-enhanced FL research. It lays the foundation for hybrid architectures combining FL, edge computing, and blockchain. Such combinations are becoming increasingly common in secure federated intrusion detection.
26. **Nilsson, A.; Smith, S.; Ulm, G.; Gustavsson, E.; Jirstrand, M. (2020)** The paper benchmarks various federated learning algorithms across different devices and network setups to assess performance. It compares FedAvg, FedProx, and adaptive FL models in terms of communication rounds, convergence speed, and accuracy loss. The authors simulate federated training in scenarios with non-IID data and varying network latencies. They find that simple averaging strategies degrade in heterogeneous environments. FedProx and adaptive update algorithms yield better convergence but require careful tuning. The study is essential for selecting suitable FL algorithms for resource-limited IoT or edge networks. Its testbed, based on real device emulations, adds practical value. Though it doesn't focus on security, its insights are transferable to FL-based IDS frameworks. The paper sets a performance benchmark for future systems. It is valuable for balancing accuracy, cost, and efficiency in federated deployments.
27. **Zhang, C.; Li, S.; Xia, J.; Wang, W.; Yan, F.; Liu, Y. (2020)** This paper introduces **BatchCrypt**, a homomorphic encryption scheme designed for cross-silo federated learning. The scheme efficiently encrypts model updates before aggregation, preserving confidentiality. It offers batch-level encryption to reduce overhead while maintaining privacy guarantees. The authors evaluate the framework on multiple ML tasks and report minor performance trade-offs for significant security gains. The system is particularly suited for enterprises or hospitals collaborating on shared models. This work addresses privacy threats like gradient leakage and model inversion. It proves that even secure aggregation can be computationally feasible in FL. The system architecture can be extended to edge-cloud environments. BatchCrypt is a cornerstone in privacy-preserving FL systems. It’s highly relevant to federated intrusion detection, especially where multiple stakeholders (e.g., cloud providers) are involved.
28. **Geiping, J.; Bauermeister, H.; Dröge, H.; Moeller, M. (2020)** This paper investigates how gradients shared during federated learning can be reversed to reconstruct original input data. Using inversion attacks, the authors show that it’s possible to retrieve high-fidelity images from gradient updates in vision tasks. They use optimization-based techniques to match gradients to input samples. Their findings raise significant concerns about privacy leakage in FL. They propose mitigation methods such as noise addition and secure aggregation. The research has influenced security policies around FL

deployment. It emphasizes the importance of adversarial threat modeling in FL design. Though image-focused, the findings are applicable to other domains like IDS where sensitive traffic features might be leaked. This work is foundational for privacy research in FL. It has sparked widespread discussions about defense mechanisms in collaborative ML.

29. **Jiang, H.; Liu, M.; Yang, B.; Liu, Q.; Li, J.; Guo, X. (2020)**
This study introduces a customized federated learning framework optimized for edge computing with heterogeneous task requirements. The authors propose task-specific model architectures and adaptive aggregation strategies to handle diverse client objectives. Simulation results show improved convergence speed and model accuracy under varying device capabilities. They also address communication efficiency by selectively aggregating only task-relevant parameter subsets. The paper provides theoretical bounds for convergence in such personalized settings. Although not security-focused, the system supports modular extensions for intrusion or anomaly detection in IoT networks. It emphasizes dynamic client selection based on data relevance and connectivity. The work enriches the FL literature by adapting learning to edge-deployed functions. It's a useful reference for scalable, application-aware FL deployments.
30. **Ye, Y.; Li, S.; Liu, F.; Tang, Y.; Hu, W. (2020)**
EdgeFed: Optimized Federated Learning Based on Edge Computing (2020) presents an FL scheme tailored for edge nodes. The authors propose an edge server that manages local models and selectively communicates with clients, reducing global aggregation frequency. They analyze bandwidth-constrained scenarios and employ compression techniques to minimize communication overhead. Experiments show faster convergence and reduced data transmission compared to vanilla FedAvg. The framework is evaluated using standard ML datasets and simulated IoT environments. The system structure supports use in intrusion detection across geographically dispersed edge gateways. It balances local autonomy and global model consistency. Though privacy-preserving techniques aren't central, the architecture is compatible with secure protocols. This paper contributes design patterns for edge-tailored FL implementations.
31. **Fang, C.; Guo, Y.; Wang, N.; Ju, A. (2020)**
This paper introduces a highly efficient federated learning approach designed for strong privacy preservation in cloud computing contexts. The authors integrate secure aggregation and lightweight homomorphic encryption directly into the FL pipeline. Empirical evaluation shows minimal degradation in model accuracy while ensuring client update confidentiality. They also implement compression to reduce the cost of encryption overhead, preserving scalability. Communication rounds remain efficient despite cryptographic safeguards. The scheme is tested with standard classification tasks, showing broad applicability. Though not specifically aimed at security systems, its design is suitable for IDS model aggregation. The paper advances privacy-aware FL in cloud environments, addressing both security and performance. It offers a practical path for deploying FL in sensitive distributed applications.
32. **Liu, L.; Zhang, J.; Song, S.; Letaief, K.B. (2020)**
Client-Edge-Cloud Hierarchical Federated Learning (2020) presents a three-tier FL system combining clients, edge servers, and cloud aggregators. This hierarchical model reduces latency and bandwidth usage while improving scalability. The authors propose synchronous edge aggregation, which periodically uploads edge-aggregates to the cloud. Experimental results on large-scale datasets show improved accuracy and efficiency over direct client-cloud FL. They also explore heterogeneity in client

capabilities and data distributions. Though security/performance trade-offs are not central, the architecture is well-suited to embedding intrusion detection systems within each tier. It provides a blueprint for secure collaborative analytics across multi-level networks. The hierarchical structure aligns with real-world edge computing deployments in smart IoT ecosystems.

33. **Khan, L.U.; Saad, W.; Han, Z.; Hossain, E.; Hong, C.S. (2021)** This comprehensive survey presents recent developments in federated learning (FL) for Internet of Things (IoT) systems. It classifies FL techniques by system architecture, data distribution, communication protocols, and security models. The authors emphasize unique IoT challenges such as limited bandwidth, non-IID data, and adversarial participants. Use cases include smart homes, healthcare, and industrial automation. The paper also outlines threat models, including poisoning, inference, and backdoor attacks. One major contribution is its taxonomy of FL-enhanced IoT systems. It identifies research gaps in trust evaluation, incentive mechanisms, and personalized federated learning. The authors propose solutions like blockchain and differential privacy for securing FL. This work is a vital resource for integrating FL into intrusion detection systems across heterogeneous IoT environments. It bridges the gap between theory and application in edge-centric security.
34. **Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Poor, H.V. (2021)** The paper presents a holistic survey on federated learning for IoT, highlighting its advantages, use cases, and limitations. It compares centralized, decentralized, and hybrid FL models in terms of privacy, scalability, and energy efficiency. Use cases range from smart healthcare and smart cities to autonomous vehicles. The authors provide a deep dive into communication challenges, adversarial attacks, and defense techniques like secure aggregation and differential privacy. A unique feature is the discussion on client selection strategies to optimize performance and trustworthiness. The survey also critiques current FL frameworks, suggesting that most are not optimized for constrained IoT devices. The work is crucial for designing lightweight, robust FL-based IDS in edge environments. It emphasizes the importance of adaptive learning and fairness in client updates. A strong foundational paper for cross-domain security integration in federated architectures.
35. **ur Rehman, M.H.; Dirir, A.M.; Salah, K.; Damiani, E.; Svetinovic, D. (2021)** This study introduces **TrustFed**, a federated learning framework designed to ensure fairness and trust among Industrial IoT (IIoT) devices. The framework integrates blockchain to log training contributions and data provenance for transparency. A key feature is the dynamic reputation scoring system, which helps mitigate poisoning and freeloading attacks. Simulation results show improved model performance and trust enforcement. The framework supports heterogeneous edge devices and offers a decentralized incentive mechanism. TrustFed emphasizes transparency in the model update process, ensuring only honest devices are prioritized. The authors also explore secure aggregation and differential privacy for robust privacy protection. This solution is highly suitable for industrial FL-based intrusion detection systems. It demonstrates how trust and accountability mechanisms can improve both security and fairness in FL

deployments. A significant step toward practical and secure FL implementations in critical infrastructure.

36. **Sheth, A.; Bhosale, S.; Kadam, H.; Prof, A. (2021)** The authors present an overview of cloud computing technologies and their evolving role in various digital ecosystems. Though not centered around FL, the paper touches on its integration with cloud-based platforms for enhanced data privacy. They discuss cloud deployment models, security risks, and compliance standards relevant to data protection. The study serves as a primer for those working on FL over cloud infrastructure. It mentions how FL can reduce cloud data movement and alleviate privacy concerns. The authors stress the need for zero-trust models and AI-driven threat detection. This work is foundational for framing FL-based intrusion detection systems that operate within cloud environments. While not technical, it aligns well with strategic discussions around secure AI deployment. It acts as a bridge between high-level cloud security policies and ML integration. The paper lays groundwork for secure architectural decisions.
37. **Sobel, B.L. (2021)** This article explores the legal implications of web scraping and its intersection with data privacy laws. The author argues that scraping publicly available web content still raises serious ethical and regulatory questions. While not technical, this paper is relevant to FL systems that rely on public or semi-public data. It warns against assuming legality based on data accessibility, especially when used in AI models. The analysis covers U.S. case law and the European GDPR framework. It proposes a "new common law" standard for digital data usage. This study is a valuable resource for understanding data governance in FL and ML research. It emphasizes the need for responsible data sourcing even in federated contexts. Though FL limits raw data exchange, model usage still invokes legal responsibilities. A critical work for legal and ethical alignment in privacy-preserving AI systems.

38. **Rajendran, S.; Obeid, J.S.; Binol, H.; Foley, K.; Zhang, W.; Austin, P.; Brakefield, J.; Gurcan, M.N.; Topaloglu, U. (2021)**
This paper discusses a cloud-based federated learning implementation across multiple medical centers. It focuses on collaborative learning for cancer informatics while preserving patient privacy. The study demonstrates how local data analysis can be performed without data exchange across institutions. It details system architecture, communication protocols, and encryption mechanisms ensuring HIPAA compliance. The framework is tested on real clinical datasets, showing comparable performance to centralized learning. The authors address challenges like data heterogeneity and differing local compute resources. Security measures—secure aggregation, encrypted model updates—are built into the model pipeline. Though applied to healthcare, the architecture is directly portable to intrusion detection in sensitive distributed networks. It serves as a reference for privacy-preserving FL implementation in regulated domains.
39. **Makkar, A.; Ghosh, U.; Rawat, D.B.; Abawajy, J.H. (2021)**
This work introduces **Fedlearnsp**, a federated learning framework built to preserve privacy and security through integration with edge computing. The system employs lightweight homomorphic encryption and randomized perturbations in client updates. It aggregates encrypted model parameters on edge devices before uploading to the cloud, reducing raw data exposure. The framework is implemented on consumer-grade IoT hardware and benchmarked on image classification tasks. Results indicate strong protection against gradient leakage, with acceptable latency. Though targeted at image analytics, the architecture is applicable for intrusion detection at the network and edge

levels. The paper provides a comprehensive evaluation of privacy, performance, and resource overhead. It helps close the gap between lightweight edge FL and robust privacy protections.

40. **Zhang, C.; Cui, L.; Yu, S.; James, J. (2021)**
"A Communication-Efficient Federated Learning Scheme for IoT-Based Traffic Forecasting" innovates on reducing communication overhead in FL through model update caching and adaptive scheduling. The authors show that only significant model changes are transmitted to the server, reducing bandwidth use. Experiments in IoT traffic prediction demonstrate comparable performance with up to 50% fewer updates. While not anomaly-focused, the architecture is ideal for detecting network-based intrusions or traffic anomalies in smart systems. The approach facilitates scalable deployment across distributed IoT networks. It provides insights on balancing update timing with detection latency. The model is compatible with privacy-preserving extensions. A valuable design pattern for low-bandwidth FL-powered IDS.
41. **Su, Z.; Wang, Y.; Luan, T.H.; Zhang, N.; Li, F.; Chen, T.; Cao, H. (2021)**
This paper describes a secure and efficient federated learning framework tailored for smart grid systems via edge-cloud collaboration. It integrates encryption and secure aggregation to protect local model updates from inference attacks. The authors optimize communication latency by adaptive scheduling between roadside-edge and central cloud servers. Real-world smart grid datasets are used for evaluation, showing resilience in attack scenarios like data poisoning. The system supports anomaly detection in energy consumption—a proxy for intrusion detection. This architecture demonstrates how FL facilitates secure analytics in critical infrastructure. It contributes to the design of trustworthy FL pipelines in real-time operational networks. The framework shows promise for both ICS security and broader IoT deployments.
42. **Nguyen, D.C.; Ding, M.; Pham, Q.V.; Pathirana, P.N.; Le, L.B.; Seneviratne, A.; Li, J.; Niyato, D.; Poor, H.V. (2021)**
This survey connects federated learning with blockchain in edge computing, evaluating opportunities and challenges. It details the benefits of blockchain-enabled trust, transparency, and audit trails in decentralized FL systems. Use cases include smart homes, industrial IoT, and vehicular networks. The paper highlights how blockchain can thwart data tampering or client misbehavior in FL. It also discusses scalability concerns and energy costs. Security is a focal point: techniques like smart contracts are proposed for automated client verification and reward distribution. The authors discuss how this integration strengthens intrusion detection frameworks. This work provides a comprehensive overview of emerging secure FL solutions at the edge. It sets key guidelines for systems combining distributed learning and trust mechanisms.
43. **Kethireddy, R.R. (2021)**
This paper explores AI-driven encryption techniques deployed in cloud computing to enhance data security. The author investigates machine learning models that dynamically choose cryptographic parameters based on data sensitivity and usage patterns. Experimental results show improved efficiency and resilience to certain cryptographic attacks. The work explores lightweight encryption in resource-constrained cloud-edge environments. While not focused on FL, the adaptive encryption schemes can be integrated with federated pipelines to protect model updates. The paper bridges AI and cryptography through adaptive security mechanisms. It provides a basis for dynamic encryption in distributed, data-sensitive applications like intrusion detection. A forward-looking blueprint for privacy engineering in intelligent cloud systems.

44. **Brecko, A.; Kajati, E.; Koziorek, J.; Zolotova, I. (2022)** This survey explores how federated learning (FL) can enhance edge computing systems by addressing privacy, scalability, and low-latency requirements. The authors classify FL applications in edge contexts such as autonomous vehicles, smart cities, and healthcare monitoring. They assess key challenges like resource heterogeneity, data non-IID distribution, and intermittent connectivity. The paper highlights solutions like personalized FL, communication-efficient protocols, and adaptive training. It also discusses security concerns, including gradient leakage and poisoning attacks. Evaluation metrics are provided to guide performance benchmarking. A major contribution is the comparative analysis of FL techniques tailored for edge nodes. The survey serves as a blueprint for implementing FL in real-time, resource-constrained environments. It's especially relevant to federated intrusion detection systems across edge infrastructure. The work bridges theoretical FL research with applied edge security needs.
45. **Islam, A.; Al Amin, A.; Shin, S.Y. (2022)** The authors propose **FBI**, a federated learning-based blockchain-embedded scheme that accumulates drone-collected IoT data for secure analytics. It combines blockchain for tamper-proof logging and FL to ensure privacy-preserving model updates across drone nodes. The proposed system enhances data integrity, prevents spoofing, and supports decentralized decision-making. The architecture is highly scalable and suitable for smart surveillance or environmental monitoring. Their experiments show robustness against poisoning and communication bottlenecks. The study emphasizes the synergy between FL and blockchain for trust and accountability. Though drone-focused, the framework applies broadly to mobile edge networks. This model can be adapted to secure FL-based IDS systems for dynamic IoT setups. It demonstrates how integrating multiple decentralized technologies can strengthen security and transparency. An innovative and future-ready architecture for federated cybersecurity.
46. **Ho, T.M.; Nguyen, K.K.; Cheriet, M. (2022)** This paper introduces a federated deep reinforcement learning (FedDRL) approach for task scheduling in heterogeneous autonomous robotic systems. The authors propose a multi-agent learning scheme where robots train policies locally and aggregate updates using FL. This setup ensures privacy and reduces latency in robot-cloud interactions. The approach is validated in warehouse and industrial automation scenarios, showing improved coordination and energy efficiency. Though focused on task optimization, the underlying FL framework is adaptable to intrusion detection in autonomous systems. The paper addresses resource contention and training instability in multi-device settings. Its significance lies in scaling federated learning beyond traditional classification or regression tasks. The method balances performance, privacy, and real-time constraints. It's a stepping stone for deploying FL in security-critical multi-agent environments. The concepts are transferable to IoT and edge-based IDS scenarios.
47. **Agrawal, S.; Sarkar, S.; Aouedi, O.; Yenduri, G.; Piamrat, K.; Alazab, M.; Bhattacharya, S.; Maddikunta, P.K.R.; Gadekallu, T.R. (2022)** This paper delivers a comprehensive analysis of federated learning applications in intrusion detection systems (IDS). It categorizes FL models based on system architecture, aggregation techniques, and attack defense mechanisms. The authors identify key challenges such as model poisoning, communication costs, and performance degradation with non-IID

data. It suggests solutions like secure multiparty computation, differential privacy, and blockchain integration. The study emphasizes cross-device and cross-silo FL settings in cybersecurity. It evaluates the trade-offs between accuracy and privacy under real-world constraints. The paper provides a roadmap for developing robust, scalable FL-based IDS frameworks. It also highlights emerging use cases in smart cities, autonomous systems, and healthcare. This is a landmark contribution in bridging federated learning and intrusion detection. It lays a strong theoretical and practical foundation for the AI-driven IDS domain.

