

LOUD COMPUTING-BASED SECURITY ANALYSIS ON WIRELESS SENSOR NODES CLUSTER USING PREDICTIVE TECHNIQUE

MUHAMMED ZAHARADEEN AHMED^{1,2*}, AISHA HASSAN ABDALLAH HASHIM^{1,3},
OTHMAN OMRAN KHALIFA^{1,4}, ALIYU MUHAMMAD WAKIL², ZEINAB E. AHMED¹,
KHMAIES OUAHADA³

¹Dept. of Electrical and Computer Engineering, International Islamic University Malaysia, Malaysia

²Dept. of Computer Engineering, University of Maiduguri, Borno State, Nigeria

³Dept. of Electrical and Electronic Engineering Science, University of Johannesburg, South Africa.

⁴Libyan Center for Engineering Research and Information Technology, Bani Waleed, Libya

*Corresponding author: zaharadeencna@gmail.com

(Received: 6 August 2024; Accepted: 4 February 2025; Published online: 15 May 2025)

ABSTRACT: Rapid technological advancements have led to the widespread deployment of wireless sensor networks (WSNs) in industrial environments, making cybersecurity a critical concern in cloud computing. This paper presents a predictive framework for cloud-based intrusion detection and prevention for WSNs. It integrates machine learning models—Multilayer Perceptron (MLP), Decision Tree, and Autoencoder—to precisely classify and mitigate various impacts of cyber intrusions on a cluster of wireless sensors. An intelligent prioritization and prevention system is also proposed, categorizing attacks—blackhole, grayhole, flooding, and scheduling—based on their impact on industrial processes. Experimental results indicate robust detection capabilities, with the Decision Tree achieving 99.48% accuracy, slightly outperforming MLP at 99.37%. The Autoencoder demonstrated superior binary classification, distinguishing between normal and anomalous instances with high precision and recall rates. This framework leverages the WSN-DS dataset to simulate and validate its efficiency in mitigating real-time threats. Future work will focus on refining the prioritization model and integrating advanced machine learning techniques for enhanced adaptability and resilience.

ABSTRAK: Kemajuan pesat dalam teknologi telah membawa kepada penggunaan meluas rangkaian penerima wayarles (WSN) dalam persekitaran industri, menjadikan keselamatan siber sebagai kebimbangan kritikal dalam pengkomputeran awan. Kajian ini membentangkan rangka kerja ramalan bagi mengesan dan mencegah pencerobohan berasaskan awan untuk WSN. Ia menyepadukan model pembelajaran mesin—Perseptron Berbilang Lapis (MLP), Pokok Keputusan (Decision Tree) dan Enkoder Automatik (Autoencoder)—bagi klasifikasi tepat dan pengurangan pelbagai kesan pencerobohan siber pada kelompok penerima wayarles. Sistem keutamaan dan pencegahan pintar turut dicadangkan, mengkategorikan serangan—lubang hitam, lubang kelabu, banjir dan penjadualan—berdasarkan kesan terhadap proses industri. Dapatan eksperimen menunjukkan keupayaan pengesanan yang mantap dengan Decision Tree mencapai ketepatan 99.48%, sedikit mengatasi prestasi MLP pada 99.37%. Autoencoder menunjukkan klasifikasi binari yang unggul, membezakan antara kejadian biasa dan anomali dengan ketepatan tinggi dan kadar ingatan semula. Rangka kerja ini memanfaatkan set data WSN-DS bagi simulasi dan pengesanan kecekapan dalam mengurangkan ancaman masa nyata. Kajian akan menumpukan pada memperhalusi model keutamaan dan menyepadukan teknik pembelajaran mesin lanjutan pada masa hadapan bagi kebolehsuaian dan daya tahan yang tinggi.

