Sustainable Computing: Informatics and Systems 43 (2024) 101019

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Optimizing risk mitigation: A simulation-based model for detecting fake IoT clients in smart city environments

Mahmoud AlJamal [a], Ala Mughaid [a], Bashar Al shboul [b], Hani Bani-Salameh [b], Shadi Alzubi [c], Laith Abualigah [d,e,f,g,*]

[a] Department of Information Technology, Faculty of Prince Al-Hussien bin Abdullah II for IT, The Hashemite University, Zarqa 13133, Jordan
[b] Department of Software Engineering, Faculty of Prince Al-Hussien bin Abdullah for IT, The Hashemite University, Zarqa, Jordan
[c] Faculty of Science and IT, Al-Zaytoonah University of Jordan, Amman, Jordan
[d] Computer Science Department, Al al-Bayt University, Mafraq 25113, Jordan
[e] Applied science research center, Applied science private university, Amman 11931, Jordan
[f] MEU Research Unit, Middle East University, Amman 11831, Jordan
[g] Jadara Research Center, Jadara University, Irbid 21110, Jordan

ARTICLE INFO

Keywords:
Fake clients
Smart city
Internet of Thing (IoT)
Machine learning (ML)
CyberSecurity

ABSTRACT

Smart cities represent the future of urban evolution, characterized by the intricate integration of the Internet of Things (IoT). This integration sees everything, from traffic management to waste disposal, governed by interconnected and digitally managed systems. As fascinating as the promise of such cities is, they have its challenges. A significant concern in this digitally connected realm is the introduction of fake clients. These entities, masquerading as legitimate system components, can execute a range of cyber-attacks. This research focuses on the issue of fake clients by devising a detailed simulated smart city model utilizing the Netsim program. Within this simulated environment, multiple sectors collaborate with numerous clients to optimize performance, comfort, and energy conservation. Fake clients, who appear genuine but with malicious intentions, are introduced into this simulation to replicate the real-world challenge. After the simulation is configured, the data flows are captured using Wireshark and saved as a CSV file, differentiating between the real and fake clients. We applied MATLAB machine learning techniques to the captured data set to address the threat these fake clients posed. Various machine learning algorithms were tested, and the k-nearest neighbors (KNN) classifier showed a remarkable detection accuracy of 98 77%. Specifically, our method increased detection accuracy by 4.66%, from 94.02% to 98.68% over three experiments conducted, and enhanced the Area Under the Curve (AUC) by 0.49%, reaching 99.81%. Precision and recall also saw substantial gains, with precision improving by 9.09%, from 88.77% to 97.86%, and recall improving by 9.87%, from 89.23% to 99.10%. The comprehensive analysis underscores the role of preprocessing in enhancing the overall performance, highlighting its superior performance in detecting fake IoT clients in smart city environments compared to conventional approaches. Our research introduces a powerful model for protecting smart cities, merging sophisticated detection techniques with robust defenses.

1. Introduction

The Internet of Things (IoT) systems, a groundbreaking technological development, interconnect many devices and objects within different environments. Embedded with sensors, software, and network connectivity, these entities exchange and analyze data, transforming everyday objects into smart, communicative devices [1,2]. Integrating the physical and digital realms fosters automation, efficiency, and convenience on an unprecedented scale. From household appliances to industrial machinery, IoT encompasses a diverse range of devices, enabling them to interact and make intelligent decisions autonomously. This network enhances our daily lives and revolutionizes various industry sectors by creating a connected world of smart, automated systems [3,4]. The advantages of IoT are manifold and far-reaching. IoT enables enhanced efficiency, convenience, and cost savings. It empowers data-driven decision-making, improves communication and connectivity, and offers opportunities for personalized experiences and

* Corresponding author at: Computer Science Department, Al al-Bayt University, Mafraq 25113, Jordan.
E-mail addresses: mahmood.yj.98@gmail.com (M. AlJamal), ala.mughaid@hu.edu.jo (A. Mughaid), bashar.alshboul@hu.edu.jo (B. Al shboul), Hani@hu.edu.jo (H. Bani-Salameh), sh.sheeh@gmail.com (S. Alzubi), aligah.2020@gmail.com (L. Abualigah).

https://doi.org/10.1016/j.suscom.2024.101019
Received 15 March 2024; Received in revised form 3 June 2024; Accepted 10 July 2024
Available online 19 July 2024

improved client satisfaction [5,6]. IoT also holds the potential to transform industries, optimize resource utilization, and drive operational optimization. IoT's advantages extend across various domains, from smart homes to industrial automation, paving the way for a connected and intelligent future [7].

IoT smart cities utilize advanced technology and interconnected devices to enhance urban living by improving sustainability and efficiency. These cities integrate IoT into their systems, enabling the collection and analysis of data from diverse sources, which is essential for informed decision-making and intelligent urban system management [8,9]. A network of sensors and devices embedded in urban infrastructure monitors various aspects of city life, from energy consumption to public safety, facilitating real-time data exchange and monitoring [10]. This data, processed through advanced analytics and machine learning, informs city planners and administrators, aiding in resource optimization and effective management. Furthermore, IoT smart cities exemplify the principles of cyber–physical systems (CPS), intertwining physical infrastructure with digital technologies, such as sensors, communication networks, and cloud computing. This integration creates a dynamic system that continually adapts and optimizes city services based on real-time feedback, thus advancing urban efficiency, sustainability, and resilience [11].

1.1. Background and motivation

The rapid advancement of IoT has given rise to smart cities, which leverage IoT technologies to enhance urban efficiency, sustainability, and quality of life. IoT devices, such as sensors, actuators, smart meters, and connected vehicles, are key components of smart city infrastructures, facilitating applications like traffic management, energy optimization, and public safety. However, this interconnectivity introduces significant cybersecurity challenges, particularly the threat of fake IoT clients. These malicious entities exploit vulnerabilities to gain unauthorized access, engage in data theft, disrupt services, and introduce malware. Traditional security measures, like firewalls and intrusion detection systems, are insufficient for the dynamic IoT environments of smart cities. Thus, there is an urgent need for advanced detection mechanisms. The motivation behind this paper is to address this need by using the Netsim program to simulate real-world smart city scenarios, introduce fake clients, and capture data flows for analysis. By applying advanced machine learning techniques, we aim to accurately identify and isolate fake clients, thereby enhancing the security and resilience of smart cities.

1.2. Scope of the study

The scope of this research is to address the emerging problem of fake clients in smart cities based on the Internet of Things. It involves simulating an actual environment representing a smart city containing both normal and fake clients connected to city controllers using the well-known NetSim simulator. This network is configured to reflect the reality of the problem, where normal clients send legitimate data flows while fake clients attempt to disrupt the systems in various ways. Data flows are captured using specialized tools and organized into a dataset suitable for cybersecurity researchers interested in developing research on detection processes, specifically using AI techniques and studying the behavior of fake clients.

1.3. Problem definition

Smart cities, while offering numerous benefits, are susceptible to various cyber-attacks. Key threats include Denial-of-Service (DoS) attacks, Man-in-the-Middle (MitM) attacks, data manipulation, and unauthorized access. Ransomware attacks, insider threats, botnet attacks, and supply chain attacks further complicate the security landscape. Fake IoT clients, specifically, pose significant risks by mimicking authentic devices, launching attacks like DDoS attacks to disrupt critical infrastructure, and gaining unauthorized access to sensitive data. These activities can affect decision making processes, affect the efficiency of smart city services, and compromise the overall security and privacy of the urban environment.

1.4. Technical information

To address these challenges, this research presents a unique method of risk reduction through the construction of a simulation-based model. The model aims to identify and minimize the danger posed by fake IoT clients in smart city settings. This is accomplished by integrating real-time data streams and machine learning algorithms to enable dynamic identification and response to emerging threats. The simulation environment, developed using NetSim software, includes a variety of clients and sectors to replicate the real-world smart city scenario. Data flows from this simulation are captured using Wireshark and analyzed using MATLAB machine learning techniques.

1.5. Fake clients in smart cities

An IoT fake client in a smart city context is a malicious entity that impersonates legitimate IoT devices to exploit vulnerabilities and disrupt city services. These fake clients can mimic authentic devices, launching attacks like denial-of-service to disrupt critical infrastructure, leading to service unavailability. They pose significant risks by gaining unauthorized access to sensitive data and compromising data integrity through security vulnerabilities. By manipulating or injecting false data, they can impact decision-making processes, affecting the efficiency of smart city services. Their actions can have cascading effects on interconnected systems, highlighting the need for robust security measures in smart city infrastructure to protect against such sophisticated threats and ensure the resilience and reliability of the ecosystem [12]. Table 1 presents the main differences between normal and fake clients.

1.6. Research contributions and objectives

The research contributions and objectives of this paper are summarized in the following points:

- We define a new term in the smart city security domain: the Client.
- We generate a novel smart city simulation using NetSim simulator containing Clients and normal clients using.
- We generate a novel dataset from the data flow of this simulation. It is considered the first of its kind due to the need for more data sets of this type.
- We utilized multiple ML classification algorithms to detect a fake client over a smart city and prevent it from accessing the system with high accuracy.

The structure of this paper is designed as follows. Section 2 shows the related works part. Section 3 shows the procedure of the proposed smart city simulation. Section 4 shows the results and discussion. The conclusion and future work direction are given in Section 5.

2. Literature review

The integration of the IoT in the development of smart cities brings a transformative impact on traditional urban environments [9,13,14]. This integration addresses the challenges of escalating urban growth and contributes significantly to environmental preservation by reducing energy consumption and mitigating pollution. Moreover, IoT technology fosters economic development and enhances the quality of life by providing increased comfort and luxury [15]. This paradigm

Table 1
Comparison between normal client and fake client.

Aspect	Normal client	Fake client
Purpose	Legitimate use of smart city services	Malicious intent, unauthorized activities
Authorization	Authenticated and authorized access	Takes the role of an authorized client
Behavior	Follows expected protocols and standards	Mimics behavior but performs malicious actions
Data Handling	Uses data as intended respects privacy	May manipulate or steal sensitive data
System Interaction	Interacts with smart city infrastructure	Attempts to exploit vulnerabilities
Service Disruption	Does not disrupt services intentionally	Can launch attacks to disrupt services
Monitoring	Regular user of services	May attempt unauthorized monitoring
Security Measures	Adheres to security protocols and measures	Seeks to bypass security measures
Impact	Contributes to the functioning of the system	Poses risks to security and privacy
Detection	Typically not flagged as suspicious	May be detected through security measures
Intent	Genuine interest in utilizing smart city services	Carrying out activities with malicious intent

shift positions information technology as a crucial tool for controlling, exchanging, and managing systems across various human life domains [16]. In the healthcare sector, the application of IoT exemplifies its potential: remote monitoring of patients by medical practitioners, with temperature and heart rate sensors providing real-time alerts for prompt medical intervention. This advances the healthcare sector by enabling flexibility, professionalism, and precision, allowing effective patient monitoring from afar [17].