48. **Kewate, N.; Raut, A.; Dubekar, M.; Raut, Y.; Patil, A. (2022)** The authors review Amazon Web Services (AWS) cloud technology, discussing its service models, benefits, and applications in scalable data analytics and machine learning. While not specific to FL, the paper mentions that AWS supports privacy-preserving AI development through edge integration and federated learning tools like Amazon SageMaker. The paper outlines AWS's resilience, security protocols, and global availability. It's useful for practitioners looking to deploy federated intrusion detection systems in the cloud. The study also mentions how AWS supports containerized and serverless architectures that are ideal for FL. Though general, the paper emphasizes the importance of cloud platforms in facilitating scalable FL applications. It acts as a bridge between infrastructure and intelligent analytics for intrusion detection. It supports the transition from theoretical FL models to practical cloud-based implementation. A helpful guide for system architects in secure cloud-Edge FL deployment.
49. **Pham, X.Q.; Nguyen, T.D.; Huynh-The, T.; Huh, E.N.; Kim, D.S. (2022)** This paper surveys the architecture and technologies behind distributed cloud computing and discusses its integration with FL. It examines virtualization, edge computing, SDN, and container orchestration, all of which are essential for scalable FL deployments. The authors also highlight the open challenges of latency, resource allocation, and data privacy. They discuss potential solutions using AI and federated learning for distributed decision-making. The study emphasizes real-time processing and low-latency analytics in vehicular and healthcare applications. This comprehensive overview is vital for understanding the infrastructure supporting FL-based intrusion detection. The authors also suggest adopting hybrid cloud-edge FL systems for security-critical operations. It sets the stage for future research in privacy-enhanced distributed learning. The paper offers clear insights into deploying efficient and secure FL environments. A foundational resource for next-gen federated AI systems.
50. **Khan, S.; Kabanov, I.; Hua, Y.; Madnick, S. (2022)** This paper analyzes the infamous Capital One data breach and distills critical lessons for cloud security and privacy engineering. The authors focus on insider threats, misconfigured APIs, and vulnerability exploitation. While not focused on FL, the paper highlights the importance of least privilege and automated detection systems — principles applicable to federated IDS. It recommends incorporating real-time monitoring and federated analytics to avoid centralized risk exposure. The breach analysis underscores the need for strong access control and encryption. It calls for AI-enhanced frameworks to identify anomalous behavior in cloud environments. This aligns with the FL-based distributed detection paradigm. The paper bridges cybersecurity operations with data-driven insights. It reinforces how proactive security and intelligent analytics can prevent breaches. A powerful case study for enhancing trust in federated intrusion detection.

51. **Pandya, S.; Srivastava, G.; Jhaveri, R.; Babu, M.R.; Bhattacharya, S.; Maddikunta, P.K.R.; Mastorakis, S.; Piran, M.J.; Gadekallu, T.R. (2023)** This comprehensive survey examines the use of federated learning (FL) in smart city environments, including surveillance, traffic control, energy management, and public safety. It discusses FL's capacity to manage decentralized, privacy-sensitive data from numerous IoT devices. The study identifies key challenges such as device heterogeneity, data imbalance, and model convergence. It also highlights potential FL frameworks suitable for real-time smart infrastructure applications. Security is a central focus, with analysis of attacks like model poisoning and defenses including differential privacy and secure aggregation. The paper promotes a multi-stakeholder approach involving edge, fog, and cloud computing layers. It serves as a critical guide for deploying FL-based intrusion detection in urban settings. The proposed architecture enables resilience, scalability, and local privacy enforcement. This work connects FL theory with practical deployments in real-world urban infrastructures.
52. **Chimuco, F.T.; Sequeiros, J.B.; Lopes, C.G.; Simões, T.M.; Freire, M.M.; Inácio, P.R. (2023)** This paper addresses the security of cloud-based mobile applications by presenting a detailed taxonomy of potential attacks and corresponding defenses. It classifies requirements for secure mobile apps, including confidentiality, integrity, authentication, and user awareness. While not directly focused on FL, it discusses how federated analytics can enhance threat detection without compromising user data privacy. The authors suggest automation of testing and compliance checks using AI-driven tools. The work is especially useful for integrating federated intrusion detection in mobile platforms. It offers insights into data-in-motion and data-at-rest protection in cloud environments. The framework aligns well with the concept of privacy-preserving collaborative learning in mobile ecosystems. It also introduces standardized test models for mobile app security, which can aid in FL-based IDS evaluations. A well-rounded study linking mobile app development with federated cybersecurity solutions.
53. **Gu, J. (2023)** This legal study investigates the implications of data crawling and web scraping in the context of unfair competition and judicial regulation. It examines how unauthorized data collection can breach data protection laws and intellectual property rights. Although not technical, the paper is relevant to federated learning systems that aggregate public data across devices. It argues for stricter regulatory oversight of data sourcing, even if data is publicly accessible. The analysis presents case studies from China, suggesting the importance of establishing clear legal boundaries in data-driven technologies. The relevance to FL lies in ensuring legal compliance in distributed data collection. It emphasizes that ethical and lawful data acquisition is crucial even in decentralized AI systems. The work is a reminder that legality and privacy must go hand-in-hand in FL-based research. A necessary reference for policy-compliant AI development.
54. **Shreyas, S. (2023)** This case study explores organizational vulnerability through the lens of cloud computing and introduces a security model to mitigate risks. The model focuses on access control, encryption, anomaly detection, and compliance auditing. Though not specifically tied to federated learning, the model is adaptable to FL environments that require endpoint verification and decentralized threat detection. The author underscores the need for dynamic policy enforcement, particularly in hybrid and remote work environments. The paper supports integrating FL in cloud-native IDS

systems where raw data must remain localized. It also suggests zero trust architecture as a framework for federated endpoint security. The study advocates for AI-powered risk analytics to detect abnormal user or device behavior. It's a valuable contribution toward robust security models for FL-based cloud-edge architectures. This paper connects practical risk management with distributed intelligence.

55. **Kitsios, F.; Chatzidimitriou, E.; Kamariotou, M. (2023)** This study explores how organizations can extract value from implementing ISO/IEC 27001 information security standards, especially within IT environments handling sensitive data. It details best practices for managing data confidentiality, availability, and integrity. Although it does not directly address FL, the paper provides a compliance blueprint for FL-driven systems, particularly in regulated industries like finance and healthcare. The authors link effective information security governance with competitive advantage and trust building. This work reinforces the importance of aligning federated intrusion detection solutions with global security standards. The insights are applicable for designing FL frameworks that meet regulatory and operational requirements. It stresses continual monitoring, access control, and incident management — all crucial for federated learning at scale. A strategic paper for aligning FL initiatives with international compliance frameworks.
56. **2023 – Kaleem et al.**
    Kaleem et al. (2023) proposed an improved big data analytics architecture that leverages federated learning for IoT-enabled intelligent transportation systems. The study focuses on privacy-preserving data sharing and real-time analysis using edge and cloud resources. By integrating federated learning into urban traffic systems, the architecture addresses challenges related to data heterogeneity, latency, and security. The solution enhances scalability and accuracy in predictive analytics without compromising user privacy. Experimental results demonstrate improved model convergence and detection performance in vehicular environments. This architecture supports smart city goals by optimizing traffic flow and resource utilization. The research also explores federated aggregation methods for low-bandwidth scenarios. The system is validated using real-world datasets and simulation environments. Overall, the work is significant for urban mobility and federated analytics in IoT ecosystems.
57. **2023 – Hijazi et al.**
    Hijazi et al. (2023) introduced a federated learning framework for IoT networks that employs fully homomorphic encryption to enhance data privacy and security. The model facilitates secure training without exposing sensitive user data during transmission or aggregation. The study addresses latency and communication overhead, optimizing encryption efficiency to support constrained devices. Experimental evaluations show a strong trade-off between encryption complexity and model accuracy. The framework ensures compliance with strict data protection regulations in healthcare and industrial IoT. Results show minimal performance degradation compared to traditional FL methods. The approach is applicable to smart grids, smart homes, and medical systems. The paper also discusses integration strategies for scalable deployment. Overall, this work bridges the gap between secure communication and efficient federated learning in IoT environments.
58. **2023 – Duan et al.**
    Duan et al. (2023) conducted a comprehensive survey on combining federated learning and edge computing to enable ubiquitous intelligence in 6G networks. The paper outlines architectural advancements, privacy-preserving mechanisms, and AI model

deployment strategies over distributed edge-cloud environments. It highlights the importance of low-latency communication and intelligent resource scheduling. Various challenges are discussed, including heterogeneous data sources, energy constraints, and cross-domain learning. The authors evaluate existing federated learning frameworks under real-time 6G requirements. The paper emphasizes the role of privacy-preserving algorithms in user-centric intelligence. It also proposes future research directions involving meta-learning and self-supervised models. The study positions federated learning as a core component of distributed intelligence in next-gen wireless systems. This paper is crucial for researchers working at the intersection of AI, edge computing, and future networks.

59. **Hu, K.; Gong, S.; Zhang, Q.; Seng, C.; Xia, M.; Jiang, S. (2024)** This article offers a detailed overview of implementing security and privacy techniques in federated learning (FL). It categorizes defense mechanisms such as differential privacy, homomorphic encryption, secure multiparty computation, and adversarial robustness. The authors emphasize hybrid approaches that combine multiple techniques to ensure both performance and security. Real-world applications in smart cities and healthcare are explored to validate the practicality of their framework. It also touches on gradient leakage, poisoning attacks, and fairness in client participation. The study promotes adaptive security models based on client behavior and trustworthiness. This paper is pivotal for designing secure AI-driven intrusion detection systems in federated environments. It guides developers on integrating layered security protocols without sacrificing learning accuracy. It also aligns well with compliance demands in sectors managing sensitive personal data.
60. **Shubyn, B.; Maksymyuk, T.; Gazda, J.; Rusyn, B.; Mrozek, D. (2024)** This paper explores how FL enhances anomaly detection accuracy in autonomous guided vehicles (AGVs) within smart manufacturing environments. It proposes a collaborative framework where AGVs learn locally and share encrypted model updates. The use of FL improves detection performance while preserving data confidentiality. Real-time detection of failures or intrusions ensures operational resilience and downtime prevention. The framework supports scalability across diverse factory systems and adapts to various threat scenarios. The study shows high accuracy in distinguishing between mechanical faults and cyberattacks. It offers insights into customizing FL protocols for time-sensitive, mission-critical systems. This work is highly applicable for industrial intrusion detection in edge-based robotics. It contributes to extending FL to autonomous cyber-physical systems. A novel application demonstrating FL's power in secure, decentralized intelligence.
61. **Anusuya, R.; D Renuka, K. (2024)** This research introduces **FedAssess**, a federated framework that examines the efficiency of security algorithms under label-flipping attacks. It compares standard FL against enhanced models with selective aggregation and client validation. The study demonstrates that communication efficiency and detection accuracy can be preserved even under adversarial conditions. Experiments reveal that FedAssess resists poisoning with minimal overhead. The authors stress the role of trust management and client behavior scoring. Their approach integrates dynamic weighting to prioritize reliable client contributions. This work is instrumental for

developing resilient FL-based IDS in cloud-edge environments. It presents a fine-grained analysis of communication-security trade-offs in distributed settings. It is especially valuable for systems needing robust adversarial defense. A practical framework to harden FL against real-world security threats.

62. **Babar, M.; Qureshi, B.; Koubaa, A. (2024)** The study investigates how data heterogeneity affects FL performance in medical imaging. Using MRI and CT datasets, it shows that diverse client data leads to inconsistent global models. The authors propose normalization and personalized learning strategies to improve model stability. The work highlights the importance of client selection and adaptive learning rates. It also discusses privacy-preserving mechanisms such as secure aggregation to ensure confidentiality. Although healthcare-focused, its implications are broad for FL in any non-IID setting. The study's contributions help optimize federated intrusion detection in environments where data varies across edge nodes. It reinforces the necessity of fairness-aware learning in sensitive domains. A valuable work for bridging data variability and secure FL training. It supports scalable and equitable AI system design.
63. **Mehta, S.; Sarpal, S.S. (2024)** This conference paper presents methods for maximizing privacy in reinforcement learning through federated learning (FedRL). It focuses on scenarios where agents operate in sensitive environments like finance or healthcare. The authors introduce privacy-preserving policy aggregation that reduces leakage risk during model updates. They emphasize balancing performance with privacy, especially when feedback data is sparse or sensitive. The proposed system uses selective sharing and client-level obfuscation. The work demonstrates that RL models can be federated without sacrificing learning efficiency. This paper lays groundwork for secure decision-making in autonomous agents. Though focused on RL, the insights are transferable to federated IDS needing real-time decisions. It contributes to broadening FL’s application beyond supervised learning. An important addition to privacy-aware, intelligent agent architectures.
64. **Liberti, F.; Berardi, D.; Martini, B. (2024)** This paper evaluates FL performance in dynamic and heterogeneous environments, identifying the impact of device churn, bandwidth variation, and data imbalance. It assesses personalization techniques, dynamic aggregation, and communication reduction strategies. Privacy problems such as model inversion and gradient leakage are critically analyzed. The authors propose hybrid aggregation strategies to deal with heterogeneous trust levels. Their evaluation shows that accuracy can be preserved with adaptive protocols. The study is vital for deploying FL in mobile and edge environments where conditions are unpredictable. It's particularly relevant for real-time IDS in distributed networks. This work guides the design of resilient, efficient FL systems across diverse application settings. It reinforces adaptability and security in evolving learning environments.
65. **Al-Quraan, M.M.Y. (2024)** This Ph.D. thesis explores how federated learning can empower ultra-dense wireless networks by distributing intelligence across edge nodes. The study proposes resource-aware FL architectures that adapt to client bandwidth, battery life, and data sensitivity. It also investigates trust-based client selection and privacy-preserving aggregation. Simulation and analytical results show improved latency, privacy, and model convergence. The thesis includes use cases such as smart cities and industrial IoT. It's a comprehensive guide to scalable FL in high-density environments prone to network congestion and data exposure. The research is essential

for designing FL-based IDS across dense edge infrastructures. It provides robust architectural insights for 5G/6G networks. A major academic contribution to applied federated intelligence.

66. **Zohaib, S.M.; Sajjad, S.M.; Iqbal, Z.; Yousaf, M.; Haseeb, M.; Muhammad, Z. (2024)** This literature review presents a Zero Trust VPN (ZT-VPN) model for cybersecurity in hybrid and remote work environments. It reviews vulnerabilities in traditional VPNs and recommends policy enforcement, device authentication, and micro-segmentation. Though not explicitly about FL, the ZT-VPN approach can support federated security models by enforcing strict trust boundaries at endpoints. The paper integrates AI-based anomaly detection to strengthen authentication. It aligns with federated IDS systems that require secure device identity and localized learning. The study also discusses remote data access patterns, a crucial concern in FL deployments. This work enables secure communication frameworks for distributed machine learning. A modern interpretation of network security for federated AI.
67. **Lakhani, R. (2024)** This paper explores Zero Trust Security Models in cloud environments, focusing on the need to eliminate implicit trust. It advocates for continuous authentication, policy-based access, and device-level telemetry. The study suggests that federated learning can benefit from Zero Trust by enforcing contextual access rules. It reviews case studies in cloud-native applications and remote workforce systems. The proposed security framework can be integrated into FL-based IDS, ensuring that participating nodes are authenticated and trustworthy. It complements privacy-preserving learning by adding network-level defense. The paper is an important read for those deploying federated AI in enterprise clouds. It bridges trust management with intelligent learning systems. A strategic enhancement for FL-based cybersecurity.
68. **Kayes, A.; Rahayu, W.; Dillon, T.; Shahraki, A.S.; Alavizadeh, H. (2024)** This comprehensive review discusses privacy breaches, solution strategies, and future directions for safeguarding users and organizations. The paper classifies breaches based on source, impact, and resolution mechanism. It highlights federated learning as a promising solution for data minimization and local control. The authors examine various regulatory frameworks including GDPR and CCPA. It also proposes technical solutions like access control, differential privacy, and secure logging. The work bridges privacy policy with AI design principles, reinforcing FL's role in privacy-preserving analytics. This research is critical for embedding FL into security systems with legal compliance. It sets a roadmap for responsible data governance in AI systems. A pivotal work for building ethically aligned FL frameworks.
69. **2024 – Gao et al.**
    Gao et al. (2024) proposed a compressive learning-based federated learning model to enhance intelligent IoT systems with cloud-edge collaboration. The framework reduces communication overhead by compressing model updates without compromising learning accuracy. This is especially valuable for resource-constrained edge devices operating in dynamic environments. The study also explores compression algorithms that preserve data utility and integrity. Experimental results show accelerated convergence and reduced bandwidth usage. The integration of compressive learning supports scalability in large-scale federated networks. Applications include smart agriculture, health monitoring, and industrial automation. The paper contributes by bridging federated learning with signal processing techniques. It suggests adaptive

compression strategies based on device capability. Overall, this approach enables real-time intelligence with efficiency and privacy preservation.