KEYWORDS: *Wireless Sensor Networks, Cloud, Security, Deep learning, and Predictive technique.*

1. INTRODUCTION

Wireless Sensor Networks (WSNs) have emerged as a critical technology in various domains, including environmental monitoring, healthcare, industrial automation, and military applications. However, the clusters of wireless sensor nodes are vulnerable to multiple security threats due to their inherent characteristics, such as limited computational power, energy constraints, and wireless communication. With the increasing adoption of cloud computing, it becomes essential to explore its potential for enhancing the security of WSNs. Technological innovations like artificial intelligence (AI), machine learning, and augmented reality are integral to Industry 4.0. These technologies provide intelligent assistance, streamline processes, and increase productivity across various industrial tasks [1]. A Wireless Sensor Network (WSN) is the interconnection of wireless sensors and devices. These networks are vital for collecting, transmitting, and analyzing real-time data essential for process optimization and decision-making. WSNs act as the industrial setup's nerve center, gathering data on motion, temperature, pressure, and other parameters. They enable seamless communication between devices, systems, and people, offering greater scalability and flexibility than traditional wired systems. WSNs can be easily expanded, reconfigured, and adapted to various industrial environments [2]. However, WSNs face several Internet security challenges. The increased number of device connections raises vulnerability to cyberattacks, necessitating strong cybersecurity measures to protect sensitive data. Signal interference and reliability issues in complex industrial environments affect network performance. Using battery-powered sensors poses challenges, as maintaining prolonged battery life and energy efficiency is crucial for consistent and reliable operation.

Due to the convergence of digital technologies, WSNs are crucial for real-time data transfer. This research is motivated by the dynamic and evolving nature of cyber threats. This requires framework development with well-defined priorities. Several Internet attacks pose a greater danger and have a more significant impact on the cloud environment. This research addresses the dynamic nature of cyber threats in WSNs, proposing a framework that leverages predictive techniques to prioritize and mitigate attacks based on their severity and impact. Unlike traditional reactive systems, this framework aims to proactively detect and neutralize threats, ensuring minimal disruption to industrial processes.

Integrating cloud computing with wireless sensor networks (WSNs) has become increasingly prevalent among researchers due to the enhanced processing capabilities and scalable resources that the cloud infrastructure provides. This integration simplifies collecting, storing, and analyzing large volumes of data generated by sensor nodes. However, the simplification also introduces additional security challenges, especially in situations involving limited computational resources and energy constraints of sensor nodes. To ensure data transmission security, storage, and processing, it is critical to have high integrity, confidentiality, and availability of sensitive information. This study leverages predictive techniques like machine learning and data analytics to identify potential security threats and vulnerabilities within a wireless sensor network. These algorithms can improve security by predicting and mitigating attacks before a system is compromised. However, implementing such an algorithm involves examining complex issues, such as trade-offs between security and performance, the accuracy of predictive models, and the efficient management of limited resources.

There are numerous open issues related to this research. These include the resource constraints of sensor nodes. The sensor nodes are challenged by processing power, memory, and battery life in this context. These constraints must be considered when implementing security mechanisms and predictive techniques to avoid node overload and resource depletion. Also, data storage and transmission challenges must be addressed by proposing an effective strategy or algorithm to minimize energy consumption while ensuring data integrity and security. The security of the internet cloud is paramount. This means data integrity and confidentiality must be assured when they are generated from the sensors. They must not be tampered with or be vulnerable to any threats during transmission. A Robust Access Control management strategy must be implemented to prevent unauthorized access to sensitive data stored in the cloud. The network system must have a better detection and mitigation strategy against attacks, such as denial-of-service (DoS), eavesdropping, and data tampering. Addressing these open issues in this paper will develop secure and efficient cloud computing-based systems that manage wireless sensor node clusters. Using predictive techniques can be a promising avenue to enhance security.

2. LITERATURE REVIEW

This section reviews relevant literature on cloud computing-based security analysis for wireless sensor node clusters using predictive techniques. Integrating cloud computing with WSNs offers numerous benefits, including scalable storage, robust computational capabilities, and centralized management. However, the distributed nature and resource limitations of WSNs introduce significant security challenges. Recent studies highlight applying predictive techniques, such as machine learning and data analytics, to provide proactive security measures by identifying potential threats before they cause harm. This review also examines current trends, challenges, and solutions associated with leveraging cloud computing and predictive approaches for WSN security.

2.1. Security Threats in WSN

Wireless Sensor Networks (WSNs) are inherently vulnerable to various cybersecurity threats due to their distributed nature, limited computational resources, energy constraints, and typical deployment in hostile or unattended environments. Common threats include eavesdropping, where unauthorized parties intercept data transmissions, potentially leading to privacy breaches or data manipulation [3]. Another significant threat is node capture, where attackers physically access nodes to extract sensitive information, such as cryptographic keys, which can then be used to launch further attacks or manipulate network operations [4], [5]. Denial-of-Service (DoS) attacks overload networks or specific nodes with traffic, disrupting services and depleting resources, rendering nodes unavailable for legitimate tasks.