The traffic department in smart cities dramatically benefits from implementing IoT, using sensors to monitor and regulate traffic flow, reducing congestion. These sensors provide data that is analyzed to understand traffic patterns, identify congested areas, and inform city administrators about effective traffic management. This information allows for real-time updates to commuters on congested routes and alternative options, enhancing urban mobility [18]. In blockchain-IoT systems, security concerns such as exploiting fake clients in Sybil attacks pose significant challenges, particularly in the Internet of Medical Things (IoMT). For instance, in a blockchain IoT healthcare system, a fake client might transmit false temperature reports, posing as a legitimate sensor. To counter this, using digital signatures has been suggested to address authentication and integrity issues. However, the effectiveness of digital signatures is limited by vulnerabilities like private key compromise, indicating that security concerns in such systems remain a critical challenge.

Wireless Sensor Networks (WSNs) are integral to IoT networks and comprise nodes with various capabilities. A recent study designed a signature-based collaborative blockchain Intrusion Detection System (IDS) for IoT, employing rules and signatures for intrusion detection and database updates across nodes. This system, however, faced challenges with internal attacks, where nodes might provide fake signatures, affecting the IDS's performance. To address this, the researchers implemented a blockchain technique that uses a distributed database to detect intrusions and reduce the impact of these fake malicious signatures [19]. Another study highlighted vulnerabilities in a handmade IoT system, particularly in an ESP32-based temperature measurement device. An experiment showed that attackers with basic network hacking and ESP32 programming skills could gain unauthorized access and manipulate data by creating a fake ESP32 client. This vulnerability underscores the need for enhanced security measures, such as adopting TCP over UDP and more robust authentication mechanisms, to protect against unauthorized access and data manipulation by fake clients [20].

A study examining model poisoning attacks in federated learning noted that the assumption of access to a significant portion of compromised genuine clients is unrealistic in large-scale systems. A new method termed the Model Poisoning Attack based on Fake Clients (MPAF) was proposed, where fake clients introduce carefully crafted false model updates into the system. This approach effectively diminishes the global model's test accuracy by steering it towards a low-accuracy base model, challenging the efficacy of conventional defenses like norm clipping. These findings highlight the need for advanced defense strategies in federated learning [21]. Moreover, another study delved into IoT vulnerabilities, focusing on insider attacks. By analyzing data from heterogeneous sources within an organizational network, various machine learning algorithms were assessed for threat detection. XGBoost emerged as the most effective, with a classification accuracy of 93.8%, while LSTM recorded the lowest at 90.2%. Random Forest also demonstrated high efficiency with an accuracy of 93.7%. This research underscores the vital role of machine learning in enhancing IoT security, especially when it involves multi-source data, providing a comprehensive perspective for identifying and mitigating cyber threats [22]. Table 2 provides a summary of the related works.

The reviewed literature highlights the multifaceted applications and benefits of IoT in smart cities, ranging from enhanced traffic management to improved urban sustainability. However, it also underscores significant cybersecurity challenges, particularly concerning fake IoT clients. While several studies have addressed various aspects of IoT security, such as blockchain-based IDS and the role of machine learning in threat detection, there remains a critical gap in the real-time detection and mitigation of fake clients in smart city environments. For instance, Zahmatkesh et al. (2020) and Brundu et al. (2016) focus on the general benefits of IoT integration but do not address specific security threats. Djahel et al. (2014) concentrate on traffic management, leaving broader IoT security issues unexplored. Studies like Cao et al. (2022) and Selvakumar et al. (2019) discuss blockchain and federated learning approaches to IoT security. Still, these methodologies differ significantly from the machine learning-based detection of fake clients proposed in our study.

Our research fills this gap by simulating a real-world smart city environment using NetSim, introducing fake clients, and capturing data flow for comprehensive analysis. By applying advanced machine learning techniques, we aim to provide a robust and scalable solution for the real-time detection and isolation of fake clients, thereby enhancing the overall security and resilience of smart cities. This approach addresses the current shortcomings in existing security measures and contributes a novel perspective to the ongoing discourse on IoT security.

3. Methodology

In our methodology to detect fake clients within IoT smart city environments, we first simulated a smart city network using a NetSim simulator, encompassing sectors like City HQ, Traffic and Police, Healthcare, and Education and introduced potential fake clients. This fake client can carry out many cyber-attacks. Our simulation contains a group of fake clients that establish a DDoS attack on legitimate clients in the smart city. Data flow from this simulation was captured using Wireshark and stored in a CSV file. The methodology involved two main parts: initially, create the dataset from the simulation, then execute three ML experiments for fake client detection. The first experiment used the original dataset, while the second involved a 100,000 record subset undergoing further preprocessing. The third experiment applied the Synthetic Minority Over-sampling Technique SMOTE model to a balanced dataset, progressively improving detection accuracy with each step. These processes and their outcomes are detailed in Fig. 1.

Table 2
Summary of literature review on IoT in smart cities.

Author	Focus area	Key findings	Methodology	Implications	Comparison with current study
Zahmatkesh et al. (2020), Brundu et al. (2016)	IoT in Smart Cities	IoT's integration in urban environments enhances sustainability, reduces energy consumption, and improves the quality of life.	Discussed the transformative impact of IoT on various urban sectors, including healthcare.	Highlights IoT's role in urban growth, environmental preservation, and healthcare advancements.	Focuses on the general benefits of IoT, whereas the current study specifically addresses cybersecurity threats from fake clients.
Djahel et al. (2014)	IoT in Traffic Management	IoT sensors in traffic departments help reduce congestion and optimize traffic flow.	Analyzed traffic data from IoT sensors for traffic management.	Emphasizes IoT's importance in urban mobility and traffic regulation.	Concentrates on traffic management; current study addresses broader IoT security issues.
Cao et al. (2022)	Blockchain-IoT Systems Security	Addressed security vulnerabilities in blockchain-IoT systems, especially regarding fake clients.	Discussed the limitations of digital signatures in securing IoT systems.	Points to the need for stronger security measures in IoT and blockchain systems.	Similar focus on fake clients; current study introduces simulation and machine learning for detection.
Selvakumar et al. (2019)	Blockchain IDS in IoT Networks	Developed a blockchain-based IDS for IoT, addressing internal attack challenges.	Implemented a blockchain technique with a distributed database for intrusion detection.	Shows the effectiveness of blockchain in enhancing IoT network security.	Uses blockchain for IDS; current study uses machine learning for fake client detection.
Barybin et al. (2019)	IoT System Vulnerabilities	Exposed security weaknesses in a handmade IoT system, highlighting the risk of unauthorized access.	Conducted an experiment to demonstrate vulnerabilities in an ESP32-based device.	Underlines the necessity of robust security protocols in IoT systems.	Focuses on vulnerabilities; current study focuses on detection and mitigation of fake clients.
Cao et al. (2022)	Model Poisoning in Federated Learning	Proposed a novel Model Poisoning Attack based on Fake Clients (MPAF) in federated learning systems.	Introduced MPAF to compromise the accuracy of federated learning models.	Calls for more sophisticated defenses in federated learning against model poisoning.	Similar focus on fake clients; current study emphasizes real-time detection in smart cities.
Chowdhury et al. (2021)	IoT Security and Machine Learning	Investigated the role of machine learning in detecting IoT insider attacks using heterogeneous data.	Utilized machine learning algorithms for threat detection in IoT networks.	Highlights the effectiveness of machine learning in IoT security against insider threats.	Uses machine learning for threat detection; current study applies it specifically to fake client detection.

Integrating IoT devices in urban environments is a cornerstone of smart city development, enabling the optimization of services like energy management, traffic control, waste management, and healthcare. However, the IoT's transformative impact on urban development brings cybersecurity challenges, particularly the issue of fake clients. These fake clients, masquerading as legitimate users or devices within the IoT ecosystem, exploit trust-based communication protocols to gain unauthorized access, manipulate interconnected devices and services, and carry out various cyber-attacks, including data breaches and Distributed Denial of Service (DDoS) attacks. This vulnerability poses a significant threat to the security and sustainability of smart cities. To address this, our work introduces a simulation of a smart city environment involving fake clients to demonstrate the impacts and risks associated with these deceptive entities, as shown in Fig. 2, and the importance of robust security measures in maintaining the resilience of urban IoT networks.

1. **Sensors and Clients:**
   Camera sensors are used in smart cities for several critical applications due to their ability to capture visual data. Here are some key reasons why camera sensors are essential in IoT-based smart cities. Camera sensors are vital in monitoring public spaces, streets, and critical infrastructure for security and surveillance purposes. They can help in crime prevention, detection of suspicious activities, and provide evidence for investigations. Cameras monitor traffic flow, detect congestion, and analyze traffic patterns. This data helps optimize traffic signal timings, manage traffic flow, and improve overall transportation efficiency. Camera sensors in smart cities provide valuable visual data for real-time decision-making, improve urban planning, and enhance the quality of life for residents and visitors.

   Temperature sensors are crucial components in IoT-based smart cities as they provide valuable data for various applications and services. Here are some key reasons why temperature sensors are used in IoT-based smart cities. Temperature sensors deployed across the city can provide real-time weather data. This information is essential for accurate weather forecasts, climate monitoring, and predicting extreme weather events. Temperature sensors help identify urban heat islands and areas in cities where temperatures are significantly higher than their surroundings. Understanding and mitigating this effect is crucial for improving urban planning and reducing heat-related health risks.