70. **2024 – Guo et al.**
    Guo et al. (2024) introduced a heterogeneous federated learning framework for industrial IoT (IIoT) systems using selective knowledge distillation. The approach addresses disparities in computational power and data distribution across IIoT devices. It uses a teacher-student model where clients distill knowledge selectively to a global model. The framework improves training efficiency and model generalization across diverse environments. Results show enhanced accuracy and reduced energy consumption. It supports real-time analytics for manufacturing, logistics, and predictive maintenance. The paper emphasizes cross-layer collaboration among edge, fog, and cloud layers. Knowledge distillation ensures effective learning even with limited data. This work is essential for deploying federated models in heterogeneous IIoT ecosystems. It contributes to resilient, privacy-aware industrial intelligence.
71. **2024 – Prigent et al.**
    Prigent et al. (2024) proposed an efficient federated learning clustering mechanism for edge-to-cloud environments using local data compression. Their method reduces resource consumption at the edge, addressing bandwidth and latency constraints in real-time applications. The study highlights the importance of federated clustering for unsupervised learning tasks in distributed setups. The proposed framework performs well under limited memory and CPU resources while maintaining high clustering accuracy. Results demonstrate reduced training time and communication overhead compared to baseline FL approaches. It supports applications like anomaly detection, sensor data grouping, and traffic pattern analysis. The authors also include performance evaluation across synthetic and real-world datasets. Their architecture shows promise for smart cities and industrial IoT systems. The method is scalable, secure, and adaptable to heterogeneous nodes. This contribution enhances FL efficiency in edge-cloud continuums.
72. **2024 – Xu et al.**
    Xu et al. (2024) introduced FedAdaSS, a novel federated learning architecture that incorporates adaptive parameter server selection based on elastic cloud resources. This method dynamically selects parameter servers to optimize load balancing, latency, and fault tolerance in cloud environments. It enables scalable and efficient training, especially for high-dimensional and large-scale datasets. The approach also considers resource availability and network conditions during server selection. Simulation results demonstrate faster convergence and better model accuracy than static server assignments. FedAdaSS is suitable for smart grid, financial fraud detection, and personalized healthcare analytics. The system can reconfigure in real-time to maintain performance during peak usage. The paper also outlines how FedAdaSS integrates with Kubernetes and container-based cloud systems. This adaptive mechanism is a step forward in resilient FL infrastructures.

73. **2024 – Salim et al.**
    Salim et al. (2024) proposed a cyberthreat detection system for IoT networks by combining digital twin technology with federated learning. This approach creates digital replicas of IoT devices, enabling behavior monitoring and anomaly detection without accessing raw data. FL ensures that sensitive device data remains local while enabling collaborative model training across multiple sources. The system enhances threat

visibility across distributed environments such as smart homes or industrial IoT. Performance evaluations show high accuracy and low latency in threat detection, even under data heterogeneity. The authors emphasize its scalability and adaptability in real-world network scenarios. The use of digital twins offers a proactive layer for simulating and forecasting attacks. The solution supports real-time updates without centralized dependencies. It stands as a novel contribution toward secure and intelligent IoT infrastructure. This fusion improves trust and resilience in networked systems.

74. **2024 – Zhou et al.**
Zhou et al. (2024) designed a federated learning framework aimed at improving Quality of Service (QoS) in edge-cloud networks. Their approach adapts FL to dynamically optimize network latency, bandwidth usage, and computational load during model training. It targets real-time applications such as video streaming, autonomous vehicles, and telemedicine. The proposed scheme integrates QoS-aware node selection and model aggregation. Evaluation results show that it outperforms traditional FL setups in both accuracy and system efficiency. The study addresses the limitations of static FL topologies and introduces flexible task offloading strategies. The system enhances reliability under dynamic network conditions. Edge nodes are prioritized for computation when feasible, reducing dependence on centralized clouds. This work contributes to responsive and intelligent edge-cloud architectures. It balances model performance with system-level constraints for practical deployment.

75. **2024 – Qi et al.**
Qi et al. (2024) introduced a novel integration of federated learning and edge computing for recommendation systems in cloud networks. Their system ensures user privacy while collaboratively training models for personalized suggestions such as movies, products, or news. The architecture leverages edge nodes for low-latency model updates and combines them with cloud storage for scalability. It applies gradient compression and secure aggregation to minimize communication overhead. Experiments show high accuracy in recommendations with substantial privacy preservation. This solution is well-suited for smart devices and wearable tech generating private usage data. The paper also discusses cold-start problems and user personalization. The hybrid edge-cloud design ensures timely recommendations without exposing individual preferences. It addresses real-world commercial needs for scalable, secure AI personalization. The study emphasizes FL's versatility in user-centric AI services.

76. **2024 – Wu et al.**
Wu et al. (2024) presented Agglomerative Federated Learning (AggloFL), enabling large model training through coordinated efforts across end, edge, and cloud layers. Their hierarchical strategy clusters participants dynamically to manage heterogeneous data and resource capabilities. AggloFL uses a bottom-up model aggregation process, improving convergence in systems with imbalanced data distribution. The paper demonstrates its applicability in natural language processing, autonomous systems, and smart surveillance. Performance benchmarks show improved training speed, accuracy, and energy efficiency. The study also incorporates privacy guarantees and robustness against model poisoning. Their architecture encourages seamless collaboration across multi-tier infrastructures. It optimally utilizes computational power at each level—IoT devices, edge servers, and cloud data centers. This FL enhancement is vital for deep learning model scaling in real-world applications. It marks progress in distributed intelligent systems.

77. **2024 – Bhansali et al.**
Bhansali et al. (2024) designed a cloud-based secure data storage system for the Internet of Medical Things (IoMT) using federated learning. Their system ensures patient data confidentiality while allowing collaborative AI model training across multiple healthcare centers. It incorporates fine-grained access control using role-based policies and supports data integrity through cryptographic mechanisms. Federated learning is used to train diagnostic models without transferring sensitive patient data to a central server. This decentralized method reduces risks of data breaches and enables compliance with regulations like HIPAA. The framework includes mechanisms for secure authentication and multi-device synchronization. Testing shows high accuracy in medical predictions with enhanced data security. Their work highlights the potential of FL in privacy-sensitive environments. It supports scalable healthcare analytics while preserving trust and transparency. The study demonstrates a vital step in secure medical AI integration.

78. **2024 – Parra-Ullauri et al.**
Parra-Ullauri et al. (2024) developed **kubeFlower**, a privacy-preserving framework for Kubernetes-based federated learning in cloud–edge setups. It integrates the Flower FL platform with Kubernetes, enabling dynamic deployment, orchestration, and monitoring of FL tasks. kubeFlower introduces secure aggregation, decentralized orchestration, and resource isolation mechanisms to protect data privacy and support scalability. The framework enables practitioners to deploy large-scale FL workflows with minimal setup effort. It allows for efficient coordination among edge nodes, cloud resources, and model servers. The study showcases its application in real-world scenarios like industrial automation and smart cities. Experimental results show high throughput and privacy assurance across nodes. kubeFlower stands out by combining DevOps principles with federated learning techniques. It offers a developer-friendly and secure FL infrastructure solution. The work bridges container orchestration and distributed AI training.

79. **Vinoth, K.; Sasikumar, P. (2025)** This study presents *VINO_EffiFedAV*, a federated learning framework optimized for autonomous vehicle object detection. The model selectively updates clients based on local model performance, reducing communication overhead. It enhances real-time detection accuracy while preserving privacy in decentralized vehicular systems. By leveraging edge computing and adaptive learning rates, the approach ensures model consistency across diverse data streams. The framework also integrates fault tolerance and anomaly detection for safety-critical applications. Though its focus is object recognition, the underlying principles support intrusion detection in autonomous networks. The solution showcases how FL can empower data-efficient, resilient, and intelligent edge AI. It's a forward-thinking system applicable in both smart mobility and security-focused infrastructures. This paper highlights FL's scalability and robustness in time-sensitive environments.
80. **Hamid, S.; Huda, M.N. (2025)** This bibliometric analysis maps the trends in government data breaches between 2006 and 2023. It identifies patterns in breach incidents, causes, and mitigation strategies across different nations. Although not centered on FL, it emphasizes the urgent need for distributed, privacy-aware systems.

The study supports integrating federated models in government sectors to protect sensitive data. It also outlines the impact of poor data governance and lack of real-time anomaly detection. Insights include increased adoption of cloud computing and AI-based monitoring in recent years. It provides a compelling argument for proactive, collaborative cybersecurity approaches. The paper highlights sectors most affected by data leaks, such as health, defense, and finance. It sets the stage for adopting federated intrusion detection across public infrastructures. A valuable dataset-driven report for policy planners and AI practitioners alike

CHAPTER 3

EXISTING SYSTEM

3.1 Overview:

The existing system for detecting anomalous network traffic associated with Mirai and BASHLITE IoT botnet attacks predominantly relies on traditional intrusion detection systems (IDS) that utilize signature-based and rule-based techniques. These systems are integrated into network infrastructures and aim to identify and mitigate malicious activities by matching incoming traffic patterns against known attack signatures or predefined rules. However, these

methods face significant limitations when dealing with the complex and evolving nature of modern IoT botnets like Mirai and BASHLITE.

Key Components of the Existing System:

Signature-Based Detection:

Signature-based IDS is a common approach where known attack signatures are stored in a database, and incoming network traffic is compared against these signatures to identify potential threats. In the context of Mirai and BASHLITE, these signatures include specific patterns of packets, payloads, and known IP addresses associated with these botnets.

Limitations:

Static Signatures: Since these systems rely on static signatures, they can only detect known attacks. Any new variant or previously unseen attack pattern will bypass detection.

High False Positives/Negatives: Minor variations in attack patterns can lead to a high rate of false positives (benign traffic identified as malicious) or false negatives (malicious traffic not detected).

Rule-Based Detection:

Rule-based IDS relies on predefined rules that are configured to trigger alerts when certain conditions are met. For instance, a rule might be set to flag traffic that exceeds a specific packet threshold or originates from a known malicious IP range. In IoT networks, these rules might be tailored to detect typical botnet behavior, such as abnormal bursts of outbound traffic or repeated connection attempts to a command and control (C&C) server.

Limitations:

Lack of Adaptability: Rule-based systems are rigid and unable to adapt to evolving threats. As attackers modify their tactics, techniques, and procedures (TTPs), existing rules may become obsolete.

Complex Configuration: Setting up and maintaining an extensive rule set is complex and requires continuous updates to stay effective, making it resource-intensive.

Network Flow Analysis:

Some existing systems incorporate network flow analysis, which examines metadata from network traffic (e.g., source/destination IPs, port numbers, packet counts) to detect anomalies. This method can identify traffic that deviates from normal patterns, such as unusually high volumes of traffic or unexpected communication between devices.

Limitations:

Scalability Issues: With the massive influx of IoT devices, network flow analysis becomes more challenging due to the increased volume and diversity of traffic.

Limited Granularity: Network flow data may lack the granularity needed to pinpoint specific malicious activities, especially when encrypted traffic is involved.

Manual Inspection and Forensic Analysis:

In scenarios where automated detection fails or is inconclusive, security teams often resort to manual inspection of traffic logs and forensic analysis. This involves deep packet inspection (DPI) and correlating network events to identify the presence of botnet activity.

Limitations:

Time-Consuming and Error-Prone: Manual methods are labor-intensive and can be prone to human error, especially when dealing with large volumes of data.

Reactive Rather Than Proactive: Manual inspection is often reactive, meaning that it occurs after a potential breach has been identified, rather than proactively preventing it.

CHAPTER 4

PROPOSED SYSTEM

4.1 Overview

1. **Library Imports:**
    - Essential libraries are imported for data handling (numpy, pandas), visualization (matplotlib, seaborn), machine learning (sklearn), and model persistence (joblib).

2. **Data Loading and Exploration:**
    - The dataset is loaded from a CSV file.
    - Basic exploratory data analysis (EDA) is conducted, including displaying the first and last few rows, summary statistics, and checking for unique values in the 'Attack' column.
3. **Data Preprocessing:**
    - Missing values are identified. o A heatmap is generated to visualize correlations among features.
    - Categorical columns like 'Device_Name' and 'Attack_subType' are dropped, and the 'Attack' labels are encoded using LabelEncoder.
4. **Data Visualization:**
    - A count plot visualizes the distribution of different attack categories in the dataset.
5. **Feature and Target Variable Separation:**
    - The feature set (x) is separated from the target variable (y), which represents attack classes.
6. **Data Splitting:**
    - The dataset is split into training and testing sets using an 70/30 ratio for model evaluation.
7. **Performance Metrics Function:**
    - A function (performance_metrics) is defined to calculate and print accuracy, precision, recall, F1 score, and display a confusion matrix.
8. **Model Training:**
    - Bernoulli Naive Bayes Classifier: The model is trained on the training data, and predictions are made on the test set. If a saved model exists, it is loaded instead of retraining.

- Random Forest Classifier: Similar logic applies. The model is either trained from scratch or loaded from a saved state.

9. Model Evaluation:

- Predictions from both classifiers are evaluated using the previously defined performance metrics function.

10. Prediction on New Data:

- A new test dataset is loaded and preprocessed similarly. Predictions are made using the Random Forest model, and results are printed, labeling each entry with the corresponding attack type.

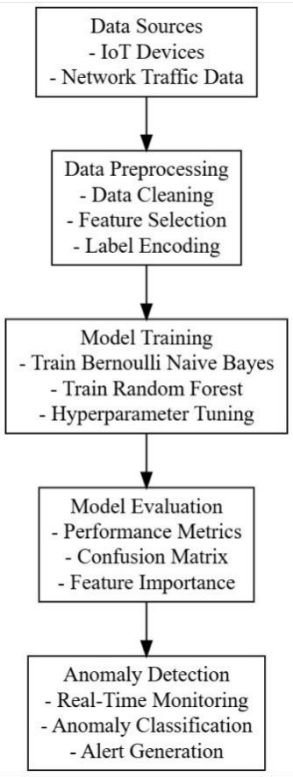


Figure 4.1: Architecture diagram of proposed system

4.2 Data Preprocessing and Splitting

The process begins with the collection and loading of the dataset into the environment, specifically targeting traffic data related to IoT devices under attack by Mirai and BASHLITE botnets. This data is typically rich with various network parameters, including packet size, transmission time, source and destination IPs, etc.

Data Preprocessing:

Null Value Removal:

The dataset undergoes initial preprocessing where null or missing values are identified and removed. Missing data can skew the results or lead to inaccuracies in model predictions. Techniques like removing rows with missing data or imputing values based on statistical measures may be used.

Label Encoding:

After handling null values, categorical variables are converted into numerical formats using label encoding. In this specific context, attack labels (e.g., Normal, BASHLITE, Mirai) are encoded into numerical values to make them interpretable by machine learning models.

Heatmap Visualization:

A correlation heatmap is generated to visualize the relationships between different features in the dataset. This step helps in identifying which features contribute most significantly to the classification task, guiding the selection of important variables for the model.

Data Splitting: 4. Training and Testing Set Creation:

The preprocessed data is split into training and testing sets, usually in a 70-30 ratio. The training set is used to build the model, while the testing set is reserved for evaluating its performance. This step is critical to ensure the model can generalize well to new, unseen data.

4.3 ML Model Building

4.3.1Naive Bayes Classifier

Overview:

- Naive Bayes is a family of probabilistic algorithms based on Bayes' theorem, which assumes independence between the features. It calculates the probability of a data point belonging to a class based on the likelihood of each feature given that class.