More advanced threats include blackhole attacks, where compromised nodes drop all received packets, causing data loss and potential network isolation, and grayhole attacks, which selectively alter or drop packets, complicating threat detection [6]. Sybil attacks involve a malicious node presenting multiple identities to disrupt data aggregation, voting, or routing protocols. Wormhole attacks tunnel messages between network locations to create false views of the network, misleading routing protocols and disrupting operations. Sinkhole attacks occur when a compromised node falsely advertises an optimal route to attract traffic, leading to data interception or denial of service. Additionally, Hello flood attacks exploit routing protocols by sending or replaying numerous "Hello" packets, causing energy depletion as nodes attempt to respond.

Mitigation strategies for these threats include encryption, authentication, intrusion detection systems (IDS), secure routing protocols, redundancy, and multipath routing. However, given the resource constraints in WSNs, these solutions must be lightweight and efficient to ensure their practicality [6].

2.2. Cloud Computing in WSNs

Integrating cloud computing with Wireless Sensor Networks (WSNs) brings significant advantages, opening new possibilities for data management, analysis, and application deployment [7]. By leveraging the cloud's computational power and storage capabilities, this integration addresses WSNs' inherent limitations, such as constrained processing power, memory, and energy efficiency. One of the key benefits of cloud computing in WSNs is its scalability, enabling administrators to efficiently handle the growing volumes of data generated by large-scale WSN deployments without overwhelming local resources.

Cloud platforms also provide robust storage and data management capabilities, allowing for long-term storage of sensor data and facilitating historical data analysis using tailored applications [8]. Furthermore, cloud-based solutions enhance data processing and analytics, offering powerful computational resources for real-time processing, machine learning, and big data analytics to extract actionable insights. The cloud's remote accessibility further enables the centralized management and monitoring of WSNs without requiring physical proximity. However, these benefits come with potential security vulnerabilities, such as unauthorized access and data breaches, highlighting the necessity for secure transmission protocols and reliable intrusion detection mechanisms to safeguard sensitive data.

2.3. Predictive Security Techniques

Predictive security techniques utilize advanced analytics, machine learning (ML), and artificial intelligence (AI) to anticipate and mitigate cyber threats before they occur. These techniques identify vulnerabilities, anomalous patterns, and emerging threats by analyzing historical and real-time data [9]. To identify potential security risks, threat intelligence and analysis aggregate data from various sources, such as threat feeds, dark web monitoring, and incident reports. In application, threat intelligence helps create predictive models recognizing early indicators of potential attacks or vulnerabilities.

Machine learning and AI-based anomaly detection algorithms identify deviations from normal network traffic and system process behavior. This approach is effective in detecting zero-day attacks, insider threats, and other unknown risks by recognizing unusual patterns that may signal malicious activity. Specifically, network traffic patterns are analyzed to detect anomalies, suspicious communications, or signs of a security breach [10]. This helps identify command-and-control communications, data exfiltration, and other malicious activities.

In predictive security, machine learning models, such as neural networks, decision trees, and autoencoders, are particularly effective in identifying abnormal patterns in network traffic. These models enable proactive defenses, reducing the likelihood of successful cyberattacks. Research into predictive techniques for security analysis, especially in wireless sensor networks (WSNs), has shown promising results for threat identification and mitigation. For example, a machine learning-based approach for anomaly detection in WSNs, using supervised learning algorithms to classify normal and abnormal behaviors, is presented in [14]. The study demonstrates the high accuracy of predictive models in detecting intrusions. Additionally, [15] examines neural networks for intrusion detection in WSNs, proposing a hybrid model that combines feature selection with neural network classifiers to enhance detection performance while minimizing computational overhead. Furthermore, predictive analytics in cloud

environments, focusing on cloud-based machine learning models to analyze large volumes of sensor data, is demonstrated in [15], aiming to predict and prevent security incidents in real-time.