   Alarm sensors are crucial in IoT-based smart cities, providing an early warning system for various critical scenarios. Alarm sensors, such as motion and door/window sensors, detect unauthorized entry or intrusions in homes, businesses, public buildings, and sensitive areas. They trigger alarms and alert security personnel or residents in real time, enabling quick responses to potential security threats. Overall, alarm sensors enhance the safety and security of smart cities by providing early detection and warning capabilities. They allow fast responses to emergencies, prevent accidents, and protect residents and infrastructure.
2. **Applications:**
   HTTP Applications: HTTP, the Hypertext Transfer Protocol, serves as the backbone of web communication, facilitating data exchange between clients (e.g., web browsers) and servers. In our simulation, we leverage HTTP applications to emulate real-world web traffic patterns and study how servers, proxies, and content delivery networks (CDNs) respond under various network conditions. By analyzing HTTP traffic, we gain insights into server loads, response times, and network performance, enabling

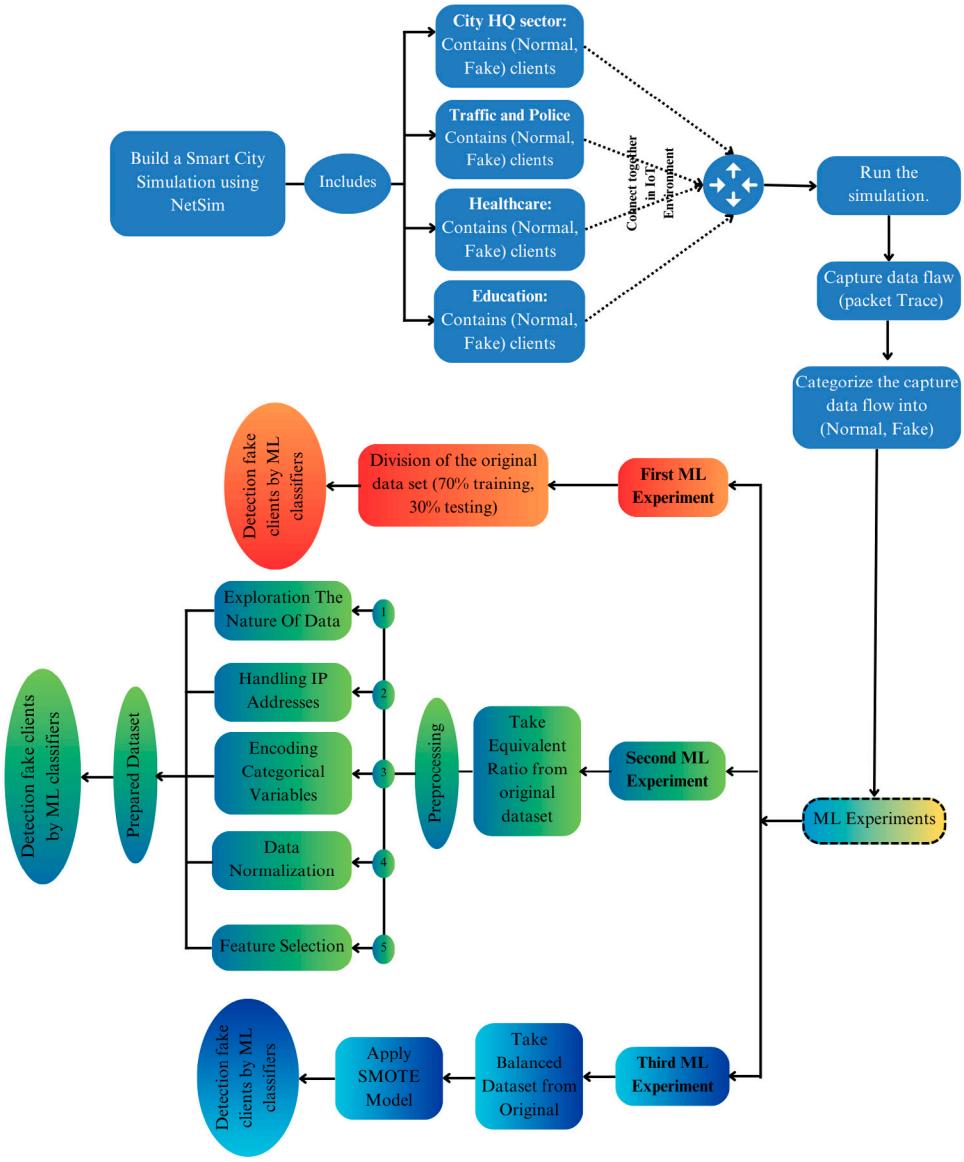


Fig. 1. Proposed methodology.

us to optimize server configurations and caching strategies for improved web service delivery.

CBR Applications: CBR applications play a vital role in evaluating the network's handling of a constant and uninterrupted data flow without rate fluctuations. In our simulation, CBR traffic helps us understand how congestion control mechanisms operate and the impact of congestion on data delivery. We use CBR simulations to study multimedia streaming applications like real-time audio or video streams. Through this analysis, we can assess the quality of multimedia content delivery over the network and optimize mechanisms for smoother and more reliable streaming experiences.

Email Applications: Incorporating email application simulations into our study allows us to gain deeper insights into email traffic patterns and usage behavior. We can better understand email users' behaviors and preferences by analyzing the volume of emails exchanged, email sizes, and communication frequencies. Additionally, this data aids in optimizing email server configurations, ensuring efficient and seamless email delivery and reception.

6lowpan-gateway: The 6lowpan gateway plays a pivotal role in our simulation, bridging the 6lowpan network and other IP-based networks like the Internet. It enables communication between devices operating on low-power and lossy networks, like IoT devices and traditional IP networks. In our setup, the 6lowpan gateway facilitates the HTTP and email traffic exchange between IoT devices and servers, ensuring seamless connectivity and integration into more extensive networks.

Ad Hoc Networking: Ad hoc networking, a decentralized form of wireless communication, plays a crucial part in our simulation. It allows devices to communicate directly without relying on a fixed infrastructure, like a centralized router. In our context, devices in the simulation can form ad hoc networks, enabling direct interactions between IoT devices, 6lowpan-gateways, and other components, fostering efficient and dynamic data exchange.

Router: The router acts as a critical component in our simulation, responsible for forwarding data packets between different networks, including ad hoc networks and traditional IP-based networks. It is critical in ensuring proper data routing and enables seamless communication between IoT devices, 6lowpan-gateways, and other connected devices. The router's efficient routing capabilities are essential in maintaining stable connections and optimizing data transmission across the simulation.

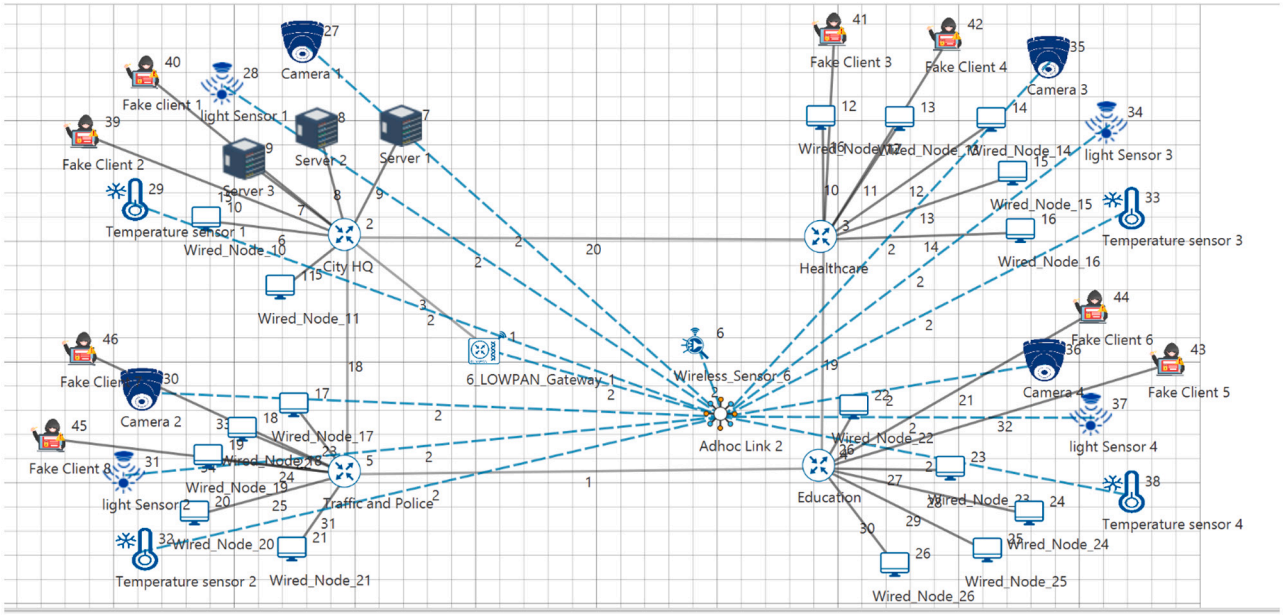


Fig. 2. Our simulation of smart city involving fake clients.

By linking these components sequentially, our simulation creates a holistic environment to study various network conditions, traffic patterns, and communication dynamics. Through this interconnected setup, we gain comprehensive insights into the performance of web applications, multimedia streaming, and email services over diverse network architectures, enabling us to make informed decisions for enhancing overall network efficiency and user experiences.

In our study, we utilized fake clients in a simulation to replicate attackers' tactics, focusing on Distributed Denial of Service (DDoS) attacks, where a network is overwhelmed by traffic from multiple sources, leading to service disruptions. These fake clients generate malicious traffic, targeting critical network components such as servers and routers, causing congestion and hindering access for legitimate users. The dual purpose of introducing fake clients was to assess the network's resilience against cyber threats and analyze its response to excessive traffic, identifying vulnerabilities and failure points. To conduct these cyber-attacks, we deployed eight clients in each smart city sector, complemented by two fake clients per sector. We modified application parameters to emulate a DDoS attack; for HTTP applications, we increased the number of pages to 1000 with a page size of 20,000 bytes, raised the packet size in CBR applications to 200,000 bytes, and set email sizes to 100,000 bytes in email applications. These changes aimed to overwhelm the server with requests, replicating the effects of a DDoS attack. This comprehensive approach allowed us to thoroughly examine the impact of such attacks on network functionality, observing factors like latency, packet loss, and data transfer rates. It provided valuable insights into the network's behavior under attack, as illustrated in Fig. 3 of our study.

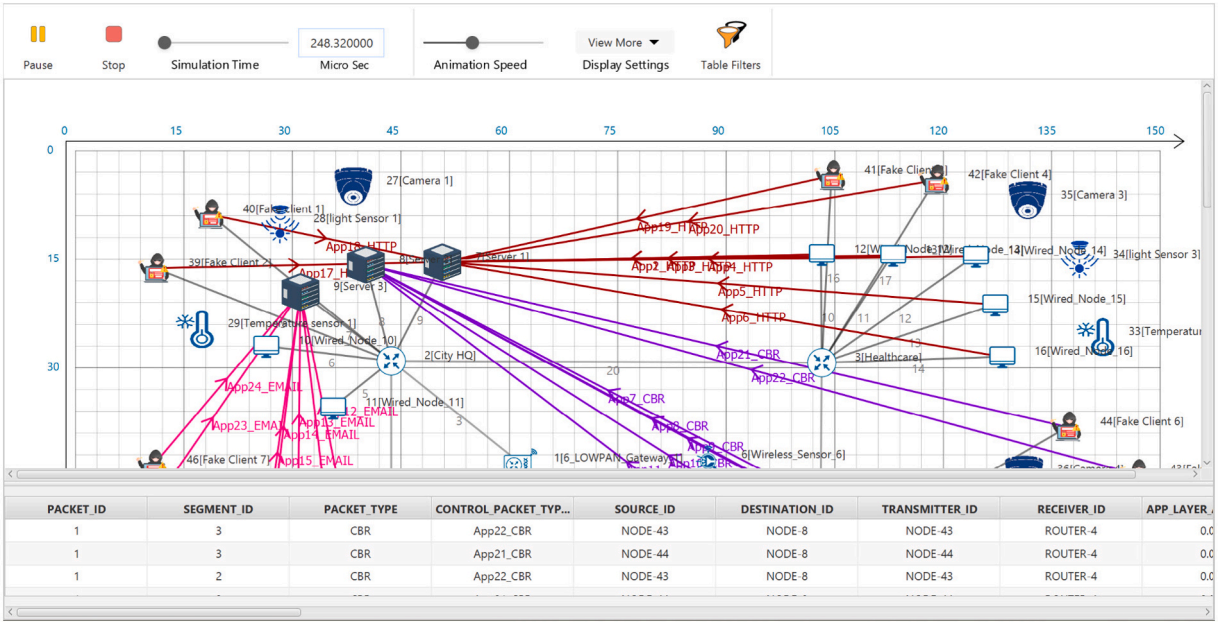
Table 3 provides a comprehensive overview of key network performance indicators related to packet queuing, dequeuing, and packet drops across multiple devices and port IDs. These metrics are necessary for evaluating network efficiency, identifying potential congestion points, and assessing the reliability of data transmission within the network infrastructure. The table's structure starts by enumerating various devices (Device IDs) and their corresponding port IDs (Port ID), serving as a crucial reference for associating specific network segments with their respective queue statistics. The "Queued Packet" column quantifies the number of packets currently awaiting processing within the queue, offering insights into each device and port load. Then, the "Dequeued Packet" column reflects the count of successfully processed and dequeued packets, providing an assessment of packet processing efficiency. Finally, the "Dropped Packet" column highlights the number of packets that have encountered discarding or deletion during processing, shedding light on potential congestion or network irregularities. Notably, Device 4, Port 1, and Port 2, as well as Device 5, Port 2, exhibit higher packet drops, indicating potential congestion or capacity issues that warrant further investigation and optimization efforts to ensure reliable network performance.