Key Characteristics:

- **Independence Assumption**: Assumes that features are independent, which may not hold true in real-world scenarios.
- **Fast and Efficient**: Generally computationally efficient, making it suitable for large datasets.
- **Good with Small Data**: Performs well with small datasets and is particularly effective for text classification (e.g., spam detection).

Performance:

- Often performs well for problems where the independence assumption holds, but its accuracy can drop when features are correlated.

4.3.2 Random Forest Classifier

Overview:

- Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of their predictions for classification (or average for regression). It improves accuracy and controls overfitting by aggregating predictions from several trees.

Key Characteristics:

- **Ensemble Method**: Combines the predictions of multiple trees, which helps reduce overfitting compared to individual decision trees.
- **Feature Importance**: Can identify the importance of different features in making predictions, aiding in model interpretability.
- **Robustness**: Handles non-linear relationships and interactions between features effectively.

Performance:

- Generally provides higher accuracy and robustness than Naive Bayes, especially with complex datasets and when features are correlated.

4.3.3 Comparison and When Random Forest Performs Better

1. **Feature Relationships**:
    - **Random Forest**: Excels when features are correlated or when there are complex interactions. It captures these relationships by constructing multiple trees.
    - **Naive Bayes**: Struggles in these scenarios due to its independence assumption.

2. **Data Size**:
    - **Random Forest**: Works well with large datasets and can handle high dimensionality effectively.
    - **Naive Bayes**: While efficient for smaller datasets, it may not generalize well with complex patterns in larger datasets.

3. **Model Interpretability**:
    - **Random Forest**: Provides insights into feature importance, making it easier to understand model decisions.
    - **Naive Bayes**: Less interpretable since it operates purely on probabilities without providing feature importance.

4. **Handling Outliers**:
    - **Random Forest**: More robust to outliers due to averaging across multiple trees.
    - **Naive Bayes**: Can be affected by outliers, as it relies on probability distributions.

5. **Overall Performance**:
    - In many practical scenarios, especially those involving complex relationships, high dimensionality, and larger datasets, **Random Forest tends to outperform**

Naive Bayes. It provides better accuracy, robustness, and interpretability, making it a preferred choice in many classification tasks.

4.4 Advantages of the Proposed System

1. **Enhanced Detection Accuracy**:
   - Utilizing ensemble methods like Random Forest improves accuracy in identifying complex attack patterns compared to simpler models, leading to more reliable detection of IoT botnet activities.
2. **Robustness to Noise and Outliers**:
   - The Random Forest classifier's inherent ability to mitigate the impact of noise and outliers results in more stable predictions, enhancing overall system reliability.
3. **Real-Time Anomaly Detection**:
   - The system can be deployed for real-time monitoring of network traffic, enabling immediate detection and response to potential threats, thereby minimizing damage.
4. **Feature Importance Analysis**:
   - The Random Forest model provides insights into feature importance, allowing stakeholders to understand which attributes contribute most significantly to attack detection, aiding in network security strategy formulation.
5. **Scalability**:
   - The proposed system can scale to accommodate larger datasets as IoT environments expand, ensuring consistent performance and adaptability to increasing data volumes.
6. **Flexibility in Handling Diverse Data Types**:
   - The approach can handle a variety of feature types (categorical, numerical) effectively, making it suitable for diverse IoT network configurations.
7. **Improved Model Generalization**:

- By using ensemble methods, the system reduces the likelihood of overfitting, leading to better generalization on unseen data, which is crucial for effective anomaly detection.

8. **Efficient Data Processing**:
    - The integration of techniques like SMOTE for handling imbalanced datasets ensures that minority classes (e.g., specific attacks) are adequately represented, improving model performance.

CHAPTER 5

UML DIAGRAM

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process also be added to; or associated with, UML.

The Unified Modeling Language Is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS: The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.

- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

Class Diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an “is-a” or “has-a” relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed “methods” of the class.

Apart from this, each class may have certain “attributes” that uniquely identify the class.

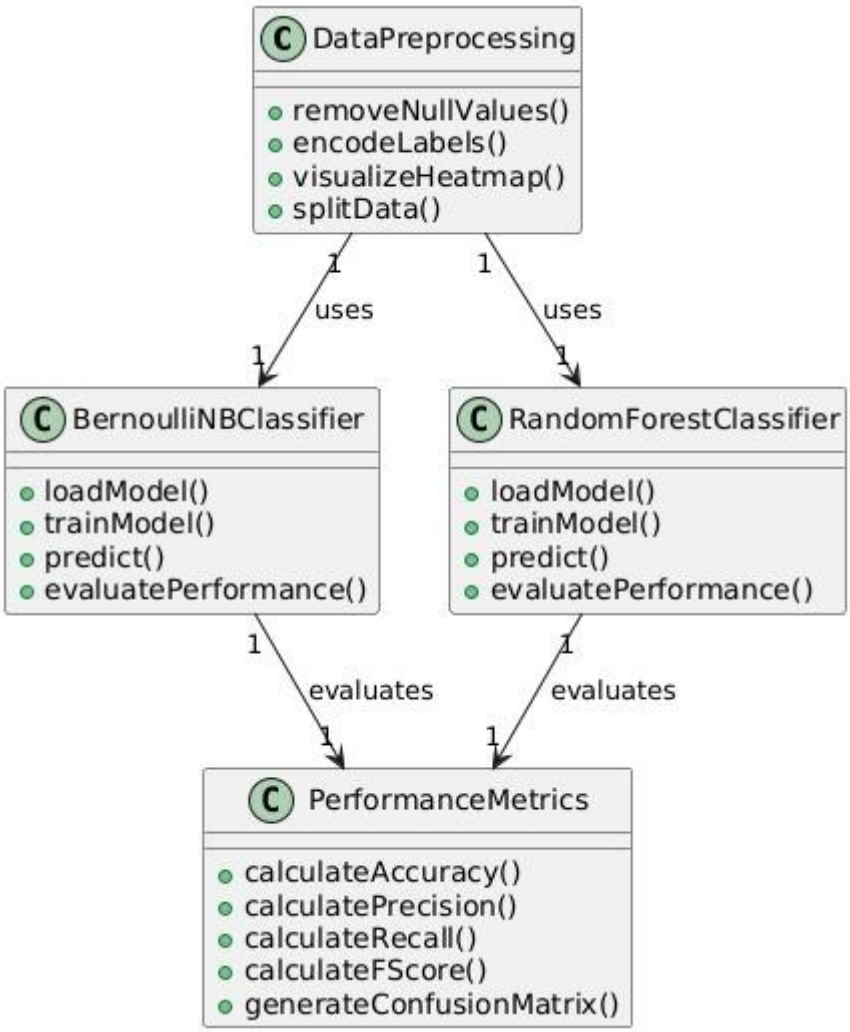


Figure 5.1: Class Diagram

Data flow diagram

A Data Flow Diagram (DFD) is a visual representation of the flow of data within a system or process. It is a structured technique that focuses on how data moves through different processes and data stores within an organization or a system. DFDs are commonly used in system analysis and design to understand, document, and communicate data flow and processing.

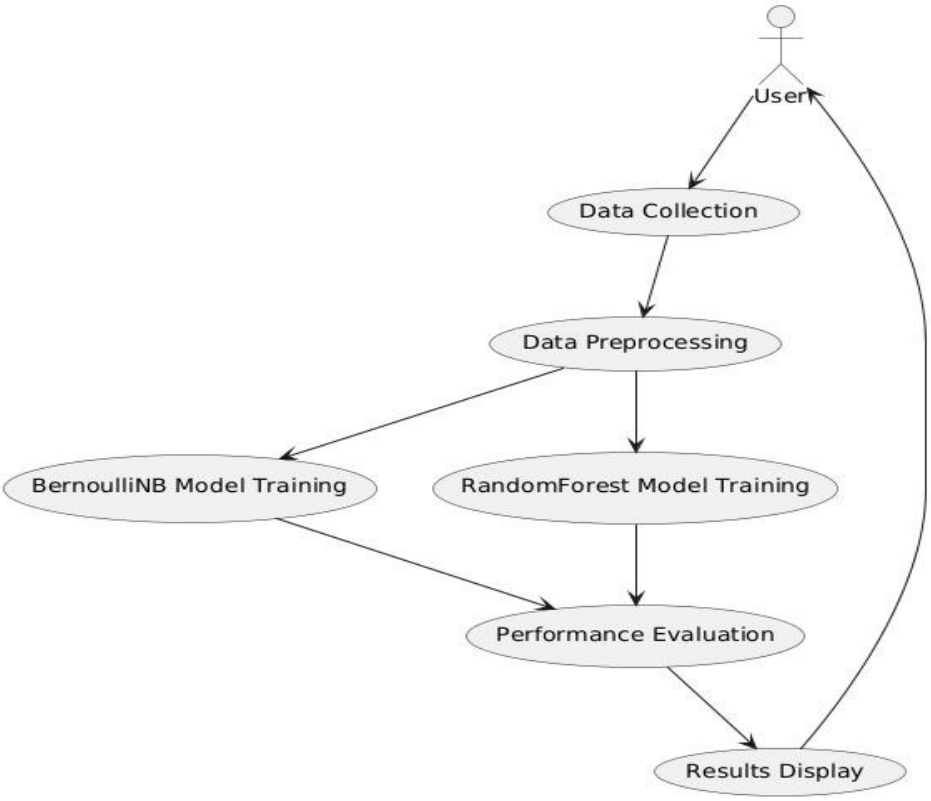


Figure 5.2: Data flow diagram

Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

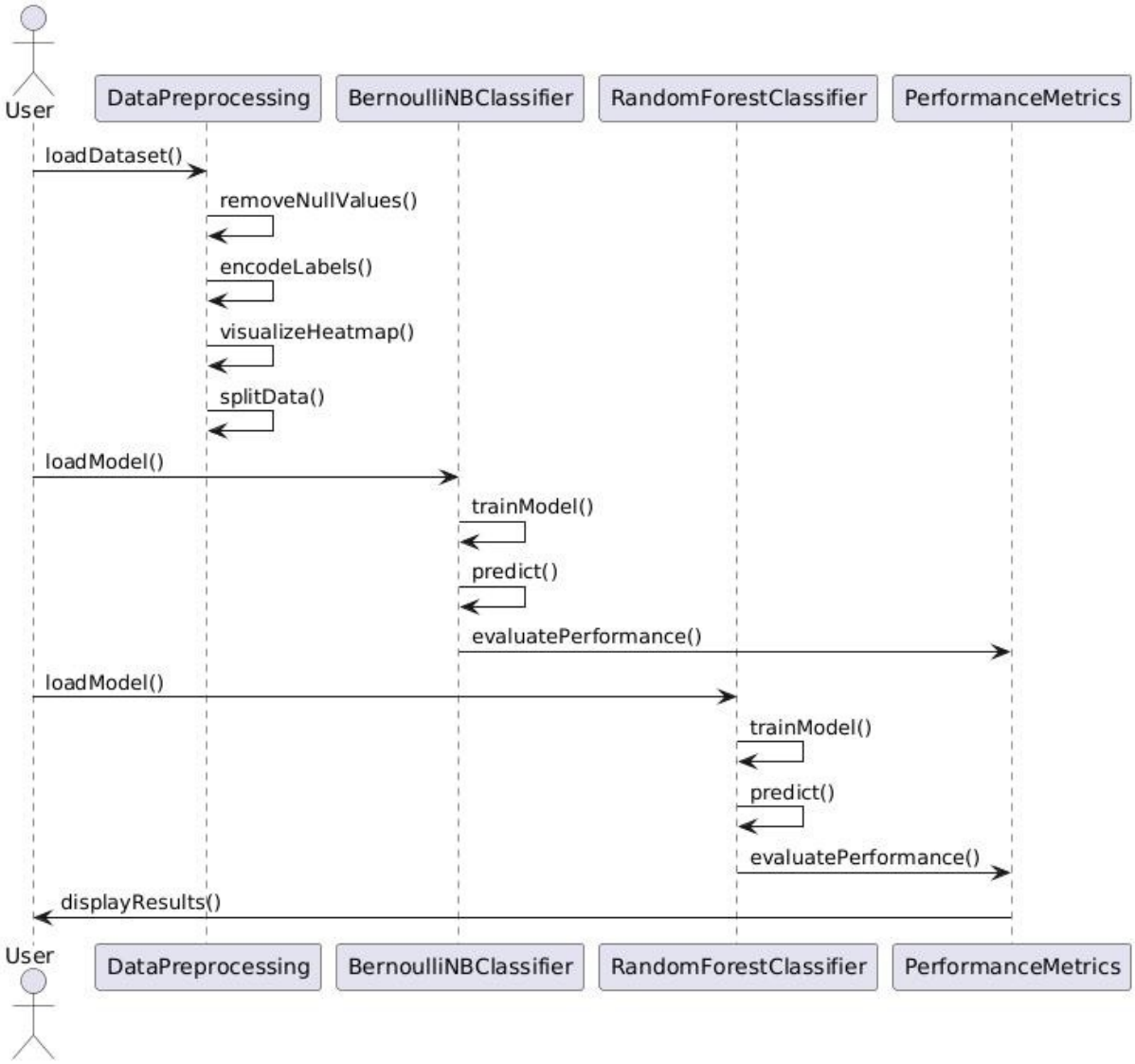


Figure 5.3: Sequence Diagram

Activity diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

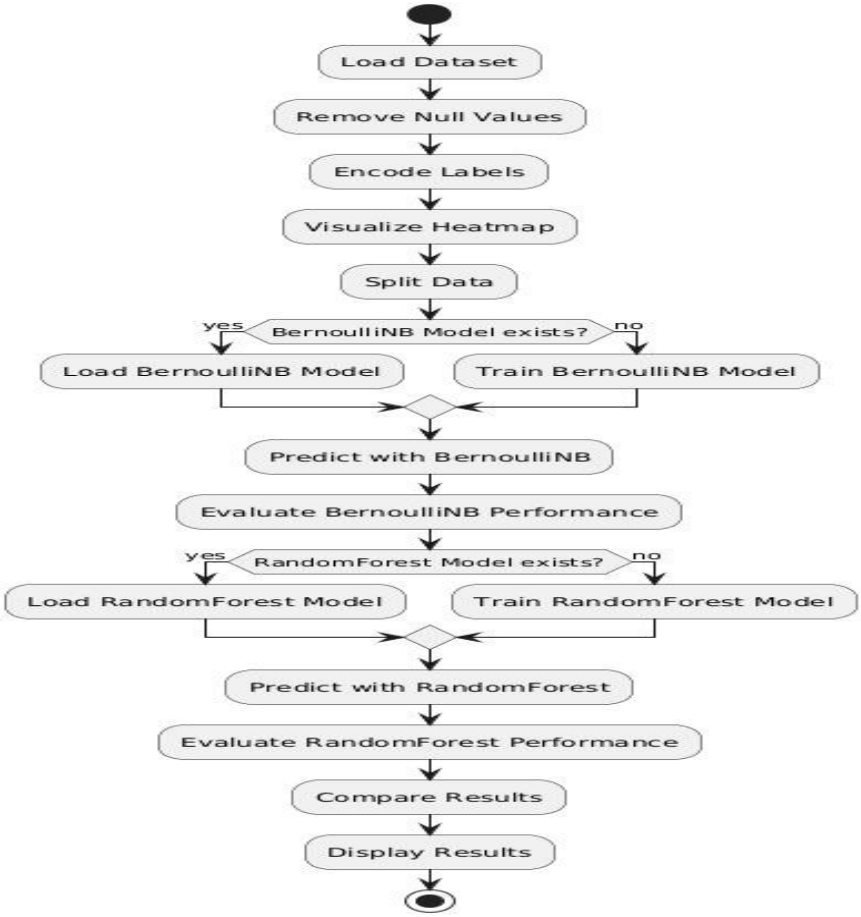


Figure 5.4: Activity diagram

Deployment diagram: The deployment diagram visualizes the physical hardware on which the software will be deployed.

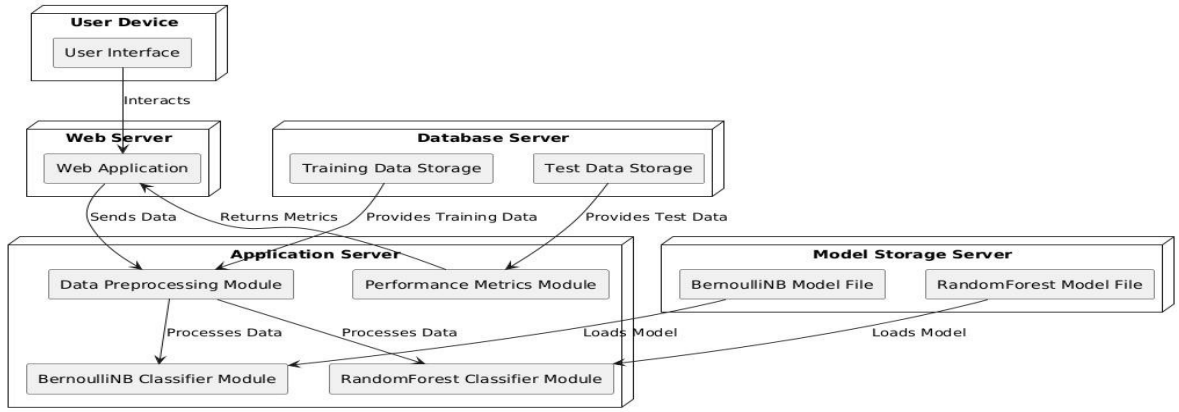


Figure 5.5: Deployment diagram

Use case diagram: The purpose of use case diagram is to capture the dynamic aspect of a system.