2.4. Summary of Related Work

Cloud computing-based analysis of security has been leveraged in WSNs to overcome resource constraints. It is also leveraged to mitigate scalable storage and enable sophisticated data processing on the internet. In [11], the authors develop a cloud-based architecture to manage WSNs. It also highlights how cloud services can centralize data storage and processing. In addition, it examines offloading computational tasks from resource-limited sensor nodes. Research in [12] discusses the advantages of using cloud computing for data aggregation and analytics. This facilitates additional efficiency in network management and decision-making processes. However, considering security threats and challenges in WSNs, some vulnerabilities of sensors with limited energy resources, inefficient computational capabilities, and susceptibility to physical tampering have been extensively studied. Some common security threats in WSNs are examined in [13]. These include eavesdropping, data injection, node compromise, and denial-of-service (DoS) attacks. In the research, more emphasis is placed on lightweight security mechanisms to operate within WSN constraints.

Table 1. Summary of Related Works

Ref	Contribution	Results	Limitation
[1]	The paper examines potential replay-attack vulnerabilities in SECS/GEM systems by proposing a detection and prevention system against attacks.	Achieved an effective mechanism that identifies and prevents replay attacks on SECS/GEM communications.	However, the authors were unable to propose an effective mechanism that resists security attacks such as DoS, forgery, modification, and man-in-the-middle attacks. These are suggested for future work.
[2]	The paper proposed a hierarchical intrusion detection algorithm to group sensor nodes based on functional assignment to lessen energy consumption during threat detection.	Presented an ideal system for WSNs with limited resources. The detection interval is moderate, and detection accuracy is high.	However, the authors were unable to enhance the detection of multiple intrusion patterns.
[3]	The paper presented a framework that combines preventative and deterrent strategies to reduce the danger of insider attacks.	Achieved a better regulatory framework on information security ethics.	However, the authors were unable to examine intrusion detection and prevention. The focus was only centered on the human and behavioral aspects of cyber misconduct and security.
[4]	The paper presented a detection strategy for creating an active defense system using deception technology. The framework can be used to conceptualize a Hybrid Threat Model.	Illustrated how deception is used to validate network resilience. Using in-network deception for threat detection exhibits how attack information can be generated to accelerate incident response and strengthen network defenses.	However, the authors were unable to manage the issues of reactive defenses, such as intrusion detection systems, which are prone to false positives and potentially lead to analyst alert fatigue and decreased effectiveness.

2. METHODOLOGY

This section provides a framework design for developing a cloud-based architecture that uses predictive techniques to integrate WSN with the Internet cloud service. This includes a

dataset aggregation design pattern using a communication protocol to effectively transmit data from a sensor node to the Internet cloud and prevent attacks and intrusions. During implementation, integrating machine learning algorithms such as decision trees, support vector machines, multilayer perceptrons (MLP), autoencoders, and neural network models to accurately detect and classify different types of attacks, including blackhole, grayhole, flooding, and scheduling attacks, is also conducted. This can potentially predict security threats on sensor data. In terms of implementation, we deployed the designed predictive model in a testbed environment using a cluster of wireless sensor nodes and the Internet cloud.

2.1. Design of the Proposed Methodology

The design describes the dataset, system architecture, block diagram of the system, flow chart, and components required for successful implementation.

2.1.1. Dataset

The models implemented in this framework will be evaluated with WSN-DS. This open-source dataset from Kaggle is for intrusion detection systems in wireless sensor networks. This dataset simulates various denial-of-service (DoS) attacks in WSNs using the LEACH (Low Energy Adaptive Clustering Hierarchy) protocol. We further determine the impact of the framework dataset by dividing it into two sections. These are: the training set and the test set. The training set comprises 80% of the total records in the dataset and will be used to train our models. However, the test set comprises 20% of the records and will be used to test and validate the model.