Table 3
Device queue metrics.

Device ID	Port ID	Queued packets	Dequeued packets	Dropped packets
1	2	18	18	0
2	1	19	19	0
2	7	9713	9713	0
2	8	407 540	407 540	0
3	1	18	18	0
3	7	715	715	0
4	1	841 886	836 299	541 242
4	2	19	19	0
5	1	18	18	0
5	2	841 911	836 307	13 689

Table 4 presents the throughput values for different applications, with higher values signifying superior data transmission efficiency. Identifying applications with low throughput serves as a basis for potential optimization efforts. Notably, the dataset exhibits a range of throughput values, with the highest recorded at 0.563677 Mbps corresponding to "App11_CBR" between Source ID 26 and Destination ID 8, and the lowest at 0.000120 Mbps attributed to "App1_HTTP" between Source ID 12 and Destination ID 7. Beyond throughput, analyzing delay and jitter values for each application is crucial, as lower values indicate faster and more stable communication. In contrast, higher delay or jitter values may suggest network congestion or other underlying issues requiring investigation. Of significance, "App3_HTTP" between Source ID 13 and Destination ID 7 exhibits the lowest jitter value in the dataset, measuring 0.646122 ms. Conversely, "App1_HTTP" between Source ID 12 and Destination ID 7 presents the highest jitter value in the dataset, reaching 12 408 000.800066 ms. These observations underscore the diverse network performance characteristics revealed by the data, prompting considerations for optimization strategies and potential network enhancements.

Table 5 reflects a detailed account of network behavior across several parameters. The high volume of packets transmitted on critical



PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYP...	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER
1	3	CBR	App22_CBR	NODE-43	NODE-8	NODE-43	ROUTER-4	0.0
1	3	CBR	App21_CBR	NODE-44	NODE-8	NODE-44	ROUTER-4	0.0
1	2	CBR	App22_CBR	NODE-43	NODE-8	NODE-43	ROUTER-4	0.0

Fig. 3. Packets flow of smart city with fake clients.

Table 4
Application metrics.

App ID	Application name	Source ID	Destination ID	Packet generated	Packet received	Payload generated (bytes)	Payload received (bytes)	Throughput (Mbps)	Delay (ms)	Jitter (ms)
1	App1_HTTP	12	7	250	150	40 000	22 500	0.000120	6 204 733.734	12 408 000.800066
2	App2_HTTP	12	7	65	64	48 750	48 000	0.003440	213.019875	7.403476
3	App3_HTTP	13	7	99	99	74 250	74 250	0.005940	206.519968	0.646122
4	App4_HTTP	7	13	1936	209	290 400	197 250	0.023763	4 352 939.247331	385 784.545305
5	App5_HTTP	15	7	99	99	74 250	74 250	0.005940	207.088334	1.978367
6	App6_HTTP	16	7	99	99	74 250	74 250	0.005940	208.497778	2.644490
7	App7_CBR	23	16	5000	4811	7 300 000	7 030 000	0.526976	168 236.782460	4650.084068
7	App7_CBR	23	16	5000	4811	7 300 000	7 030 000	0.526976	168 236.782460	4650.084068
8	App8_CBR	24	8	5000	4812	7 300 000	7 052 360	0.561925	168 336.027689	657.924989
9	App9_CBR	25	8	5000	4812	7 300 000	7 052 360	0.562042	168 330.259172	658.043856
10	App10_CBR	24	8	5000	4812	7 300 000	7 052 360	0.562742	168 310.259213	657.085650
11	App11_CBR	26	8	5000	4826	7 300 000	7 045 680	0.563677	168 423.759017	658.172053
12	App12_EMAIL	17	9	1368	1050	1 941 120	1 541 120	0.121930	541 530.679491	951.852211
13	App13_EMAIL	18	9	1368	559	784 440	738 440	0.062877	426 005.501917	1594.153978
14	App14_EMAIL	19	9	1368	1320	1 956 800	1 584 760	0.157876	2855.682937	241.286257
15	App15_EMAIL	20	9	1368	513	741 320	713 320	0.059086	402 767.349280	1458.860313
16	App16_EMAIL	21	9	1368	505	728 960	696 960	0.058122	400 379.471284	1468.573568
17	App17_HTTP	7	39	1936	0	742 500	0	0.057334	7875.302677	291.445689
18	App18_HTTP	40	7	1936	0	742 500	0	0.057340	7875.306274	291.446596
19	App19_HTTP	41	7	1936	0	742 500	0	0.057341	7875.306871	291.446503
20	App20_HTTP	42	7	1936	0	742 500	0	0.057342	7875.307468	291.446410

links, such as Link_ids 1, 8, and 9, underscores their importance in the network's architecture, potentially identifying them as essential junctions for data flow and thus prime targets for security measures against fake clients. The error rates on certain links, particularly Link_id 18, warrant a closer examination for underlying vulnerabilities or the need for enhanced error management protocols. Notably, the absence of packet collisions across all links suggests high efficiency in network management. Furthermore, the transmission overhead data points to the bandwidth consumed by control signaling, which, while necessary for network regulation, also represents an area for optimization to free up capacity for increased payload transmission.

3.1. Extract dataset

This dataset is prepared to study and detect fake clients' behavior within a simulated IoT smart city environment. It provides a view of packet transmissions and interactions, enabling machine learning researchers to develop, train, and test algorithms to identify fake clients in the network. The dataset contains 960,648 fake clients and 127,927 regular clients; the total number is 1,088,575, with 27 features.

The potential uses of this dataset are varied and impactful. Firstly, it enables the development of machine learning models that can automatically detect and flag fake clients in real time, enhancing security measures in smart city infrastructures. Additionally, by analyzing the transmission patterns within this data, researchers can uncover common characteristics or behaviors of fake clients. This analysis is crucial for understanding and mitigating the risks they pose. Lastly, the dataset is instrumental in enhancing smart city networks' overall security and reliability. It aids in identifying vulnerabilities and potential attack vectors, creating more secure and resilient IoT environments. This dataset provides an extensive resource for detecting fake clients and a foundational tool for strengthening cybersecurity measures in the rapidly evolving domain of smart cities.

Table 6 serves as an essential guide to navigating the intricacies of the dataset. Each entry in the table elucidates the purpose and significance of a particular column, ensuring clarity and understanding for the

Table 5
Link metrics from smart city network simulation.

Link ID	Packet transmitted		Packet errored		Packet collided		Bytes transmitted (bytes)	Payload transmitted (bytes)	Overhead transmitted (bytes)
	Data	Control	Data	Control	Data	Control			
All	5 996 333	317	7237	0	0	0	8 987 068 268	8 706 972 140	280 096 128
1	836 278	38	1026	0	0	0	1 249 635 876	1 224 900 280	24 935 596
2	0	0	0	0	0	0	0	0	0
3	0	37	0	0	0	0	3004	0	3004
4	209 836	0	261	0	0	0	311 023 354	299 317 450	11 705 904
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	19 142	0	19	0	0	0	28 658 308	27 596 520	1 061 788
8	815 450	0	1001	0	0	0	1 239 685 980	1 194 374 780	45 511 200
9	828 857	42	1014	0	0	0	1 228 257 916	1 182 025 330	46 232 586
10	283	42	0	0	0	0	374 278	355 900	18 378
11	306	0	1	0	0	0	389 402	371 290	18 112
12	309	0	0	0	0	0	390 936	374 250	16 686
13	306	0	0	0	0	0	389 402	372 770	16 632
14	309	0	0	0	0	0	390 936	374 250	16 686
15	209 835	0	251	0	0	0	311 023 980	299 325 730	11 698 250
16	209 641	0	237	0	0	0	310 730 704	299 065 090	11 665 614
17	196 554	0	259	0	0	0	291 325 566	280 336 970	10 988 596
18	845 980	39	962	0	0	0	1 263 874 996	1 238 732 740	25 142 256
19	0	37	0	0	0	0	3108	0	3108
20	408 172	82	460	0	0	0	594 073 688	581 972 410	12 101 278
21	6 800 000	0	792	0	0	0	1 036 720 000	996 834 240	37 885 760
22	2769	0	5	0	0	0	4 105 086	3 948 160	156 926
23	2768	0	2	0	0	0	4 103 552	3 951 120	152 432
24	2767	0	6	0	0	0	4 102 018	3 943 720	158 298
25	2771	0	3	0	0	0	4 108 154	3 954 080	154 074
26	5000	0	5	0	0	0	7 570 000	7 292 700	277 300
27	5000	0	10	0	0	0	7 570 000	7 285 400	284 600
28	5000	0	10	0	0	0	7 570 000	7 285 400	284 600
29	5000	0	9	0	0	0	7 570 000	7 286 860	283 140
30	5000	0	7	0	0	0	7 570 000	7 289 780	280 220
31	2765	0	6	0	0	0	4 098 950	3 940 760	158 190

Table 6
Features and descriptions for our dataset.