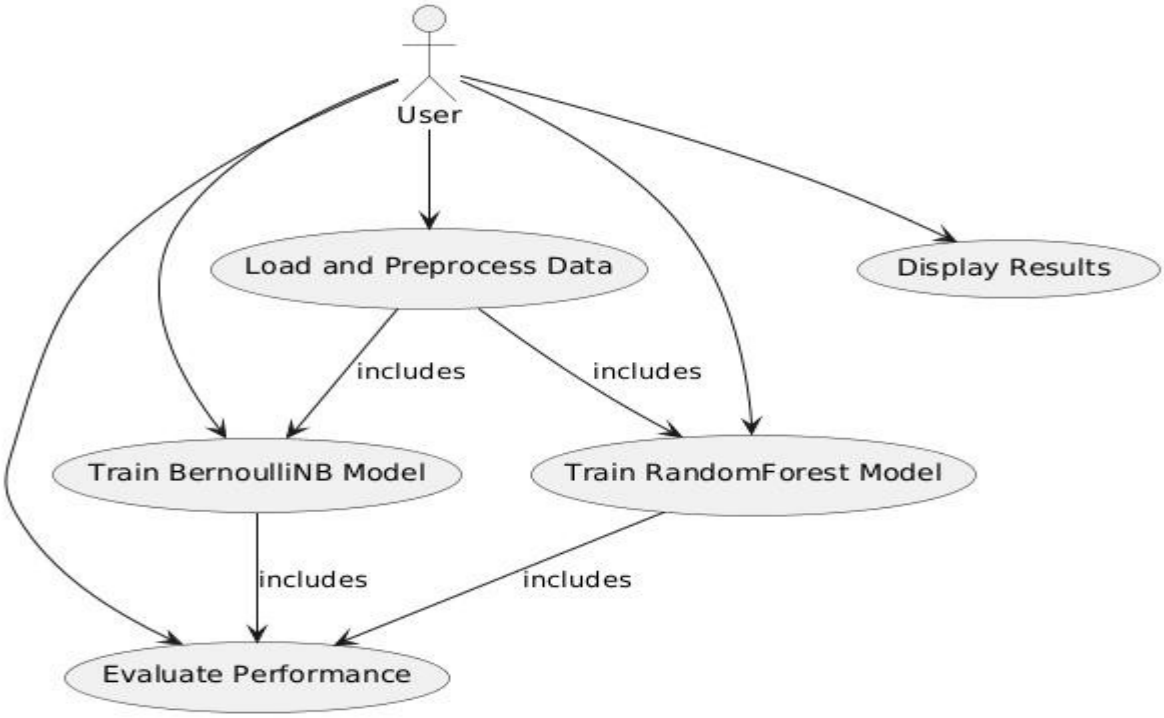


Figure 5.6: Use case diagram

Component diagram: Component diagram describes the organization and wiring of the physical components in a system.

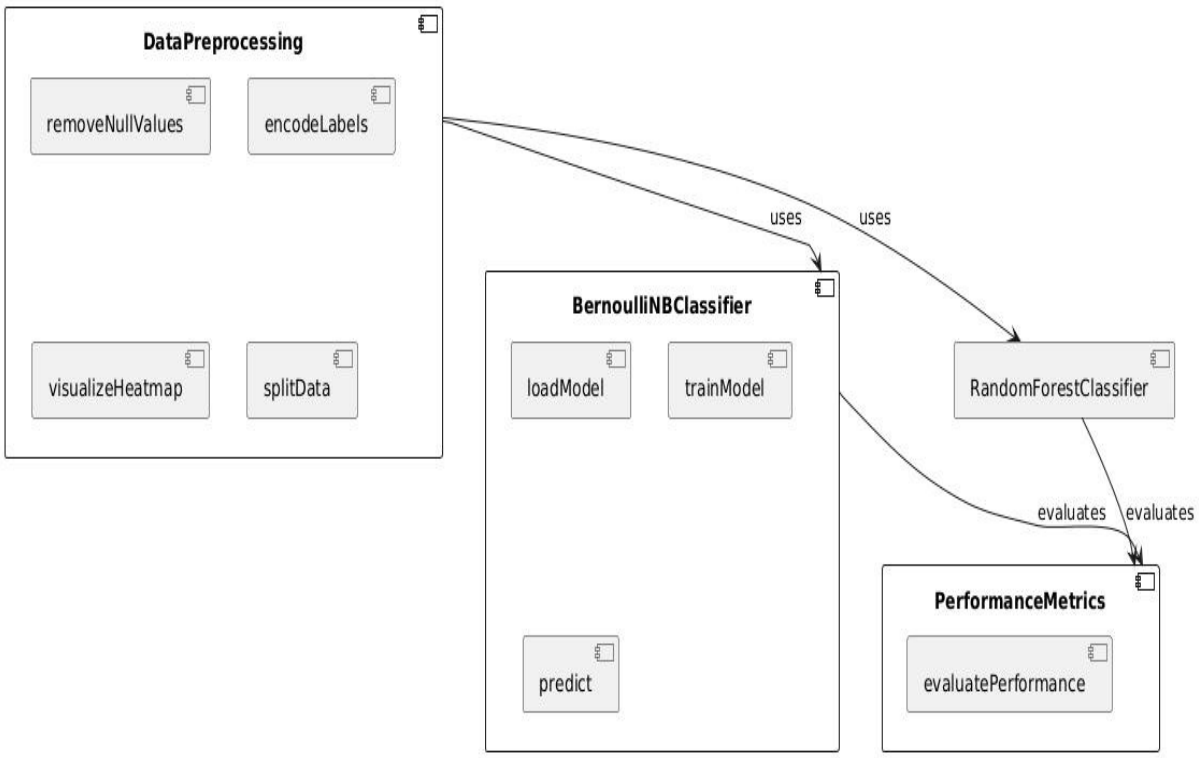


Figure 5.7: Component diagram

CHAPTER 6
SOFTWARE ENVIRONMENT

What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python

Let’s see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. **Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. **Affordable**

   Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3.Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build

web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the clientside. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum

never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function.
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background without breaking.

Modules Used in Project

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background without breaking.

Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular highlevel programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

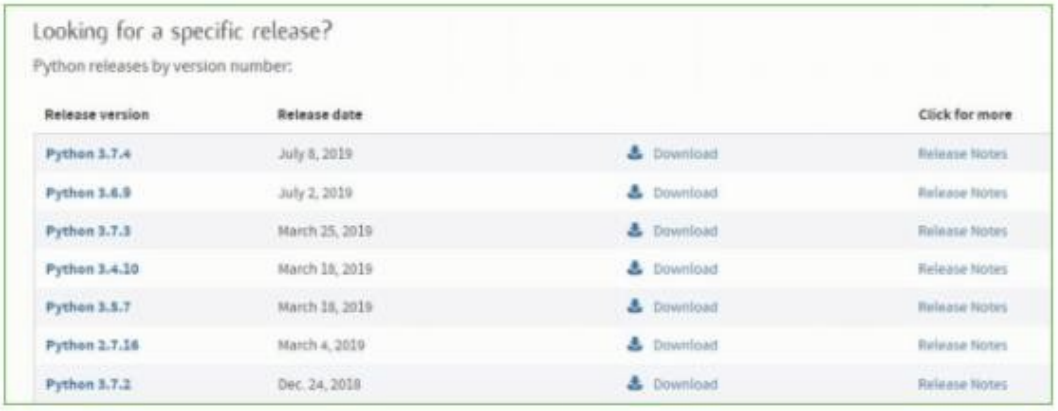
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab010f0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b243f	28082845	SIG
Windows help file	Windows		d639995573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0bfabe8318aa40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76de6bdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c6088bd72ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8b41879fda9413357413908	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d64514763b4789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa6d317df82c30983ea371d87c	1324608	SIG

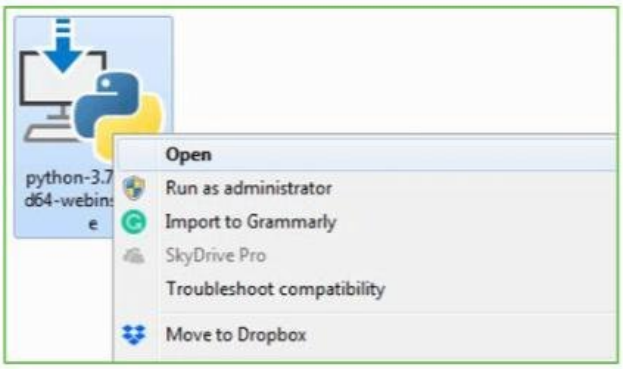
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.

Step 3: Click on Install NOW After the installation is successful. Click on Close.

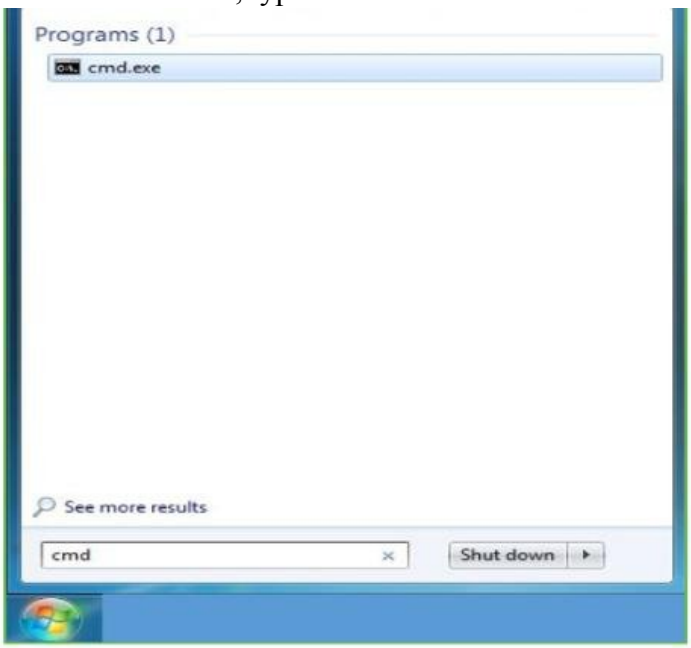
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

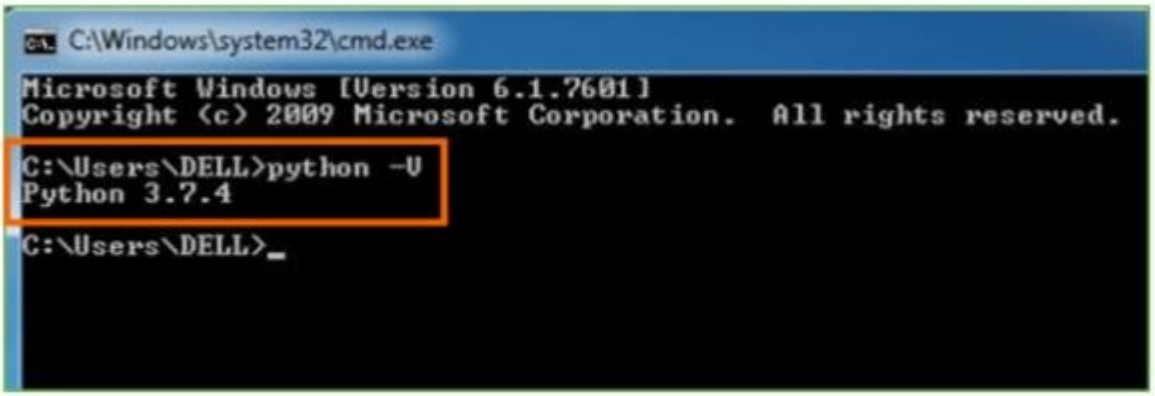
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.



Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

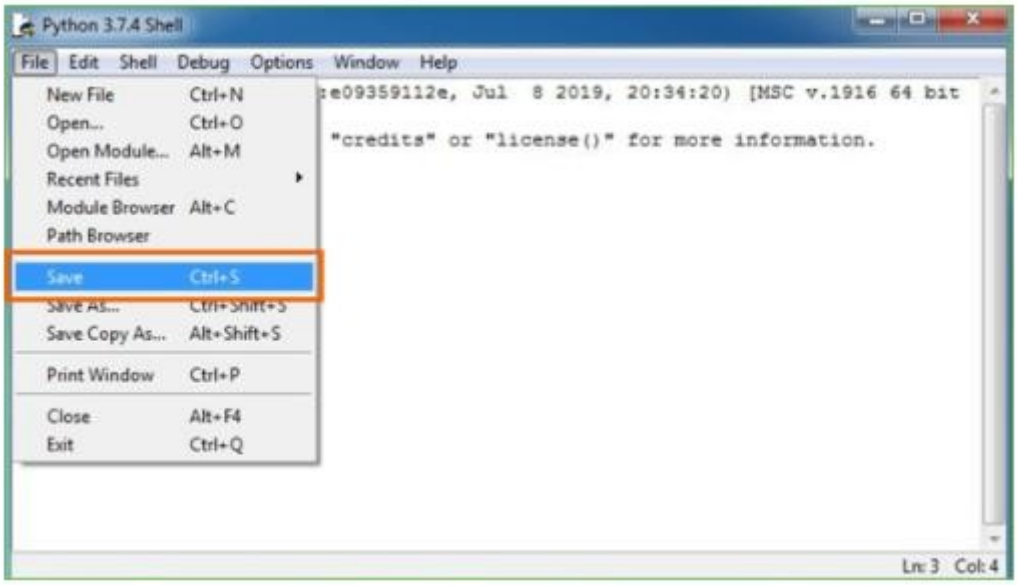
Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

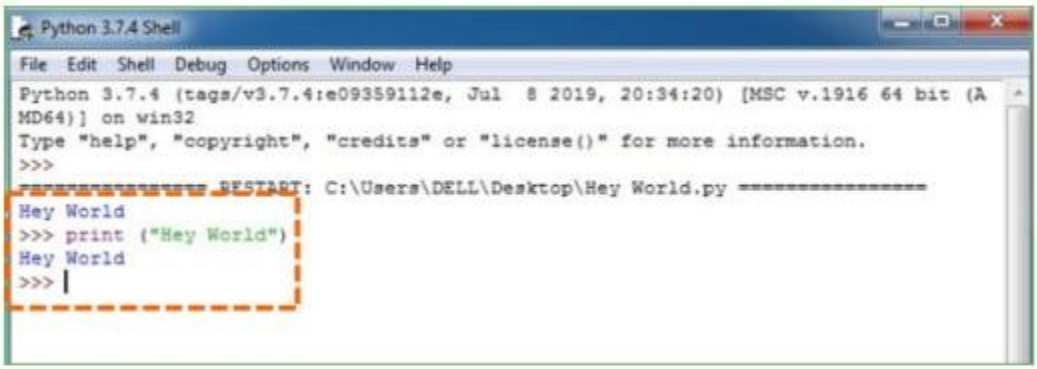
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g., enter print ("Hey World") and Press Enter.