Id	Time	Is_CH	who CH	Dist_To_C	ADV_S	ADV_R	JOIN_S	JOIN_R	SCH_S	SCH_R	Rank	DATA_S	DATA_R	Data_Sen	dist_CH	T	send_cod	Expanded E	Attack type
101000	50	1	101000	0	1	0	0	25	1	0	0	0	1200	48	130.0854	0	2.4694	Normal	
101001	50	0	101044	75.32345	0	4	1	0	0	1	2	38	0	0	0	0	4	0.06957	Normal
101002	50	0	101010	46.95453	0	4	1	0	0	1	19	41	0	0	0	0	3	0.06898	Normal
101003	50	0	101044	64.85231	0	4	1	0	0	1	16	38	0	0	0	0	4	0.06673	Normal
101004	50	0	101010	4.83341	0	4	1	0	0	1	25	41	0	0	0	0	3	0.06534	Normal
101005	50	0	101010	31.91198	0	4	1	0	0	1	18	41	0	0	0	0	3	0.06717	Normal
101006	50	0	101044	24.34167	0	4	1	0	0	1	5	38	0	0	0	0	4	0.06214	Normal
101007	50	0	101010	26.75033	0	4	1	0	0	1	21	41	0	0	0	0	3	0.06662	Normal
101008	50	0	101044	63.66485	0	4	1	0	0	1	17	38	0	0	0	0	4	0.06649	Normal
101009	50	0	101000	32.90217	0	4	1	0	0	1	12	48	0	0	0	0	1	0.07903	Normal
101010	50	1	101010	0	1	0	0	30	1	0	0	0	1230	41	108.7716	0	2.3611	Normal	
101011	50	0	101044	13.17446	0	4	1	0	0	1	10	38	0	0	0	0	4	0.0613	Normal
101012	50	0	101044	48.16567	0	4	1	0	0	1	13	38	0	0	0	0	4	0.06425	Normal
101013	50	0	101010	66.9102	0	4	1	0	0	1	16	41	0	0	0	0	3	0.07263	Normal
101014	50	0	101010	31.69105	0	4	1	0	0	1	17	41	0	0	0	0	3	0.06716	Normal
101015	50	0	101010	21.52629	0	4	1	0	0	1	8	41	0	0	0	0	3	0.06654	Normal
101016	50	0	101010	74.73928	0	4	1	0	0	1	4	41	0	0	0	0	3	0.0749	Normal
101017	50	0	101044	27.78157	0	4	1	0	0	1	29	38	0	0	0	0	4	0.06139	Normal
101018	50	0	101010	25.5197	0	4	1	0	0	1	26	41	0	0	0	0	3	0.06618	Normal
101019	50	0	101044	41.21473	0	4	1	0	0	1	28	38	0	0	0	0	4	0.0628	Normal

Figure 1. Intrusion Detection Systems Dataset for Wireless Sensor Networks

2.1.2. Block Diagram

The block diagram provides a high-level overview of the components' interactions within the wireless sensor network system. The primary components include the Raspberry Pi 3 microcontroller, sensors, power source, and the MCP 3008 Analog-to-Digital converter.