No.	Feature	Description
1	PACKET_ID	A unique identifier for each packet.
2	SEGMENT_ID	A unique identifier for each segment of the packet.
3	PACKET_TYPE	The type of the packet.
4	CONTROL_PACKET_TYPE_APP_NAME	The name of the application associated with the control packet type.
5	SOURCE_ID	The identifier of the source of the packet.
6	DESTINATION_ID	The identifier of the destination of the packet.
7	TRANSMITTER_ID	The identifier of the transmitter of the packet.
8	RECEIVER_ID	The identifier of the receiver of the packet.
9	APP_LAYER_ARRIVAL_TIME_US_	The arrival time of the packet at the application layer in microseconds.
10	TRX_LAYER_ARRIVAL_TIME_US_	The arrival time of the packet at the transport layer in microseconds.
11	NW_LAYER_ARRIVAL_TIME_US_	The arrival time of the packet at the network layer in microseconds.
12	MAC_LAYER_ARRIVAL_TIME_US_	The arrival time of the packet at the media access control (MAC) layer in microseconds.
13	PHY_LAYER_ARRIVAL_TIME_US_	The arrival time of the packet at the physical layer in microseconds.
14	PHY_LAYER_START_TIME_US_	The start time of the packet processing at the physical layer in microseconds.
15	PHY_LAYER_END_TIME_US_	The end time of the packet processing at the physical layer in microseconds.
16	APP_LAYER_PAYLOAD_Bytes_	The size of the packet payload at the application layer in Bytes.
17	TRX_LAYER_PAYLOAD_Bytes_	The size of the packet payload at the transport layer in Bytes.
18	NW_LAYER_PAYLOAD_Bytes_	The size of the packet payload at the network layer in Bytes.
19	MAC_LAYER_PAYLOAD_Bytes_	The size of the packet payload at the media access control (MAC) layer in Bytes.
20	PHY_LAYER_PAYLOAD_Bytes_	The size of the packet payload at the physical layer in Bytes.
21	PHY_LAYER_OVERHEAD_Bytes_	The overhead size added by the physical layer in Bytes.
22	PACKET_STATUS	The status of the packet.
23	SOURCE_IP	The source IP address of the packet.
24	DESTINATION_IP	The destination IP address of the packet.
25	GATEWAY_IP	The IP address of the gateway.
26	NEXT_HOP_IP	The IP address of the next hop.
27	Node	The node associated with the packet.

dataset's users. From unique identifiers for packets to intricate details of transmission dynamics, the table methodically catalogs each attribute, shedding light on its role within the broader framework of the dataset. Such a detailed breakdown is instrumental for researchers, offering an explicit data structure roadmap. By familiarizing themselves with this table, users can effectively strategize their data analysis, ensuring that they harness the full potential of the dataset in their endeavors to detect and understand fake clients in IoT smart city environments.

3.2. Machine learning classifier

- **Random Forest**: Random Forest is an ensemble learning algorithm that combines multiple decision trees to make predictions.

It randomly selects subsets of data and features for each tree to reduce overfitting. Each tree independently classifies or regresses on the data, and the final prediction is determined by aggregating the results from all trees. The algorithm leverages the "wisdom of the crowd" principle to improve Accuracy and handle large datasets. It is versatile, takes missing values, and provides feature importance measures. Random Forest is widely used for classification and regression tasks in various domains, including finance, healthcare, and image recognition [23].

Random Forest combines the predictions of multiple decision trees by either voting (classification) or averaging (regression). For classification tasks, the mathematical equation for Random Forest can be represented as

$$\hat{y}(x) = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j=1}^{T} w_j \cdot h_j(x)\right) \tag{1}$$

where: $\hat{y}(x)$ represents the predicted output for input x, N denotes the total number of decision trees in the forest, T represents the total number of nodes in each decision tree, w_j represents the weight of the jth tree, and $h_j(x)$ signifies the predicted output of the jth decision tree for input x.

- **Naïve Bayes (NB) classifier:** Naive Bayes is a simple probabilistic classifier based on Bayes' theorem. It assumes that the features are conditionally independent given the class. In other words, each feature contributes independently to the probability of a particular class. It calculates the probability of a class given the features using prior probabilities and likelihoods. The classifier assigns the class with the highest probability as the predicted class. Naive Bayes is computationally efficient and works well with high-dimensional data. However, its feature independence assumption may sometimes limit its performance. It is commonly used in text classification, spam filtering, and recommendation systems [24].

  The Naïve Bayes (NB) classifier is based on Bayes' theorem and assumes independence between features. The mathematical equation for the Naïve Bayes classifier can be expressed as follows [25]:

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(y) \cdot P(x_1|y) \cdot P(x_2|y) \cdot \ldots \cdot P(x_n|y)}{P(x_1, x_2, \ldots, x_n)} \tag{2}$$

  where:

  - $P(y|x_1, x_2, \ldots, x_n)$ is the posterior probability of the class y given the features $x_1, x_2, \ldots, x_n$.
  - $P(y)$ is the prior probability of the class y.
  - $P(x_1|y), P(x_2|y), \ldots, P(x_n|y)$ are the conditional probabilities of each feature $x_1, x_2, \ldots, x_n$ given the class y.
  - $P(x_1, x_2, \ldots, x_n)$ is the probability of observing the features $x_1, x_2, \ldots, x_n$.

  In the Naïve Bayes classifier, the assumption of feature independence allows us to simplify the equation further:

$$P(y|x_1, x_2, \ldots, x_n) = P(y) \cdot P(x_1|y) \cdot P(x_2|y) \cdot \ldots \cdot P(x_n|y) \tag{3}$$

  This equation can be used to calculate the probability of each class given the input features, and the classifier can select the class with the highest probability as the predicted class.

- **Classification and Regression Trees (CART)** is a decision tree-based machine learning algorithm for classification and regression tasks. CART employs binary splitting based on feature values to recursively partition the data, optimizing on criteria like Gini impurity for classification and the sum of squared differences for regression. The tree structure provides intuitive interpretability, making CART suitable for applications where decision transparency is essential. Care is required to manage potential overfitting through pruning and address data instability and class imbalance challenges [26].

  The Gini Impurity is a measure used in the CART algorithm to quantify the disorder or impurity of a dataset concerning its class labels. It provides a metric to evaluate how often a randomly chosen sample would be incorrectly classified if it was randomly labeled based on the distribution of labels in the dataset. The formula for Gini Impurity is [27]:

$$\text{Gini}(p_1, p_2, \ldots, p_k) = 1 - \sum_{i=1}^{k} p_i^2 \tag{4}$$

  where:

  - p_i represents the proportion of the data that belongs to class i.
  - k is the total number of classes.

- **The k-Nearest Neighbors (KNN)** is a supervised machine learning algorithm utilized for both classification and regression tasks. It predicts an output based on the majority vote or average of its k closest data points from the training set. KNN is computationally intensive, requiring distance calculations for each test point against the entire dataset, making feature scaling crucial for accurate results. The choice of k and distance metric can significantly influence its performance. While KNN's simplicity and adaptability are strengths, it is essential to consider its sensitivity to irrelevant features and the storage requirements of retaining the whole dataset [28].

  The core of the KNN algorithm revolves around the computation of the distance between data points. Euclidean distance is one of the most commonly used distance metrics in the context of KNN [29].

  For two data points in a 2-dimensional space, $P(x_1, y_1)$ and $Q(x_2, y_2)$, the Euclidean distance d between them is given by:

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{5}$$

  For an n-dimensional space, where the two points are $P(x_1, x_2, \ldots, x_n)$ and $Q(y_1, y_2, \ldots, y_n)$, the formula generalizes to:

$$d(P, Q) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \tag{6}$$

  In the context of KNN, once the distances from a test point to all training points are computed using the formula above (or another distance metric), the k smallest distances are selected. A majority vote (for classification) or average (for regression) of their corresponding outputs is taken as the prediction.

- **Long Short-Term Memory (LSTM) classifier:** is a type of recurrent neural network (RNN) designed to analyze sequential data, such as time series or text. When applied to a CSV dataset, an LSTM classifier processes the data by considering the sequential relationships between rows or entries. It utilizes a network architecture composed of LSTM cells that can capture and remember long-range dependencies in the data. Each row of the CSV dataset is treated as a time step, and the LSTM cells within the classifier maintain internal states that allow them to remember or forget information over time selectively. This enables the LSTM classifier to learn patterns, trends, and contextual information within the dataset, making it practical for tasks like classification where the order of data is essential.

3.3. Evaluation metrics

The dataset split allocated 70% for training and 30% for testing. The evaluation used confusion matrix-based metrics: TP, FP, FN, and TN. Four key measures were applied: Accuracy, Precision, Recall (or Sensitivity), and F1-score to assess model performance [30]. Accuracy

indicates overall prediction correctness; Precision is the ratio of correct optimistic predictions, Recall is the identification rate of actual positives, and the F1-score balances Precision and Recall.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{8}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{9}$$

$$\text{F1-score} = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision} \tag{10}$$

3.4. Experiments

In the previous stage, we created the simulation of the environment proposed in this paper, which presents a smart city with many clients, including (normal and fake), and we produced our dataset. This part presents several experiments on this dataset to prepare a robust model for detecting fake clients. So, we conducted the following experiments:

3.4.1. First experiment

This experiment, implemented as Benchmarking and Initial Assessment, was oriented towards establishing a foundational understanding of how various machine learning algorithms fare in identifying fake IoT clients without any preprocessing interventions on the inherently imbalanced dataset. This stage was crucial for setting a benchmark for detection capabilities, aiming to answer the pressing research question: How effectively can different machine learning algorithms pinpoint fake IoT clients in a simulated smart city environment using raw, unprocessed data. We hypothesized that, although machine learning algorithms possess an inherent capacity to detect fake IoT clients to a certain extent, the initial performance metrics would reveal substantial opportunities for refinement, particularly in achieving a balanced sensitivity and specificity. This hypothesis underscores our expectation that raw data, with its intrinsic imperfections and class imbalances, would present a challenging yet insightful starting point for our exploration.

In the first experiment, we will examine the original data set resulting from the simulation, which contains 960,648 fake clients and 127,927 normal clients, and the total number of them is 1,088,575, with 27 features. as shown in Table 6. Then, we divide the data set into two parts. 70% training and 30% testing.

3.4.2. Second experiment

Moving into the second experiment, our focus shifted towards the Impact of Preprocessing techniques and their influence on ML algorithms' performance with fake client detection. Specifically, we explored the effects of encoding categorical variables, converting IP addresses into a machine-readable format, and normalizing the data to improve algorithmic interpretations. The central research question guiding this experiment was: What impact do specific preprocessing techniques have on the accuracy, precision, recall, F-measure, and AUC of machine learning models in the context of fake IoT client detection? Anticipating the outcomes, we posited that applying preprocessing would result in marked enhancements across all evaluated performance metrics, attributing this improvement to ameliorating data quality and the heightened relevance of features for machine learning analysis. This experiment demonstrated the transformative potential of methodical data preprocessing in elevating model performance.