You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

CHAPTER 7

SYSTEM REQUIREMENTS SPECIFICATIONS

Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system	:	Windows, Linux
Processor	:	minimum intel i3
Ram	:	minimum 4 GB
Hard disk	:	minimum 250GB

CHAPTER 8

FUNCTIONAL REQUIREMENTS

Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization Internal Outputs whose destination is within organization and they are the User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

Input Stages

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control

- Data transmission
- Data validation
- Data correction

Input Types

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

Input Media

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

ErrorAvoidance

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

User Interface Systems Can Be Broadly Clasified As:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

User Initiated Intergfaces

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has

been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

CHAPTER 9

SOURCE CODE

AI Framework for Identifying Anomalous Network Traffic in Mirai and BASHLITE IoT Botnet Attacks ## Importing Libraries import numpy as np import pandas as pd import matplotlib.pyplot as plt import seaborn as sns import warnings warnings.filterwarnings('ignore') from sklearn.preprocessing import LabelEncoder from imblearn.over_sampling import SMOTE from sklearn.model_selection import train_test_split from sklearn.metrics import precision_score from sklearn.metrics import recall_score from sklearn.metrics import f1_score from sklearn.metrics import accuracy_score,confusion_matrix,classification_report import os import joblib from sklearn.naive_bayes import BernoulliNB from sklearn.ensemble import RandomForestClassifier

Data Analysis data = pd.read_csv(r"C:\Users\USER\Desktop\saint martins\Traffic\BoTNeTIoT-L01-v2.csv") data data.head() data.tail() data.describe() data.info() data['Attack'].unique() #it refers to a value that appears only once or a distinct value within a specific dataset or column of a dataset

Data preprocessing

data.isnull().sum()

data.shape ###

Heatmap

labels = set(data['Attack']) labels

plt.figure(figsize=(15,10))

sns.heatmap(data.corr(),cmap = 'Accent',annot =

True) plt.xticks(rotation = 80) plt.yticks(rotation = 45)

```
plt.show() labels = ['Normal', 'BASHLITE', 'Mirai']

labels columns = ['Device_Name','Attack_subType']

data = data.drop(columns = columns) data

Labels = ['Attack']
for i in Labels:
    data[i] = LabelEncoder().fit_transform(data[i])

data ### countplot sns.set(style="darkgrid")

plt.figure(figsize=(12, 6)) ax =

sns.countplot(x=data['Attack'], palette="Set2")

plt.title("Count Plot") plt.xlabel("Categories")

plt.ylabel("Count") plt.show() x =

data.drop(['Attack'], axis = 1) x y = data['Attack'] y

## Splitting the Data x_train, x_test, y_train, y_test = train_test_split(x,y, test_size =

0.30, random_state = 42) x_train y_train x_test y_test x_train.shape y_train.shape

## Performance Evaluation
precision = [] recall = [] fscore = [] accuracy = []


def performance_metrics(algorithm, predict, testY):

    testY = testY.astype('int') predict = predict.astype('int') p =

    precision_score(testY, predict,average='macro') * 100 r =

    recall_score(testY, predict,average='macro') * 100 f =

    f1_score(testY, predict,average='macro') * 100 a =

    accuracy_score(testY,predict)*100 accuracy.append(a)

    precision.append(p) recall.append(r) fscore.append(f)

    print(algorithm+' Accuracy: '+str(a)) print(algorithm+'
```

```
    Precision    : '+str(p)) print(algorithm+' Recall    : '+str(r))
    print(algorithm+' FSCORE: '+str(f))
    report=classification_report(predict,
    testY,target_names=labels) print('\n',algorithm+" classification
    report\n",report) conf_matrix = confusion_matrix(testY,
    predict) plt.figure(figsize =(5, 5)) ax =
    sns.heatmap(conf_matrix, xticklabels = labels, yticklabels =
    labels, annot = True,
cmap="Blues" ,fmt ="g");
    ax.set_ylim([0,len(labels)])
    plt.title(algorithm+" Confusion matrix")
    plt.ylabel('True                    class')
    plt.xlabel('Predicted class') plt.show()
## BernoulliNBClassifier Algorithm if
os.path.exists('BernoulliNBClassifier.pkl'):
   # Load the Bernoulli Naive Bayes Classifier model
bnb_classifier = joblib.load('BernoulliNBClassifier.pkl')
predict = bnb_classifier.predict(x_test) else:
    # Train and save the Bernoulli Naive Bayes Classifier model
    bnb_classifier = BernoulliNB() bnb_classifier.fit(x_train,
    y_train)                    joblib.dump(bnb_classifier,
    'BernoulliNBClassifier.pkl')
```

```
# Predict using the trained Bernoulli Naive Bayes Classifier model y_pred_bnb
= bnb_classifier.predict(x_test)

# Evaluate the Bernoulli Naive Bayes Classifier model
performance_metrics('BernoulliNBClassifier', y_pred_bnb, y_test) ##
RandomForestClassifier if os.path.exists('RandomForest_weights.pkl'): # Load the model
from the pkl file
   classifier = joblib.load('RandomForest_weights.pkl')
else:
   # Train the classifier on the training data classifier =
   RandomForestClassifier(random_state=42)
   classifier.fit(x_train, y_train)
   # Save the model weights to a pkl file joblib.dump(classifier,
   'RandomForest_weights.pkl') print("RandomForest classifier model
   trained and model weights saved.")

y_pred = classifier.predict(x_test)
performance_metrics("RandomForest Classifier", y_pred, y_test)
test = pd.read_csv(r"test.csv") test test.info()
columns = ['Device_Name','Attack_subType']
test = test.drop(columns = columns)
test predict =
classifier.predict(test) predict
A='Normal'
```

```
B='BASHLITE'
c='mirai'
#test = pd.read_csv((r"test.csv")
predict = classifier.predict(test)
for i in range(len(predict)):
    if predict[i] == 0: print("{} :{}
    ".format(test.iloc[i,:],A)) elif predict[i]
    == 1: print("{} {}
    ".format(test.iloc[i,:],B)) elif
    predict[i]== 2:
        print("{} :{} ".format(test.iloc[i, :],c))
```

CHAPTER 10
RESULTSAND DISCUSSION

10.1 Implementation Description

1. Data Collection and Preprocessing: The project begins by acquiring network traffic data, which includes a range of features capturing different aspects of network behavior. The data is then preprocessed to prepare it for analysis. This involves handling missing values, dropping irrelevant columns, and encoding categorical variables using label encoding to convert them into a numerical format. Additionally, a heatmap is generated to understand the correlation between features, which helps in feature selection and dimensionality reduction.

2. Exploratory Data Analysis (EDA): EDA is performed to gain insights into the data distribution and the prevalence of different attack types, such as 'Normal,' 'BASHLITE,' and 'Mirai.' Visualization techniques like count plots are used to visualize the frequency of attack labels in the dataset. This step helps in understanding the dataset's structure and identifying any imbalances in the data, which is critical for model training.

3. Data Splitting: The preprocessed data is split into training and testing sets using an 80-20 split. The training set is used to train the machine learning models, while the testing set is reserved for evaluating the model's performance. A stratified sampling approach is applied to ensure that the distribution of attack types in both the training and testing sets remains consistent with the overall dataset.

4. Model Training: Two machine learning algorithms, Bernoulli Naive Bayes and Random Forest, are implemented for anomaly detection in the IoT network traffic. Each model is trained using the training data:

Bernoulli Naive Bayes: This algorithm is selected due to its simplicity and efficiency in handling binary or categorical data. It is particularly well-suited for high-dimensional datasets and provides a baseline model for comparison.

Random Forest Classifier: A more complex ensemble learning method, Random Forest, is employed to enhance detection accuracy. This model leverages multiple decision trees to improve generalization and robustness, making it suitable for identifying complex attack patterns.

5. Model Evaluation: After training, the models are evaluated using the testing set. Several performance metrics are calculated, including accuracy, precision, recall, and F1-score. These metrics provide a comprehensive understanding of the model's ability to correctly identify normal and anomalous traffic. Additionally, confusion matrices are generated to visualize the classification performance, highlighting the true positives, false positives, true negatives, and false negatives for each attack type.

6. Performance Comparison: The performance of both the Bernoulli Naive Bayes and Random Forest models is compared. While the Bernoulli Naive Bayes model provides a quick and efficient baseline, the Random Forest model typically offers higher accuracy and better generalization. The comparison helps in determining the most effective model for deployment in real-world scenarios.

7. Anomaly Detection in New Data: Once the models are trained and evaluated, they are used to predict anomalies in new, unseen data. The trained Random Forest model is particularly employed to classify new network traffic data into 'Normal,' 'BASHLITE,' and 'Mirai' categories. The prediction results are then analyzed to understand the model's ability to detect and classify network attacks in real-time.

8. Results Interpretation: The predictions are further analyzed by mapping the numerical predictions back to their respective categories ('Normal,' 'BASHLITE,' 'Mirai'). This step involves interpreting the model's output and understanding the classification of each network flow in the test dataset. The results help in validating the effectiveness of the AI framework in accurately identifying anomalous traffic.

9. Model Deployment and Future Enhancements: Finally, the trained models are saved for deployment in a real-time network monitoring system. The deployment phase involves integrating the models into a network security infrastructure where they can continuously analyze incoming traffic and detect potential botnet attacks. Future enhancements may include fine-tuning the models, incorporating additional features, or leveraging more advanced techniques such as deep learning for improved detection accuracy.

10.2 Dataset Description

The dataset utilized in this project is crucial for training and evaluating the AI framework designed to detect anomalous network traffic associated with Mirai and BASHLITE IoT botnet attacks. The dataset, sourced from real-world IoT network traffic, contains a comprehensive

collection of features that describe both normal and malicious activities, enabling the development of accurate and reliable machine learning models.

10.2.1 Data Source

The dataset, named "BoTNeTIoT-L01-v2.csv," was obtained from a publicly available repository that focuses on IoT security. This dataset is specifically tailored to reflect the network behaviors of IoT devices under both normal conditions and during botnet attacks, providing a rich and diverse set of data points for analysis.

10.2.2 Features and Attributes

The dataset contains numerous features that capture various aspects of network traffic, such as packet sizes, communication protocols, and time-based metrics. Some of the key features include:

Src IP (Source IP): The IP address from which the network traffic originates.

Dst IP (Destination IP): The IP address to which the network traffic is directed.

Src Port (Source Port): The port number from which the traffic is sent.

Dst Port (Destination Port): The port number to which the traffic is destined.

Protocol: The communication protocol used (e.g., TCP, UDP).

Flow Duration: The duration of the network flow in microseconds.

Total Fwd Packets: The total number of packets sent in the forward direction.

Total Backward Packets: The total number of packets sent in the backward direction.

Attack: The target variable indicating whether the traffic is associated with an attack. This feature has been encoded into numeric labels for machine learning purposes: 0 for Normal traffic, 1 for BASHLITE, and 2 for Mirai.

10.2.3 Data Preprocessing

Before feeding the data into the machine learning models, several preprocessing steps were performed:

Handling Missing Values: The dataset was checked for any missing values, and necessary imputation or removal was carried out to ensure data integrity.

Feature Selection: Redundant or irrelevant features, such as Device_Name and Attack_subType, were dropped from the dataset to focus on the most impactful attributes.

Label Encoding: The categorical target variable Attack was converted into numeric labels using LabelEncoder to facilitate model training.

Data Normalization: Continuous features were normalized to ensure that the models could learn effectively without being biased by the scale of the data.

Class Imbalance Handling: Given the potential imbalance between normal and attack classes, techniques like SMOTE (Synthetic Minority Over-sampling Technique) were applied to ensure that the models could accurately learn from all categories.

10.2.4 Data Splitting

The dataset was divided into training and testing subsets to evaluate the performance of the machine learning models. Typically, 70% of the data was used for training, while the remaining 30% was reserved for testing. This split ensured that the models were trained on a sufficient amount of data while still being evaluated on unseen samples to assess their generalization capability.

10.2.5 Data Visualization

To gain initial insights into the data, visualizations such as heatmaps and count plots were generated. The heatmap illustrated the correlation between different features, helping identify the most influential attributes for detecting anomalous traffic. The count plot provided a visual distribution of the different attack types in the dataset, highlighting any class imbalances that needed to be addressed during model training.

10.2.6 Dataset Usage

The dataset was employed to train and evaluate two machine learning algorithms: Bernoulli Naive Bayes (BNB) and Random Forest Classifier (RFC). These models were trained to distinguish between normal traffic and traffic indicative of Mirai and BASHLITE botnet attacks. The processed dataset enabled the development of an AI framework that effectively identifies anomalous traffic patterns, contributing to enhanced network security in IoT environments.

10.3 Results Description

The results of the analysis and machine learning models applied to the dataset are detailed below. Figures and tables included in this section visually represent key aspects of the dataset, model performance, and predictions.

	MI_dir_L0.1_weight	MI_dir_L0.1_mean	MI_dir_L0.1_variance	H_L0.1_weight	H_L0.1_mean	H_L0.1_variance	HH_L0.1_weight	HH_L0.1_mean	HH_L0.1_
0	1.000000	98.000000	0.000000e+00	1.000000	98.000000	0.000000e+00	1.000000	98.000000	0.000000e-
1	1.931640	98.000000	1.818989e-12	1.931640	98.000000	1.818989e-12	1.931640	98.000000	1.348699e
2	2.904273	86.981750	2.311822e+02	2.904273	86.981750	2.311822e+02	1.000000	66.000000	0.000000e-
3	3.902546	83.655268	2.040614e+02	3.902546	83.655268	2.040614e+02	1.000000	74.000000	0.000000e-
4	4.902545	81.685828	1.775746e+02	4.902545	81.685828	1.775746e+02	2.000000	74.000000	9.536743e
...	...	...	...	...	...	...	...	...	
7062601	2.937269	217.763487	1.770682e+04	2.937269	217.763487	1.770682e+04	1.220882	60.000000	9.540000e
7062602	1.730254	282.630543	1.054589e+04	1.730254	282.630543	1.054589e+04	1.213342	330.000000	5.390000e
7062603	2.730251	299.980395	7.204117e+03	2.730251	299.980395	7.204117e+03	1.213352	330.000000	6.610000e
7062604	2.882414	216.723647	1.775308e+04	2.882414	216.723647	1.775308e+04	1.209274	60.000000	6.740000e
7062605	2.032574	154.377267	1.303249e+04	2.032574	154.377267	1.303249e+04	1.299681	145.339354	1.010891e-

Figure 10.1: Overview of the Dataset

This figure displays the first few rows and last few rows of the dataset.

	MI_dir_L0.1_weight	MI_dir_L0.1_mean	MI_dir_L0.1_variance	H_L0.1_weight	H_L0.1_mean	H_L0.1_variance	HH_L0.1_weight	HH_L0.1_mean	HH_L0.1_std
count	7.062606e+06	7.062606e+06	7.062606e+06	7.062606e+06	7.062606e+06	7.062606e+06	7.062606e+06	7.062606e+06	7.062606e+0
mean	3.400682e+03	1.794441e+02	1.931062e+04	3.400682e+03	1.794441e+02	1.931066e+04	1.892359e+03	1.792406e+02	4.415659e+0
std	2.897012e+03	1.537109e+02	2.636844e+04	2.897012e+03	1.537107e+02	2.636842e+04	2.523083e+03	2.059018e+02	2.243629e+0
min	1.000000e+00	6.000000e+01	0.000000e+00	1.000000e+00	6.000000e+01	0.000000e+00	1.000000e+00	6.000000e+01	0.000000e+0
25%	1.000000e+00	6.000000e+01	0.000000e+00	1.000000e+00	6.000000e+01	0.000000e+00	1.000000e+00	6.000000e+01	0.000000e+0
50%	3.644882e+03	7.412707e+01	9.807711e+01	3.644882e+03	7.412707e+01	9.810144e+01	1.071281e+00	7.020764e+01	0.000000e+0
75%	6.354692e+03	3.486463e+02	4.887076e+04	6.354692e+03	3.486463e+02	4.887076e+04	4.201684e+03	9.314709e+01	3.293467e+0
max	8.946997e+03	1.401994e+03	4.520011e+05	8.946997e+03	1.401994e+03	4.520011e+05	7.944987e+03	1.470000e+03	6.784580e+0

Figure 10.2: Dataset Description

This figure shows the descriptive statistics of the dataset, including measures such as count, mean, standard deviation, min, and max values for each feature. It offers insights into the range and distribution of data points.