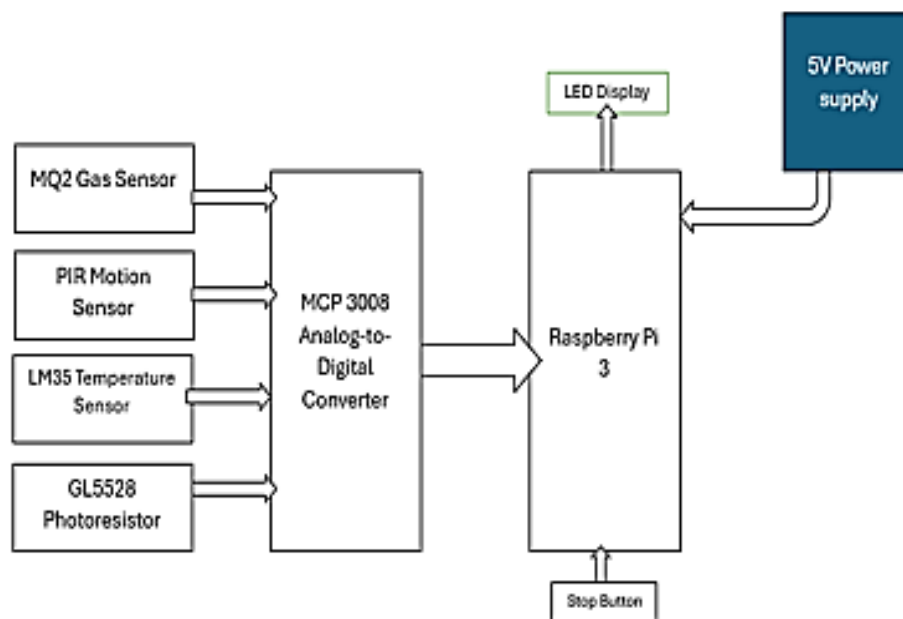


Figure 2. Block Diagram for Cloud-based WSN using Deep learning

Based on Figure 2, the Raspberry Pi microcontroller implements the intrusion detection and prevention logic. The Raspberry Pi is ideal for this project due to its affordability, versatility, and built-in connectivity, which make it suitable for prototyping and deploying IoT-based systems. It has extensive community support that provides valuable resources for troubleshooting and development. The Raspberry Pi has built-in Wi-Fi and Ethernet connectivity, which is essential for creating networked systems. This feature is vital for simulating real-world Industry 4.0 environments, where seamless communication between devices is a cornerstone. The Raspberry Pi can also run various AI and machine learning frameworks, such as TensorFlow and scikit-learn. This compatibility is essential for implementing the AI-driven detection and classification models integral to the project. Using the Raspberry Pi allows for the development of a prototype that can be scaled and implemented in real-world industrial settings. Its portability and ease of use make it an ideal choice for testing and demonstrating the proposed cybersecurity framework.

For the MCP 3008 Analog-to-Digital Converter, since the Raspberry Pi is fundamentally a digital device, any I/O done through its GPIO pins will happen through high (one) and low (zero) states. When input signals are analog (as in the case of our WSNs), they need to be converted to the digital domain so the Raspberry Pi can understand them. The MCP3008 is a 10-bit 8-channel analog-to-digital converter chip that performs this operation.

The Stop Button is connected to one of the Raspberry Pi's GPIO pins. Its function serves as an interrupt/end button for the code's implementation. The algorithm is set to run continuously until a high-priority threat is detected and preventive actions are implemented. The user can end the program/system at their convenience by providing a stop button.

Based on the prototype sensors, the WSN comprises four different sensor types, each of which outputs an analog signal. The choice of these four specific sensors was made with several considerations. Each sensor is meant to simulate one of the four attack types. This framework is trained to conduct smart detection, which means blackhole, flooding, gray hole, and scheduling attacks.

2.1.3. Flowchart

The flowchart outlines the sequential steps in the system's operation, from data acquisition to decision-making and action execution, as shown in Figure 3.