In this experiment, we will use an equivalent ratio of the original data set of 100,000 records, including 90 836 fake and 9164 normal, to facilitate dealing with the Huge data and preserve the original dataset form. Then we apply preprocessing techniques to make the dataset more suitable for AI classifiers to enhance the results of detection, The preprocessing phase ensures that the dataset is primed for machine learning algorithms. The raw data, abundant in information, can also be riddled with redundancies, inconsistencies, and inaccuracies. Through preprocessing, we refine this data, bolstering the predictive capability of our models. After that, we remove data duplication. Then, we applied the same classification algorithms used before; the goal was to take an equivalent ratio of the original data set to speed up the classification process and preserve the original characteristics. As for the feature selection process, the goal is to increase the classification accuracy. The following illustrates each step of this experiment:

1. **Equivalent Ratio** The decision to select an equivalent ratio of 100,000 records for the second experiment in our study was guided by a blend of statistical rigor and the practical necessity of managing the huge size of the original dataset. Given the vast and complex nature of the data captured from our simulation, processing the entire dataset due to the limited capabilities of the personal computers on which we experiment is considered the main challenge. To address this, we determined that an equivalent ratio of 100,000 records would strike an optimal balance between computational feasibility and the maintenance of analytical integrity. This size was deemed manageable for efficiently conducting machine learning experiments without sacrificing the quality of analysis. Our methodology for achieving a statistically representative subset involved a statistical sampling approach that ensured the preservation of interaction distributions between real and fake clients within the dataset. By identifying key variables and proportionally sampling records based on their distribution, we maintained the diversity necessary for training robust machine learning models capable of generalizing well to unseen data. This careful approach to sampling underscored our commitment to both efficiency and statistical validity, ensuring that the selected subset was practical for analysis and rich in the diversity of scenarios necessary for detecting fake client behaviors effectively as shown in Algorithm 1.
2. **Handling IP Addresses:**
   The dataset uniquely contained IPv4 addresses in columns such as `SOURCE_IP`, `DESTINATION_IP`, `GATEWAY_IP`, and `NEXT_HOP_IP`. The direct conversion was applied to these addresses to:
   - Transform each IP address into a distinct integer representation.
   - Retain the specificity of each IP while presenting it in a numerical format amenable to machine learning algorithms.
3. **Encoding Categorical Variables** Machine learning models, by their nature, operate on numerical data. This is because the mathematical computations and algorithms underlying these models require numerical values for processing. However, real-world datasets often contain categorical data, essentially non-numerical values representing different categories or classes.
   In our dataset, certain features like `PACKET_TYPE`, `CONTROL_PACKET_TYPE_APP_NAME`, `SOURCE_ID`, `DESTINATION_ID`, `TRANSMITTER_ID`, `RECEIVER_ID`, and `PACKET_STATUS` were categorical in nature. These features had different categories or classes represented as strings or symbols.
   Label Encoding is a technique that converts each unique category in a feature into a distinct integer. For example, categories "black", "Blue", and "Green" might be encoded as 0, 1, and 2. This method ensures consistent mapping, meaning "black" will always be 0. It also keeps the data compact, as the encoded feature remains a single column without expanding the dataset's dimensions.

4. **Data Normalization**
   Each feature can have its scale or range of values in datasets with multiple features. For instance, one feature might range between 0 and 1, while another could vary between 0 and 1000. This discrepancy in scales can pose challenges when modeling, as algorithms may inadvertently assign more importance to features with larger scales, even if they are not necessarily more informative. The Standard Scaler method is a solution to this problem. It is a normalization technique that adjusts each feature to have a mean (average) of 0 and a standard deviation (a measure of data spread) of 1.

Algorithm 1 Sample Dataset Based on Value Distribution

```
▷ Input: Original.CSV dataset
▷ Output: Partial dataset saved in 'equivalent_ratio.CSV'

algorithm SampleDataset
    ▷ Read the CSV file into a data structure
    data ← load('Original.CSV')
    ▷ Identify the target column 'Node' irrespective of case
    variables ← column names of data
    for name in variables do
        if name matches 'Node' then
            node_column_name ← name
            break
        end if
    end for
    targetValues ← data[node_column_name]

    ▷ Calculate the distribution of values
    uniqueValues, counts ← unique values and their counts in targetValues
    percentages ← counts/sum of counts

    ▷ Determine sample sizes for each unique value
    numSamples ← round(percentages × 100000)

    ▷ Create a partial dataset with proportional representation
    partialData ← empty data structure
    for i from 1 to length of uniqueValues do
        value ← uniqueValues[i]
        indices ← find(targetValues == value)
        samples ← randomly select numSamples[i] from indices
        for index in samples do
            partialData ← partialData + data[index]
        end for
    end for

    ▷ Save the partial dataset to a new CSV file
    save(partialData,' equivalent_ratio.CSV')
end algorithm
```

5. **Feature Selection**
   The filter feature selection method is used in machine learning and data mining to identify a dataset's most relevant and informative features (variables or attributes). The "filter" method involves filtering the features based on their characteristics rather than considering the interactions between features or the specific learning algorithm used. The filter feature selection method applies a statistical measure or scoring metric to each feature in the dataset. These measures assess the correlation or dependency between each feature and the target variable without considering the relationship between the features themselves. The features are then ranked or scored based on their relevance to the target variable [31].
   The Mutual Information (MI) metric was employed for our dataset. MI is a non-parametric method that measures the dependency between variables. Specifically, it quantifies the information gained about one variable through another. A higher MI score implies that knowing the value of the feature can provide more information about the target variable, so we adopted values more significant than the mean of MI values as a threshold to select most feature effects in the classification. Fig. 4 present two bar charts; (a) presents the MI for all features, and (b) the MI values for the selected features.

3.4.3. Third experiment

The final experiment in our series sought to address the challenge of class imbalance through the implementation of the Synthetic Minority Over-sampling Technique (SMOTE), aiming to evaluate its impact on enhancing the detection accuracy of fake IoT clients. This endeavor was driven by the research question: How does the balancing of the dataset with SMOTE influence the performance of ML algorithms in detecting fake IoT clients? Our hypothesis posited that the strategic balancing of the dataset would significantly amplify the models' capability to accurately identify fake clients, as evidenced by improved accuracy and AUC scores. This expectation was grounded in the notion that addressing the bias towards the majority class, a common issue in imbalanced datasets, would be instrumental in fostering more equitable and effective model training processes.

In the previous experiment, a conspicuous dataset imbalance was observed, characterized by a preponderance of fake records compared to their regular counterparts. This observed disparity may raise concerns regarding the equity of the dataset. A sequence of preprocessing procedures will be undertaken to achieve a balanced dataset that ensures equitable representation of record categories. The process involves the removal of duplicate records, followed by applying the SMOTE, a data augmentation method in machine learning that generates synthetic examples for the minority class by interpolating between existing instances, effectively addressing the class imbalance. A stochastic reshuffling of the data will also be conducted, and a randomized subset of 127,927 samples will be extracted for both the fake and normal categories, achieving a perfect dataset. Subsequently, the same classification models employed in the antecedent experiment will be applied.

4. Results and discussion

This section comprehensively analyzes the outcomes derived from a series of ML experiments designed to detect fake client activities within a simulated IoT smart city environment. This section delves into the performance of various sophisticated algorithms ranging from the ensemble-based Random Forest to the intricate Long Short-Term Memory networks. the discussion interprets the nuanced improvements and the algorithms' adaptability to the evolving complexities of the data. We explore the implications of these results, examining the role of data preprocessing and the challenges posed by imbalanced datasets. Table 7 presents the results of the adopted ML experiments. To evaluate the effectiveness of the machine learning models, several performances were analyzed:

- Accuracy: Measures the overall correctness of the model by calculating the ratio of correctly predicted instances to the total instances.
- Precision: Indicates the proportion of true positive predictions among all positive predictions made by the model.
- Recall (Sensitivity): Measures the model's ability to correctly identify true positive instances from all actual positive instances.
- F1-score: Provides a harmonic mean of precision and recall, offering a balance between the two.

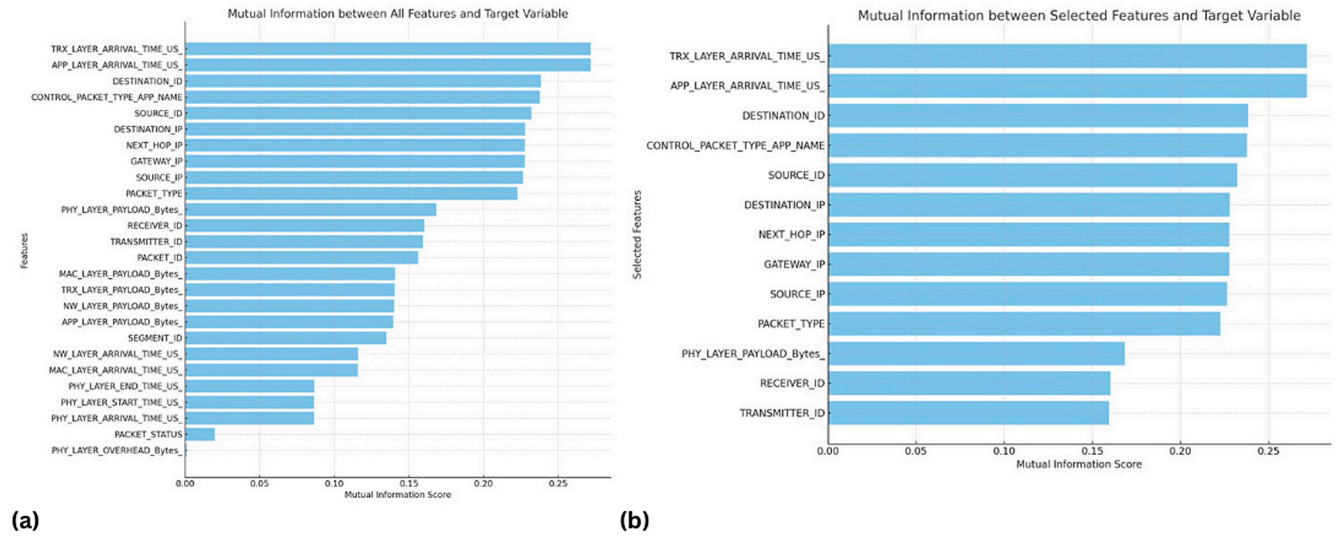


Fig. 4. (a) MI scores of all features. (b) MI scores of all selected features.

Table 7
Comparison of machine learning algorithm performance across three experiments.

Algorithm	Accuracy (%)			Precision (%)			Recall (%)			F_measure (%)			AUC (%)		
	First Exp	Second Exp	Third Exp	First Exp	Second Exp	Third Exp	First Exp	Second Exp	Third Exp	First Exp	Second Exp	Third Exp	First Exp	Second Exp	Third Exp
RF	94.02	98.00	98.68	96.52	88.77	97.86	96.75	89.23	99.10	96.63	89.00	98.47	98.8	99.77	98.83
CART	93.88	97.98	98.34	96.50	88.87	97.75	96.56	89.42	98.42	96.53	89.14	98.09	84.92	98.40	99.23
NB	96.29	97.98	95.30	95.97	89.61	98.70	1.000	88.21	90.22	97.94	88.91	94.27	98.53	98.51	99.25
KNN	94.38	97.94	98.77	96.37	89.24	97.89	97.29	88.14	99.25	96.83	88.68	98.57	99.32	93.53	99.81
LSTM	96.21	98.06	96.85	92.44	96.05	98.52	99.80	85.92	94.06	97.89	89.06	96.24	95.6	99.77	99.78

- Area Under the Curve (AUC): Represents the model's ability to discriminate between classes, with higher values indicating better performance.