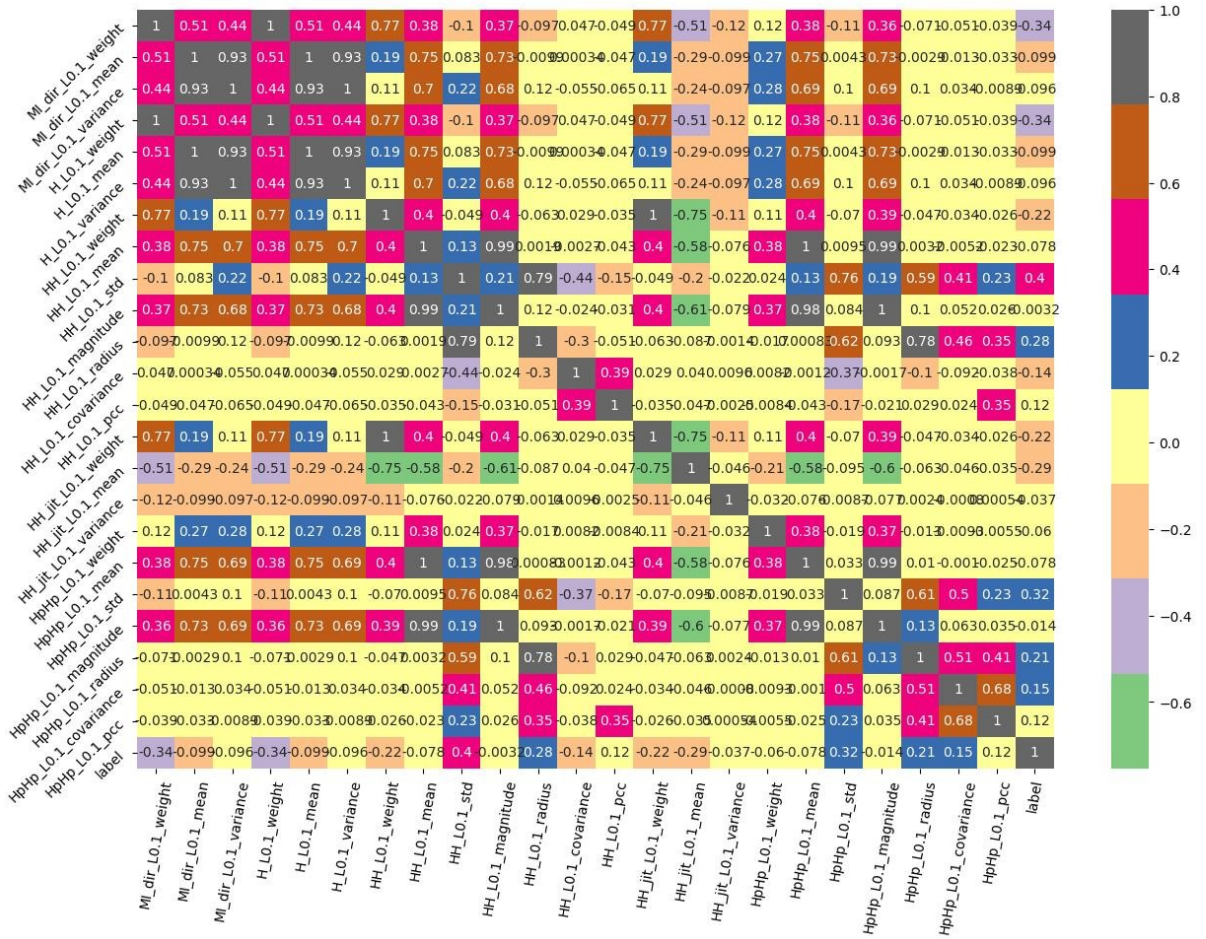


Figure 10.3: Correlation Heatmap

This heatmap illustrates the correlation between different features in the dataset. The color gradient indicates the strength and direction of correlations, helping to identify which features are most related to each other.

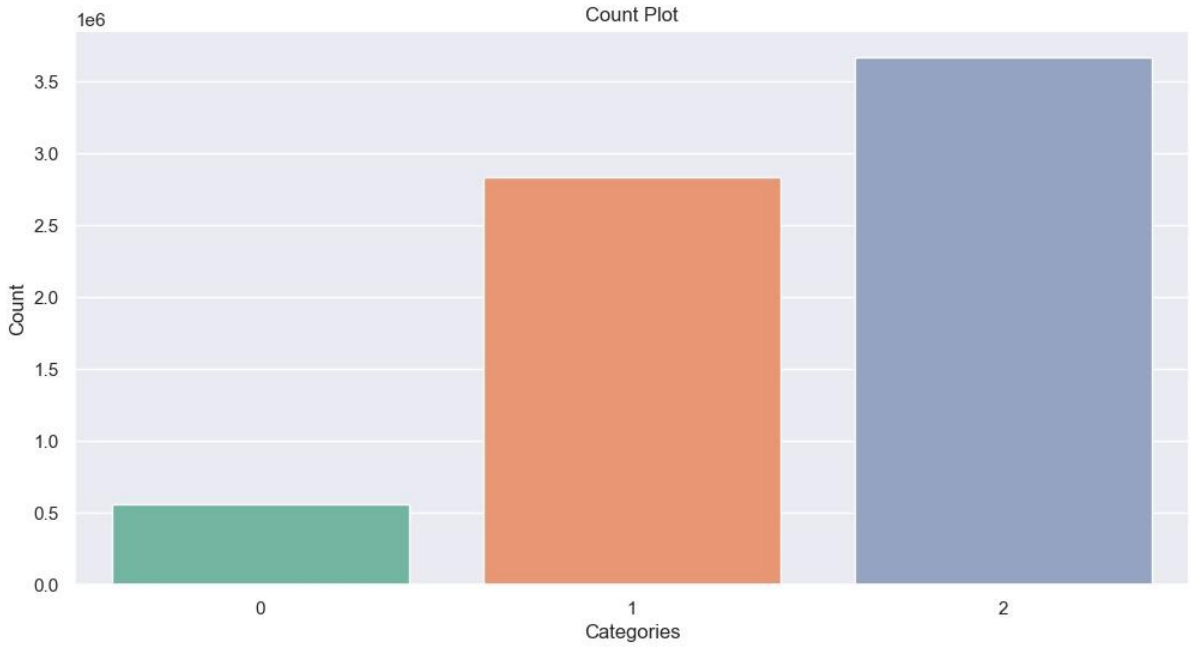


Figure 10.4: Count Plot of Attack Categories

This count plot visualizes the distribution of different attack categories in the dataset. It shows the number of instances for each class: Normal, BASHLITE, and Mirai.

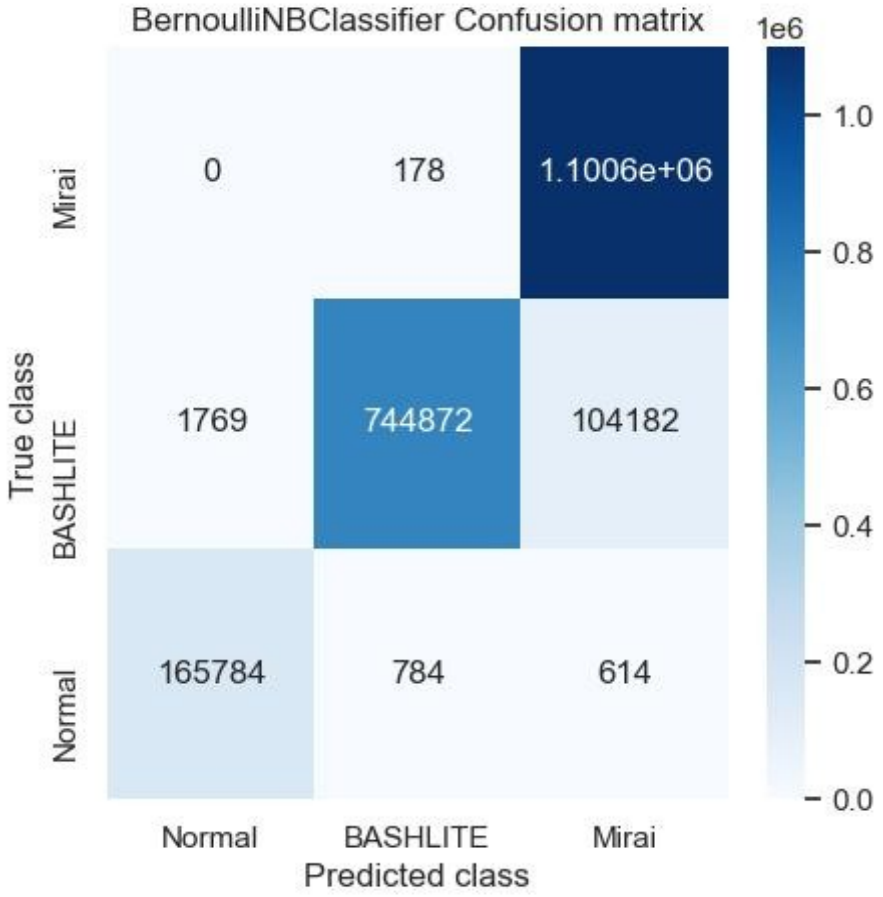


Figure 10.5: Confusion Matrix of Bernoulli Naive Bayes Classifier

This confusion matrix displays the performance of the Bernoulli Naive Bayes Classifier on the test data. It shows the number of true positives, false positives, true negatives, and false negatives for each class.

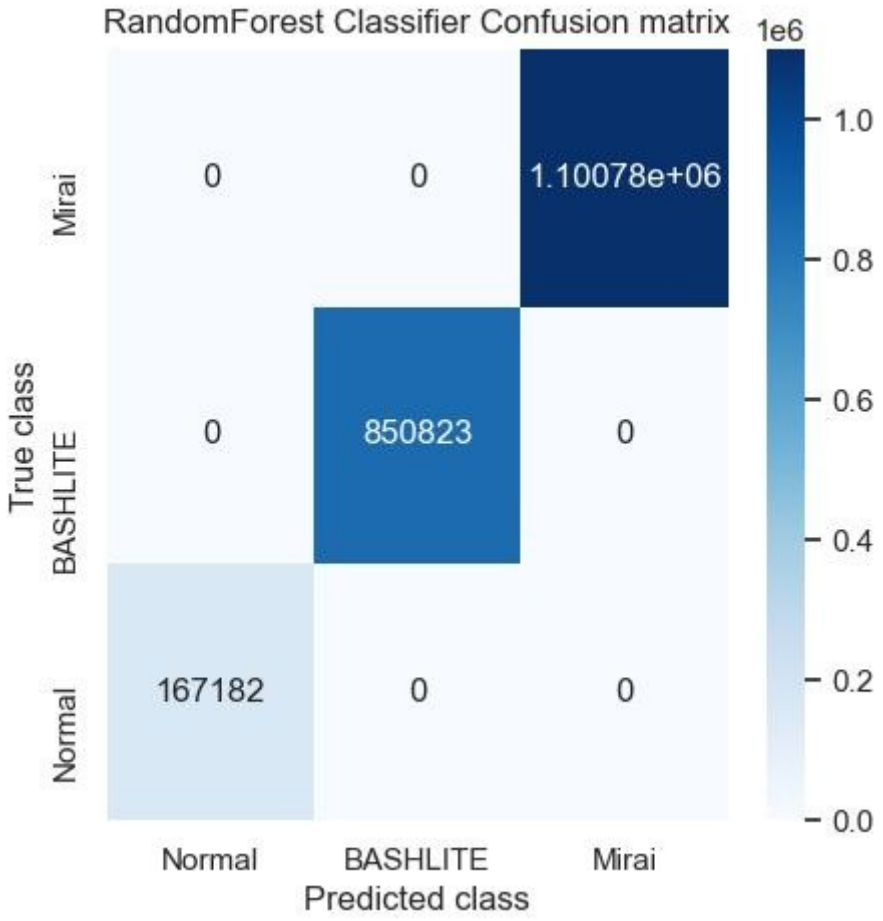


Figure 10.6: Confusion Matrix of Random Forest Classifier

This figure presents the confusion matrix for the Random Forest Classifier. It provides a detailed breakdown of the classifier's performance, highlighting the correct and incorrect predictions for each attack category.

```
array([1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0])
```

Figure 10.7: Prediction on New Test Data (First 15 Rows)

This figure presents the predictions made by the Random Forest Classifier on new test data, specifically for the first 15 rows. It shows the predicted class for each instance and the corresponding actual values for comparison.

CHAPTER 11

CONCLUSION AND FUTURE SCOPE

11.1 CONCLUSION

This project successfully demonstrates the application of machine learning techniques to enhance the detection of sophisticated IoT-based botnet attacks. By leveraging the Bernoulli Naive Bayes and Random Forest classifiers, the framework addresses the challenges posed by the increasing volume and complexity of network traffic due to the proliferation of IoT devices.

The implementation focuses on two prevalent botnets, Mirai and BASHLITE, which have historically exploited vulnerabilities in IoT devices to orchestrate large-scale Distributed Denial of Service (DDoS) attacks and other cyber threats. The use of machine learning enables the system to identify anomalous patterns in network traffic, which traditional rulebased systems might overlook. The Random Forest classifier, in particular, provides a high degree of accuracy and robustness, making it a suitable choice for real-world deployment in dynamic network environments.

This AI-driven approach offers significant improvements over manual inspection and traditional anomaly detection methods. The ability to process and analyze vast amounts of data in real time, coupled with the adaptability of machine learning models to evolving attack patterns, ensures that the framework remains effective even as cyber threats become more sophisticated. Additionally, the integration of performance metrics like accuracy, precision, recall, and F1-score provides a comprehensive evaluation of the models, ensuring they meet the necessary standards for deployment.

11.2 FUTURE SCOPE

1.Integration with RealTime Systems:

One of the primary future directions for this project is to integrate the trained machine learning models into real-time network monitoring systems. This would involve deploying the models within network security infrastructures to continuously monitor and analyze incoming traffic for anomalies, providing instant alerts and automated responses to potential threats. Real-time deployment would also require optimizing the models for speed and efficiency, ensuring minimal latency in detection.

2.IncorporatingAdvanced Machine Learning Techniques:

While Bernoulli Naive Bayes and Random Forest classifiers provide a solid foundation, future enhancements could explore more advanced machine learning techniques, such as deep learning. Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) could be employed to capture more complex patterns in network traffic data, potentially improving detection rates for sophisticated or novel attack vectors.

3.Expanding the Scope to Include Other Botnets:

The current framework focuses on detecting Mirai and BASHLITE botnets. Future work could expand the scope to include other emerging IoT botnets, such as Hajime, Amnesia, or Reaper. By incorporating a broader range of attack types, the framework could be made more versatile and capable of handling a wider array of threats.

4.Feature Engineering and Dimensionality Reduction:

Further exploration into feature engineering could enhance the model's performance. Techniques like Principal Component Analysis (PCA) or t-SNE could be employed to reduce dimensionality, enabling the models to focus on the most critical features while reducing computational overhead. Additionally, the exploration of new features, derived from more granular network traffic data, could improve the model's ability to distinguish between normal and malicious traffic.

5.Addressing Data Imbalance and Improving Model Generalization:

Given the nature of cyber attack datasets, there is often a significant imbalance between normal and anomalous traffic. Future work could involve the implementation of more sophisticated data balancing techniques, such as Synthetic Minority Over-sampling Technique (SMOTE) or Adaptive Synthetic (ADASYN), to enhance model training.

Additionally, techniques like cross-validation could be employed to improve model generalization and reduce overfitting.

6.Adaptive Learning and model Updating:

As cyber threats evolve, it is crucial for detection systems to adapt over time. Future developments could include the integration of adaptive learning mechanisms, allowing the model to update itself as new data becomes available. This could involve periodic retraining of the model or the implementation of online learning techniques, where the model continuously learns from new network traffic data without needing a complete retraining cycle.

7.Collaborative Threat Intelligence:

Another potential area for future development is the integration of collaborative threat intelligence. By sharing anonymized network traffic patterns and attack signatures across different organizations, the framework could benefit from a broader knowledge base, improving its ability to detect and respond to emerging threats. This could be facilitated through secure, decentralized platforms that allow for the exchange of threat data while maintaining privacy and confidentiality.

8.Compliance and Ethical Considerations:

As the framework is developed further, attention should be given to ensuring compliance with cybersecurity regulations and ethical considerations. This includes ensuring that the system respects user privacy and data protection laws, as well as addressing any potential biases in the model that could lead to false positives or negatives.

9.User Interface and Visualization:

To make the system more user-friendly, future work could involve developing an intuitive user interface that allows network administrators to easily interpret the model's predictions and gain insights into detected anomalies. Visualization tools, such as interactive dashboards and real-time alerts, could be integrated to provide a more comprehensive view of the network's security status.

REFERENCES

1) Soni, V.; Modi, P.; Chaudhri, V. Detecting Sinkhole attack in wireless sensor network. Int. J. Appl. Innov. Eng. Manag. 2013, 2, 29–32.

2) Abomhara, M.; Køien, G.M. Cyber security and the internet of things: Vulnerabilities, threats, intruders and attacks. J. Cyber Secur. Mobil. 2015, 4, 65–88.

3) Andrea, I.; Chrysostomou, C.; Hadjichristofi, G. Internet of things: Security vulnerabilities and challenges. In Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC), Larnaca, Cyprus, 6–9 July 2015; pp. 180–187.

4) Sivaraman, V.; Gharakheili, H.H.; Vishwanath, A.; Boreli, R.; Mehani, O. Network-level security and privacy control for smart-home IoT devices. In Proceedings of the 2015 [11] IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Abu Dhabi, UAE, 19–21 October 2015; pp. 163–167.

5) Ronen, E.; Shamir, A. Extended functionality attacks on IoT devices: The case of smart lights. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrücken, Germany, 21–24 March 2016; pp. 3–12.

6) Deogirikar, J.; Vidhate, A. Security attacks in IoT: A survey. In Proceedings of the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 10–11 February 2017; pp. 32–37.

7) Meidan, Y.; Bohadana, M.; Shabtai, A.; Ochoa, M.; Tippenhauer, N.O.; Guarnizo, J.D.; Elovici, Y. Detection of unauthorized IoT devices using machine learning techniques. arXiv 2017, arXiv:1709.04647.