In the quest to enhance the detection of fake clients within a simulated smart city environment, our study meticulously examined various machine learning algorithms, uncovering the profound influence of preprocessing on the proposed detection model's effectiveness. Through three experiments, we observed nuanced shifts in performance metrics—Accuracy, Precision, Recall, F-measure, and AUC—underscoring the pivotal role of data quality in refining detection capabilities.

The initial foray into this investigation, Experiment 1, leveraged the original dataset obtained from the simulation, spotlighting the NB classifier for its remarkable Recall of 100%, signifying its excellent sensitivity to detecting fake clients, with a fairly good accuracy of 96.29%. However, We acknowledge that the outcomes achieved thus far are moderately satisfactory. Nonetheless, there is potential for enhancement through the implementation of preprocessing techniques. Preprocessing is a widely recognized and essential practice in Artificial Intelligence (AI), designed to refine the dataset, thus rendering it more suitable for classification via ML algorithms. Notably, our observations from the second experiment confirm that applying these preprocessing methods has indeed contributed to improving detection results.

Experiment 2 marked a turning point in a narrative as diverse preprocessing techniques, from encoding categorical variables to data normalization, were introduced. In coordination with the preprocessing, the received results have demonstrated a substantial change. Specifically, the accuracy of the Random Forest algorithm skyrocketed from 94.02% to an impressive 98.00%. This represented better discriminative capabilities of the algorithm and the impact of preprocessing on the noise removal and maintenance of feature relevance. At the same time, the precision of the Long Short-Term Memory networks also significantly increased — from 92.44% to 96.05%. The enhanced performance of the LSTM network was achieved due to the clarity and organization of the data provided by preprocessing, which helped the network establish complex, temporal dependencies within the layers assigned to the noise.

Experiment 3 revamped the narrative by focusing on the class imbalance through the Synthetic Minority Over-sampling Technique in the pursuit of the k-Nearest Neighbors. This approach helped the KNN algorithm to demonstrate the peak of performance with an Accuracy of 98.68% and an AUC score of a whopping 99.81%. This result showed that with preprocessed, balanced classes, the noise reduction was so significant that both the sensitivity and the specificity increased, making KNN the best algorithm in fake client detection. Meanwhile, The LSTM network reduced the accuracy to 96.85% but upheld an AUC score. This nuanced conclusion demonstrated significant variability in how balanced classes impacted algorithms and showed that token tactics should be used when training the model. Experiment comparison allowed us to uncover rich insights into the machine learning mechanics in cybersecurity. The initial experiment's hardship of class imbalance and raw data transformed into the structured analysis of preprocesses in the following steps, outlining the path to improvement. Preprocessing in shaping the data and class balancing were deemed essential learning points in raising the model's accuracy and precision. The class balancing unveiled the path to the equilibrium between fake client detection and understanding false alarms, giving the ground for an appropriate precision–recall balance. In this regard, this study demystified the differences in the performance of multiple machine learning algorithms, advocated for preprocessing, and discovered the importance of the class works for fake client detection in smart city applications. Thus, the data preparation, algorithm identification, postfix noise assessment, and balanced class approach were identified as the components of an effective fake-client detection strategy in smart cities through this narrative.

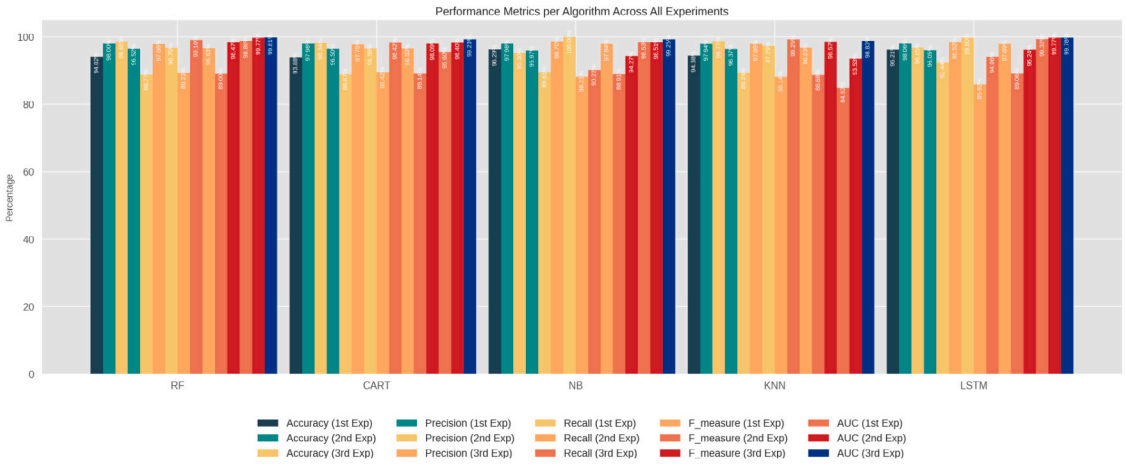


Fig. 5. Packets flow of smart city with fake clients.

Table 8
Comparison of our work with the related work.

Authors	Dataset used	ML classifier	Highest accuracy
[22]	Collected dataset	XGBoost	93.8%
[32]	–	Logistic Regression	92.14%
[33]	KDD Cup 99	Decision tree	92.9%
Proposed model	Simulation dataset	KNN	98.77%

The bar chart in Fig. 5 demonstrates the performance of five ML classifiers, RF, CART, NB, KNN, and LSTM, across three experiments in detecting fake clients. Performance generally improves by the third experiment, suggesting the positive impact of data preprocessing and balancing. While NB shows a high initial Recall that later drops, LSTM consistently scores well in Accuracy. The overall trend indicates that careful data preparation significantly enhances model effectiveness, particularly in complex scenarios like smart city simulations. demonstrated

Fig. 6 presents ROC curves for the ROC curve for the best classifier across each experiment, NB, KNN, and LSTM classifiers. Upon reviewing the AUC values, LSTM leads marginally with a score of 99.78, closely followed by KNN at 99.77, indicating excellent model performance and discriminative capability. While slightly trailing with an AUC of 98.53, NB also demonstrates predictive solid power. The proximity of KNN and LSTM ROC curves to the top-left corner reflects their high accurate positive rates and low false favorable rates, affirming their efficacy in accurately classifying instances in each experiment conducted.

Table 8 contrasts machine learning methods for detecting fake clients. Author [22] reached 93.8% accuracy with an XGBoost model on a custom dataset, showcasing XGBoost's adaptability. Author [32], using an unspecified dataset and Logistic Regression, achieved 92.14% accuracy, demonstrating the model's effectiveness despite its simplicity. Author [33] utilized the KDD Cup 99 dataset with a decision tree to attain 92.9% accuracy, reflecting the decision tree's capability with non-linear data. Distinguished from these, the proposed model applied a KNN classifier to a simulation dataset, achieving a high accuracy of 98.77%, indicating KNN's proficiency in a simulated smart city context. This suggests that well-crafted simulations aligned with advanced classifiers like KNN can significantly enhance model performance, emphasizing the importance of dataset selection and classifier choice in smart city modeling.

5. Conclusion

The proliferation of the IoT in urban landscapes has accentuated the necessity for verifying the legitimacy of interconnected devices. Addressing the predicament of identifying counterfeit IoT devices in a mock smart city setting, this study embarked on three separate experiments, each emphasizing varied preprocessing and data harmonization methodologies. While the inaugural experiment delved directly into the dataset, the subsequent trial incorporated preprocessing and feature discernment, and the final one concentrated on rectifying dataset imbalances via the SMOTE. An array of evaluation matrices revealed the efficacy of data normalization and feature selection in bolstering the prowess of machine learning paradigms. Augmenting the dataset's equilibrium using SMOTE further elevated this efficacy. Among the gamut of assessed algorithms, KNN emerged as a stalwart performer. Yet, distinct algorithms like NB and LSTM manifested their respective merits. Building on these insights, a subsequent algorithm was conceived to respond to these predictions, effectively sidelining and ostracizing deceptive devices, underscoring the pragmatic application of the research. This culminates in an earnest call to action for prompt preventive measures to fortify the security fabric of smart city infrastructures.

Future work and limitations

Future research should expand the scope of attack types beyond simulated DDoS attacks, including Man-in-the-Middle (MitM) attacks, data manipulation, unauthorized access, and botnet attacks. Leveraging advanced deep learning models, such as CNNs, RNNs, LSTMs, and GANs, can further enhance detection capabilities. Real-world implementation and testing in smart city environments are crucial for practical applicability. This study has limitations, including reliance on simulations. Future work should explore more powerful computational infrastructure. Incorporating Quantum Key Distribution (QKD) to enhance IoT security, as discussed by Golec et al. (2024) [34], can provide secure communication channels against fake clients. Advanced cryptographic techniques, such as homomorphic encryption and zero-knowledge proofs, are essential for maintaining data privacy. Quantum cloud computing can improve data processing capabilities, enhancing the efficiency and accuracy of machine learning algorithms for detecting fake clients in real-time. Addressing these challenges and integrating these technologies will contribute to the development of secure, efficient, and resilient smart city infrastructures.

Ethical approval

This article does not contain any studies with human participants or animals performed by any authors.

Informed consent

Informed consent was obtained from all individual participants included in the study.

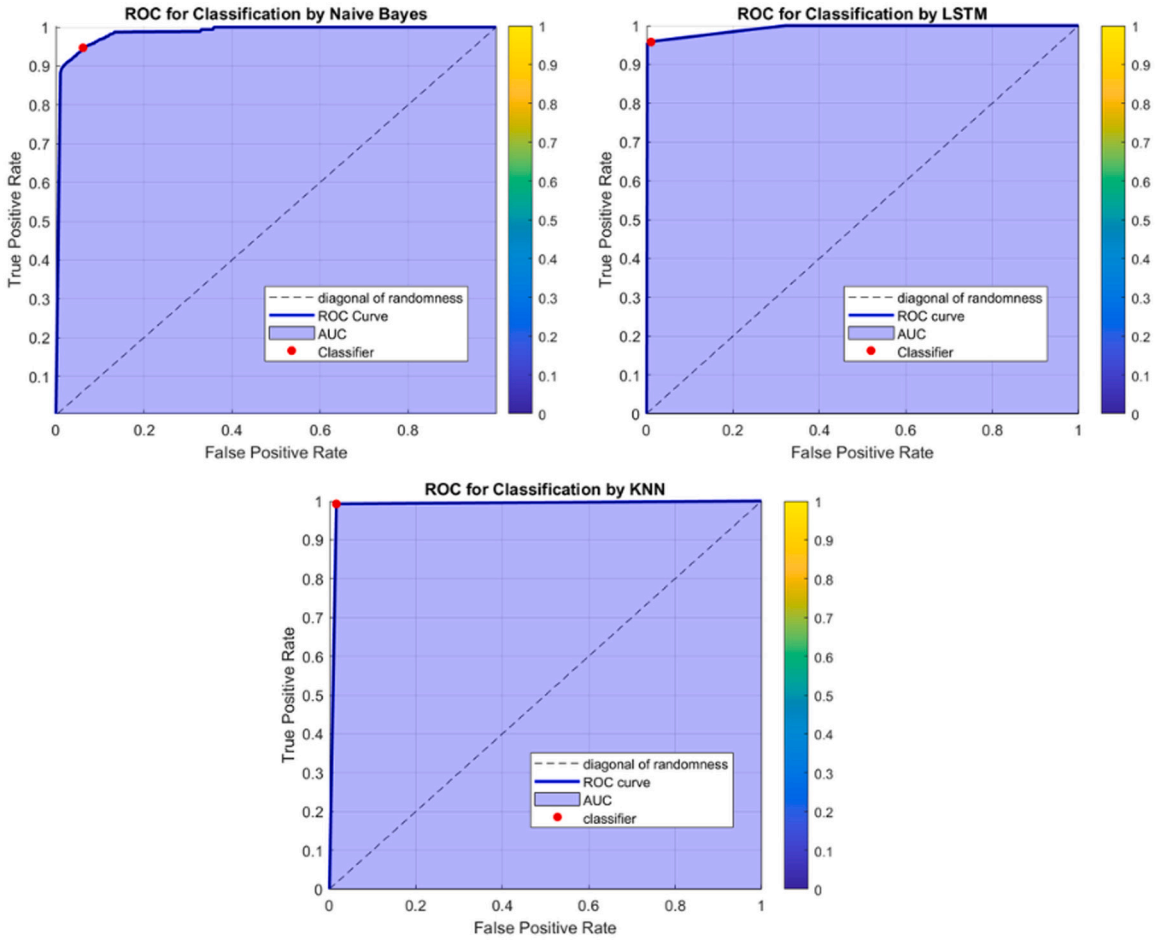


Fig. 6. ROC curve for the best classifier across each experiment.

CRediT authorship contribution statement

Mahmoud AlJamal: Software, Resources, Writing – original draft, Supervision, Methodology, Conceptualization, Formal analysis, Writing – review & editing. **Ala Mughaid:** Supervision, Methodology, Conceptualization, Writing – original draft. **Bashar Al shboul:** Supervision, Methodology, Conceptualization, Writing – original draft. **Hani Bani-Salameh:** Supervision, Methodology, Conceptualization, Writing – original draft. **Shadi Alzubi:** Supervision, Methodology, Conceptualization, Writing – original draft. **Laith Abualigah:** Supervision, Methodology, Conceptualization, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] S. Madakam, V. Lake, V. Lake, V. Lake, et al., Internet of things (IoT): A literature review, J. Comput. Commun. 3 (05) (2015) 164.

[2] A. Mughaid, A. Alqahtani, S. AlZu'bi, I. Obaidat, R. Alqura'n, M. AlJamal, R. AL-Marayah, Utilizing machine learning algorithms for effectively detection IoT DDoS attacks, in: International Conference on Advances in Computing Research, Springer, 2023, pp. 617–629.

[3] F. Zhu, Y. Lv, Y. Chen, X. Wang, G. Xiong, F.-Y. Wang, Parallel transportation systems: Toward IoT-enabled smart urban traffic control and management, IEEE Trans. Intell. Transp. Syst. 21 (10) (2019) 4063–4071.

[4] M. Aljamal, A. Mughaid, R. Alquran, M. Almiani, S. AlZu'bi, Simulated model for preventing IoT fake clients over the smart cities environment, in: 2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), IEEE, 2023, pp. 0757–0761.

[5] A.A.A. Sen, M. Yamin, Advantages of using fog in IoT applications, Int. J. Inf. Technol. 13 (2021) 829–837.

[6] L. Abualigah, A.M. Hussein, M.H. Almomani, R.A. Zitar, H. Migdady, A.I. Alzahrani, A. Alwadain, Improved jaya synergistic swarm optimization algorithm to optimize task scheduling problems in cloud computing, Sustain. Comput.: Inform. Syst. (2024) 101012.

[7] L. Abualigah, Metaheuristic Optimization Algorithms: Optimizers, Analysis, and Applications, Elsevier, 2024.

[8] M. Songhorabadi, M. Rahimi, A. MoghadamFarid, M.H. Kashani, Fog computing approaches in IoT-enabled smart cities, J. Netw. Comput. Appl. 211 (2023) 103557.

[9] F. Younas, A. Raza, N. Thalji, L. Abualigah, R.A. Zitar, H. Jia, An efficient artificial intelligence approach for early detection of cross-site scripting attacks, Decis. Anal. J. 11 (2024) 100466.

[10] P.M. Rao, B. Deebak, Security and privacy issues in smart cities/industries: technologies, applications, and challenges, J. Ambient Intell. Humaniz. Comput. (2022) 1–37.

[11] A. Puliafito, G. Tricomi, A. Zafeiropoulos, S. Papavassiliou, Smart cities of the future as cyber physical systems: Challenges and enabling technologies, Sensors 21 (10) (2021) 3349.

[12] L. Abualigah, A. Forestiero, M.A. Elaziz, Bio-inspired agents for a distributed NLP-based clustering in smart environments, in: International Conference on Soft Computing and Pattern Recognition, Springer, 2021, pp. 678–687.

[13] J. Xiao, X. Pan, J. Liu, J. Wang, P. Zhang, L. Abualigah, Load balancing strategy for SDN multi-controller clusters based on load prediction, J. Supercomput. 80 (4) (2024) 5136–5162.

[14] L. Abualigah, S. AlNajdawi, A.M. Ikotun, A. Forestiero, F. Gul, A.E. Ezugwu, H. Jia, M. Zare, S. Mahajan, M. Alshinwan, Quantum approximate optimization algorithm: a review study and problems, Metaheuristic Optim. Algor. (2024) 147–165.

[15] F.G. Brundu, E. Patti, A. Osello, M. Del Giudice, N. Rapetti, A. Krylovskiy, M. Jahn, V. Verda, E. Guelpa, L. Rietto, et al., IoT software infrastructure for energy management and simulation in smart cities, IEEE Trans. Ind. Inform. 13 (2) (2016) 832–840.

[16] H. Zahmatkesh, F. Al-Turjman, Fog computing for sustainable smart cities in the IoT era: Caching techniques and enabling technologies-an overview, Sustain. Cities Soc. 59 (2020) 102139.

[17] F. Casino, C. Patsakis, E. Batista, O. Postolache, A. Martínez-Ballesté, A. Solanas, Smart healthcare in the IoT era: A context-aware recommendation example, in: 2018 International Symposium in Sensing and Instrumentation in IoT Era, ISSI, IEEE, 2018, pp. 1–4.

[18] S. Djahel, R. Doolan, G.-M. Muntean, J. Murphy, A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches, IEEE Commun. Surv. Tutor. 17 (1) (2014) 125–151.

[19] K. Selvakumar, M. Karuppiah, L. SaiRamesh, S.H. Islam, M.M. Hassan, G. Fortino, K.-K.R. Choo, Intelligent temporal classification and fuzzy rough set-based feature selection algorithm for intrusion detection system in WSNs, Inform. Sci. 497 (2019) 77–90.

[20] O. Barybin, E. Zaitseva, V. Brazhnyi, Testing the security ESP32 internet of things devices, in: 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology, PIC S&T, IEEE, 2019, pp. 143–146.

[21] X. Cao, N.Z. Gong, Mpaf: Model poisoning attacks to federated learning based on fake clients, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3396–3404.

[22] M. Chowdhury, B. Ray, S. Chowdhury, S. Rajasegarar, A novel insider attack and machine learning based detection for the internet of things, ACM Trans. Internet Things 2 (4) (2021) 1–23.

[23] A. Subasi, E. Alickovic, J. Kevric, Diagnosis of chronic kidney disease by using random forest, in: CMBEBIH 2017: Proceedings of the International Conference on Medical and Biological Engineering 2017, Springer, 2017, pp. 589–594.

[24] E.M.K. Reddy, A. Gurrala, V.B. Hasitha, K.V.R. Kumar, Introduction to naive Bayes and a review on its subtypes with applications, Bayesian Reason. Gaussian Process. Mach. Learn. Appl. (2022) 1–14.

[25] D. Berrar, Bayes' theorem and naive Bayes classifier, Ency. Bioinform. Comput. Biol.: ABC Bioinform. 403 (2018) 412.

[26] R.-C. Chen, C. Dewi, S.-W. Huang, R.E. Caraka, Selecting critical features for data classification based on machine learning methods, J. Big Data 7 (1) (2020) 52.

[27] X. Gao, C. Shan, C. Hu, Z. Niu, Z. Liu, An adaptive ensemble machine learning model for intrusion detection, IEEE Access 7 (2019) 82512–82521.

[28] S. Uddin, I. Haque, H. Lu, M.A. Moni, E. Gide, Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction, Sci. Rep. 12 (1) (2022) 6256.

[29] K. Shah, H. Patel, D. Sanghvi, M. Shah, A comparative analysis of logistic regression, random forest and KNN models for the text classification, Augment. Hum. Res. 5 (2020) 1–16.

[30] K. Anjana, S. Urolagin, Churn prediction in telecom industry using machine learning algorithms with K-best and principal component analysis, in: Applications of Artificial Intelligence in Engineering: Proceedings of First Global Conference on Artificial Intelligence and Applications, GCAIA 2020, Springer, 2021, pp. 499–507.

[31] M. Cherrington, F. Thabtah, J. Lu, Q. Xu, Feature selection: filter methods performance challenges, in: 2019 International Conference on Computer and Information Sciences, ICCIS, IEEE, 2019, pp. 1–4.

[32] A. Mehbodniya, J.L. Webber, M. Shabaz, H. Mohafez, K. Yadav, Machine learning technique to detect sybil attack on IoT based sensor network, IETE J. Res. (2021) 1–9.

[33] R. Vinayakumar, M. Alazab, K.P. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman, Deep learning approach for intelligent intrusion detection system, IEEE Access 7 (2019) 41525–41550, http://dx.doi.org/10.1109/ACCESS.2019.2895334.

[34] M. Golec, E.S. Hatay, M. Golec, M. Uyar, M. Golec, S.S. Gill, Quantum cloud computing: Trends and challenges, 2024, arXiv preprint arXiv:2404.19612.