8) Alaba, F.A.; Othman, M.; Hashem, I.A.T.; Alotaibi, F. Internet of things security: A survey. J. Netw. Comput. Appl. 2017, 88, 10–28.

9) McDermott, C.D.; Majdani, F.; Petrovski, A.V. Botnet detection in the internet of things using deep learning approaches. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.

10) Doshi, R.; Apthorpe, N.; Feamster, N. Machine learning DDoS detection for consumer internet of things devices. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24–24 May 2018; pp. 29–35.

11) Brun, O.; Yin, Y.; Gelenbe, E. Deep learning with dense random neural network for detecting attacks against IoT-connected home environments. Procedia Comput. Sci. 2018, 134, 458–463.

12) Yavuz, F.Y.; Devrim, Ü.N.A.L.; Ensar, G.Ü.L. Deep learning for detection of routing attacks in the internet of things. Int. J. Comput. Intell. Syst. 2018, 12, 39–58.

13) Shiranzaei, A.; Khan, R.Z. An Approach to Discover the Sinkhole and Selective Forwarding Attack in IoT. J. Inf. Secur. Res. 2018, 9, 107.

14) Palacharla, S.; Chandan, M.; GnanaSuryaTeja, K.; Varshitha, G. Wormhole attack: A major security concern in internet of things (IoT). Int. J. Eng. Technol. 2018, 7, 147–150.

15) Neshenko, N.; Bou-Harb, E.; Crichigno, J.; Kaddoum, G.; Ghani, N. Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internetscale IoT exploitations. IEEE Commun. Surv. Tutor. 2019, 21, 2702–2733.

16) Demotes-Mainard, J.; Cornu, C.; Guerin, A.; Bertoye, P.H.; Boidin, R.; Bureau, S.; Chrétien, J.M.; Delval, C.; Deplanque, D.; Dubray, C.; et al. How the new European data protection regulation affects clinical research and recommendations? *Therapies* **2019**, *74*, 31–42.

17) Barrett, C. Are the EU GDPR and the California CCPA becoming the de facto global standards for data privacy and protection? *Scitech Lawyer* **2019**, *15*, 24–29.

18) Hao, M.; Li, H.; Luo, X.; Xu, G.; Yang, H.; Liu, S. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Trans. Ind. Inform.* **2019**, *16*, 6532–6542.

19) Bhaskar, V.V.S.R.; Etikani, P.; Shiva, K.; Choppadandi, A.; Dave, A. Building explainable AI systems with federated learning on the cloud. *Webology* **2019**, *16*, 1–14.

20) Zhao, Z.; Feng, C.; Yang, H.H.; Luo, X. Federated-learning-enabled intelligent fog radio access networks: Fundamental theory, key techniques, and future trends. *IEEE Wirel. Commun.* **2020**, *27*, 22–28.

21) Drainakis, G.; Katsaros, K.V.; Pantazopoulos, P.; Sourlas, V.; Amditis, A. Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis.

In Proceedings of the 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 24–27 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–8.

22) Sunyaev, A.; Sunyaev, A. Cloud computing. In *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 195–236.

23) Li, L.; Fan, Y.; Tse, M.; Lin, K.Y. A review of applications in federated learning. *Comput. Ind. Eng.* **2020**, *149*, 106854.

24) Kholod, I.; Yanaki, E.; Fomichev, D.; Shalugin, E.; Novikova, E.; Filippov, E.; Nordlund, M. Open-source federated learning frameworks for IoT: A comparative review and analysis. *Sensors* **2020**, *21*, 167.

25) Golosova, J.; Romanovs, A. The advantages and disadvantages of the blockchain technology. In Proceedings of the 2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), Vilnius, Lithuania, 8–10 November 2018; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.

26) Nilsson, A.; Smith, S.; Ulm, G.; Gustavsson, E.; Jirstrand, M. A performance evaluation of federated learning algorithms. In Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning, Rennes, France, 10 December 2020; pp. 1–8.

27) Zhang, C.; Li, S.; Xia, J.; Wang, W.; Yan, F.; Liu, Y. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 20), Virtual, 14–16 July 2020; pp. 493–506.

28) Geiping, J.; Bauermeister, H.; Dröge, H.; Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 16937–16947.

29) Jiang, H.; Liu, M.; Yang, B.; Liu, Q.; Li, J.; Guo, X. Customized federated learning for accelerated edge computing with heterogeneous task targets. *Comput. Netw.* **2020**, *183*, 107569

30) Ye, Y.; Li, S.; Liu, F.; Tang, Y.; Hu, W. EdgeFed: Optimized federated learning based on edge computing. *IEEE Access* **2020**, *8*, 209191–209198.

31) Fang, C.; Guo, Y.; Wang, N.; Ju, A. Highly efficient federated learning with strong privacy preservation in cloud computing. *Comput. Secur.* **2020**, *96*, 101889.

32) Liu, L.; Zhang, J.; Song, S.; Letaief, K.B. Client-edge-cloud hierarchical federated learning. In Proceedings of the ICC 2020–2020 IEEE international conference on communications (ICC), Dublin, Ireland, 7–11 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.

33) Khan, L.U.; Saad, W.; Han, Z.; Hossain, E.; Hong, C.S. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1759–1799.

34) Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Poor, H.V. Federated learning for internet of things: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1622–1658.

35) ur Rehman, M.H.; Dirir, A.M.; Salah, K.; Damiani, E.; Svetinovic, D. TrustFed: A framework for fair and trustworthy cross-device federated learning in IIoT. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8485–8494.

36) Sheth, A.; Bhosale, S.; Kadam, H.; Prof, A. Research paper on cloud computing. *Int. J. Innov. Sci. Res. Technol.* **2021**, *6*, 2021.

37) Sobel, B.L. A new common law of web scraping. *Lewis Clark L. Rev.* **2021**, *25*, 147.

38) Rajendran, S.; Obeid, J.S.; Binol, H.; Foley, K.; Zhang, W.; Austin, P.; Brakefield, J.; Gurcan, M.N.; Topaloglu, U. Cloud-based federated learning implementation across medical centers. *JCO Clin. Cancer Inform.* **2021**, *5*, 1–11.

39) Makkar, A.; Ghosh, U.; Rawat, D.B.; Abawajy, J.H. Fedlearnsp: Preserving privacy and security using federated learning and edge computing. *IEEE Consum. Electron. Mag.* **2021**, *11*, 21–27.

40) Zhang, C.; Cui, L.; Yu, S.; James, J. A communication-efficient federated learning scheme for iot-based traffic forecasting. *IEEE Internet Things J.* **2021**, *9*, 11918–11931.

41) Su, Z.; Wang, Y.; Luan, T.H.; Zhang, N.; Li, F.; Chen, T.; Cao, H. Secure and efficient federated learning for smart grid with edge-cloud collaboration. *IEEE Trans. Ind. Inform.* **2021**, *18*, 1333–1344.

42) Nguyen, D.C.; Ding, M.; Pham, Q.V.; Pathirana, P.N.; Le, L.B.; Seneviratne, A.; Li, J.; Niyato, D.; Poor, H.V. Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet Things J.* **2021**, *8*, 12806–12825.

43) Kethireddy, R.R. AI-Driven Encryption Techniques for Data Security in Cloud Computing. *J. Recent Trends Comput. Sci. Eng. (JRTCSE)* **2021**, *9*, 27–38.

44) Brecko, A.; Kajati, E.; Koziorek, J.; Zolotova, I. Federated learning for edge computing: A survey. *Appl. Sci.* **2022**, *12*, 9124.

45) Islam, A.; Al Amin, A.; Shin, S.Y. FBI: A federated learning-based blockchain-embedded data accumulation scheme using drones for Internet of Things. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 972–976.

46) Ho, T.M.; Nguyen, K.K.; Cheriet, M. Federated deep reinforcement learning for task scheduling in heterogeneous autonomous robotic system. *IEEE Trans. Autom. Sci. Eng.* **2022**, *21*, 528–540.

47) Agrawal, S.; Sarkar, S.; Aouedi, O.; Yenduri, G.; Piamrat, K.; Alazab, M.; Bhattacharya, S.; Maddikunta, P.K.R.; Gadekallu, T.R. Federated learning for intrusion detection system: Concepts, challenges and future directions. *Comput. Commun.* **2022**, *195*, 346–361.

48) Kewate, N.; Raut, A.; Dubekar, M.; Raut, Y.; Patil, A. A review on AWS-cloud computing technology. *Int. J. Res. Appl. Sci. Eng. Technol.* **2022**, *10*, 258–263.

49) Pham, X.Q.; Nguyen, T.D.; Huynh-The, T.; Huh, E.N.; Kim, D.S. Distributed cloud computing: Architecture, enabling technologies, and open challenges. *IEEE Consum. Electron. Mag.* **2022**, *12*, 98–106.

50) Khan, S.; Kabanov, I.; Hua, Y.; Madnick, S. A systematic analysis of the capital one data breach: Critical lessons learned. *ACM Trans. Priv. Secur.* **2022**, *26*, 1–29.

51) Pandya, S.; Srivastava, G.; Jhaveri, R.; Babu, M.R.; Bhattacharya, S.; Maddikunta, P.K.R.; Mastorakis, S.; Piran, M.J.; Gadekallu, T.R. Federated learning for smart cities: A comprehensive survey. *Sustain. Energy Technol. Assess.* **2023**, *55*, 102987.

52) Chimuco, F.T.; Sequeiros, J.B.; Lopes, C.G.; Simões, T.M.; Freire, M.M.; Inácio, P.R. Secure cloud-based mobile apps: Attack taxonomy, requirements, mechanisms, tests and automation. *Int. J. Inf. Secur.* **2023**, *22*, 833–867.

53) Gu, J. An Empirical Study on the Judicial Regulation of Data Crawling Unfair Competition. *Int. J. Educ. Humanit.* **2023**, *9*, 61–66.

54) Shreyas, S. Security Model for Cloud Computing: Case Report of Organizational Vulnerability. *J. Inf. Secur.* **2023**, *14*, 250–263.

55) Kitsios, F.; Chatzidimitriou, E.; Kamariotou, M. The ISO/IEC 27001 information security management standard: How to extract value from data in the IT sector. *Sustainability* **2023**, *15*, 5828.

56) Kaleem, S.; Sohail, A.; Tariq, M.U.; Asim, M. An improved big data analytics architecture using federated learning for IoT-enabled urban intelligent transportation systems. *Sustainability* **2023**, *15*, 15333.

57) Hijazi, N.M.; Aloqaily, M.; Guizani, M.; Ouni, B.; Karray, F. Secure federated learning with fully homomorphic encryption for iot communications. *IEEE Internet Things J.* **2023**, *11*, 4289–4300.

58) Duan, Q.; Huang, J.; Hu, S.; Deng, R.; Lu, Z.; Yu, S. Combining federated learning and edge computing toward ubiquitous intelligence in 6G network: Challenges, recent advances, and future directions. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 2892–2950.

59) Hu, K.; Gong, S.; Zhang, Q.; Seng, C.; Xia, M.; Jiang, S. An overview of implementing security and privacy in federated learning. *Artif. Intell. Rev.* **2024**, *57*, 204.

60) Shubyn, B.; Maksymyuk, T.; Gazda, J.; Rusyn, B.; Mrozek, D. Federated Learning: A Solution for Improving Anomaly Detection Accuracy of Autonomous Guided Vehicles in Smart Manufacturing. In *Digital Ecosystems: Interconnecting Advanced Networks with AI Applications*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 746–761.

61) Anusuya, R.; D Renuka, K. FedAssess: Analysis for Efficient Communication and Security Algorithms over Various Federated Learning Frameworks and Mitigation of Label Flipping Attack. *Bull. Pol. Acad. Sci. Tech. Sci.* **2024**, *72*, e148944.

62) Babar, M.; Qureshi, B.; Koubaa, A. Investigating the impact of data heterogeneity on the performance of federated learning algorithm using medical imaging. *PLoS ONE* **2024**, *19*, e0302539.

63) Mehta, S.; Sarpal, S.S. Maximizing Privacy in Reinforcement Learning with Federated Approaches. In Proceedings of the 2023 4th International Conference on Intelligent Technologies (CONIT), Hubballi, India, 21–23 June 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 1–5.

64) Liberti, F.; Berardi, D.; Martini, B. Federated Learning in Dynamic and Heterogeneous Environments: Advantages, Performances, and Privacy Problems. *Appl. Sci.* **2024**, *14*, 8490.

65) Al-Quraan, M.M.Y. Federated Learning Empowered Ultra-Dense Next-Generation Wireless Networks. Ph.D. Thesis, University of Glasgow, Glasgow, Scotland, 2024.

66) Zohaib, S.M.; Sajjad, S.M.; Iqbal, Z.; Yousaf, M.; Haseeb, M.; Muhammad, Z. Zero Trust VPN (ZT-VPN): A Systematic Literature Review and Cybersecurity Framework for Hybrid and Remote Work. *Information* **2024**, *15*, 734.

67) Lakhani, R. Zero Trust Security Models: Redefining Network Security in Cloud Computing Environments. *Int. J. Innov. Res. Comput. Commun. Eng.* **2024**, *12*, 141–156.

68) Kayes, A.; Rahayu, W.; Dillon, T.; Shahraki, A.S.; Alavizadeh, H. Safeguarding Individuals and Organisations from Privacy Breaches: A Comprehensive Review of Problem Domains, Solution Strategies, and Prospective Research Directions. *IEEE Internet Things J.* **2024**, *12*, 1247–1265.

69) Gao, X.; Hou, L.; Chen, B.; Yao, X.; Suo, Z. Compressive Learning Based Federated Learning for Intelligent IoT with Cloud-Edge Collaboration. *IEEE Internet Things J.* **2024**, *12*, 2291–2294.

70) Guo, S.; Chen, H.; Liu, Y.; Yang, C.; Li, Z.; Jin, C.H. Heterogeneous Federated Learning Framework for IIoT Based on Selective Knowledge Distillation. *IEEE Trans. Ind. Inform.* **2024**, *21*, 1078–1089.

71) Prigent, C.; Chelli, M.; Costan, A.; Cudennec, L.; Schubotz, R.; Antoniu, G. Efficient Resource-Constrained Federated Learning Clustering with Local Data Compression on the Edge-to-Cloud Continuum. In Proceedings of the HiPC 2024-31st IEEE International Conference on High Performance Computing, Data, and Analytics, Bangalore, India, 18–21 December 2024.

72) Xu, Y.; Zhao, B.; Zhou, H.; Su, J. FedAdaSS: Federated Learning with Adaptive Parameter Server Selection Based on Elastic Cloud Resources. *CMES-Comput. Model. Eng. Sci.* **2024**, *141*, 609–629.

73) Salim, M.M.; Camacho, D.; Park, J.H. Digital Twin and federated learning enabled cyberthreat detection system for IoT networks. *Future Gener. Comput. Syst.* **2024**, *161*, 701–713.

74) Zhou, J.; Pal, S.; Dong, C.; Wang, K. Enhancing quality of service through federated learning in edge-cloud architecture. *Ad Hoc Netw.* **2024**, *156*, 103430.

75) Qi, Y.; Feng, Y.; Wang, X.; Li, H.; Tian, J. Leveraging Federated Learning and Edge Computing for Recommendation Systems within Cloud Computing Networks. *arXiv* **2024**, arXiv:2403.03165.

76) Wu, Z.; Sun, S.; Wang, Y.; Liu, M.; Gao, B.; Pan, Q.; He, T.; Jiang, X. Agglomerative federated learning: Empowering larger model training via end-edge-cloud collaboration. In Proceedings of the IEEE INFOCOM 2024-IEEE Conference on Computer Communications, Vancouver, BC, Canada, 20–23 May 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 131–140

77) Bhansali, P.K.; Hiran, D.; Kothari, H.; Gulati, K. Cloud-based secure data storage and access control for internet of medical things using federated learning. *Int. J. Pervasive Comput. Commun.* **2024**, *20*, 228–239.

78) Parra-Ullauri, J.M.; Madhukumar, H.; Nicolaescu, A.C.; Zhang, X.; Bravalheri, A.; Hussain, R.; Vasilakos, X.; Nejabati, R.; Simeonidou, D. kubeFlower: A privacy-

preserving framework for Kubernetes-based federated learning in cloud–edge environments. *Future Gener. Comput. Syst.* **2024**, *157*, 558–572.

79) Vinoth, K.; Sasikumar, P. VINO_EffiFedAV: VINO with efficient federated learning through selective client updates for real-time autonomous vehicle object detection. *Results Eng.* **2025**, *25*, 103700.

80) Hamid, S.; Huda, M.N. Mapping the landscape of government data breaches: A bibliometric analysis of literature from 2006 to 2023. *Soc. Sci. Humanit. Open* **2025**, *11*, 101234.