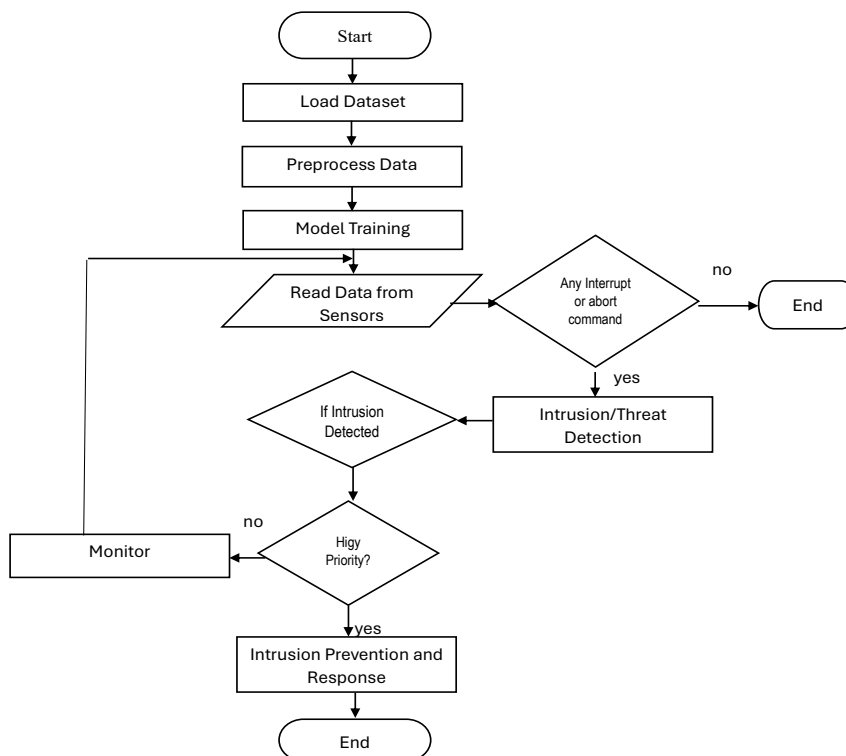


Figure 3. Flowchart for cloud-based WSN using deep learning

We initialize the Raspberry Pi microcontroller and other design components in the simulation process. The dataset from a different Excel application is loaded and uploaded using the import command on the WSN-DS dataset. For data preprocessing, we run commands such as clean, normalize, and prepare a dataset for analysis. This includes extracting features (X) and labels (y) from the data frame. Data is split into training and testing sets, and standardization is applied. The next process is model training. This is where the core logic of the framework is implemented. Three models are used to train the framework comprehensively: the autoencoder, decision tree, and multilayer perceptron. We present additional information for the model training discussion in this paper's implementation section. Also, data is read from the sensors and preprocessed. If there is no command from the user to halt execution, then proceed to intrusion detection; otherwise, terminate. We then check input from sensors against the trained data from the models. If the result comes back as positive, i.e., intrusion/threat is detected, then check for the type of attack and the priority assigned to it. The priority of an attack depends on what kind of industry the system is being used in, and as such, is dynamic. Also, if the attack is of high priority, preventive actions should be immediately implemented, and the program should end. Then, if the attack is of lower priority, raise a warning and closely monitor the inputs from that sensor while continuing to take inputs.

2.2. Implementation

Implementing the proposed methodology involves using a simulation tool to model the cloud-based intrusion detection and prevention framework using predictive techniques and a

deep learning algorithm on wireless sensors. We also conduct performance evaluations of some parameters to deploy on a functional system prototype.

2.2.1. Simulation Tool

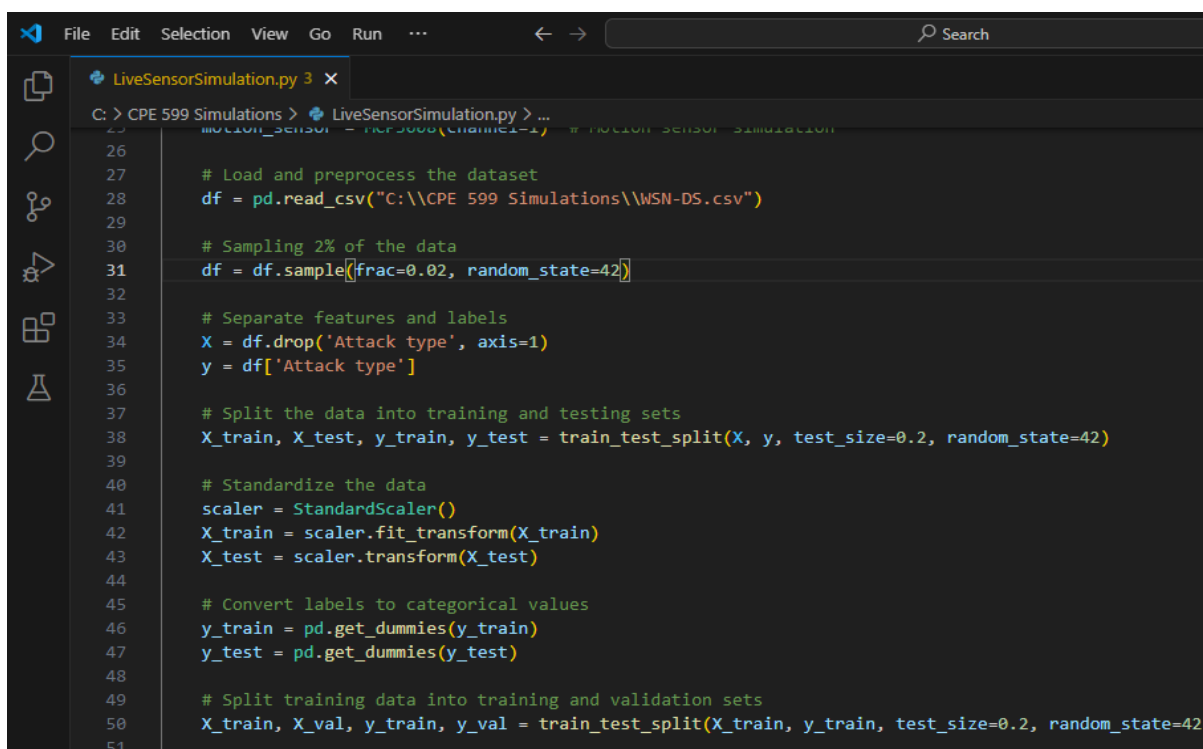
The implementation utilizes a QEMU-based simulation environment to replicate real-world scenarios. Python libraries, including TensorFlow and scikit-learn, support developing and testing machine learning models. The simulation evaluates the system's performance under various cyberattack scenarios, ensuring its robustness and reliability.

2.2.2. Data Collection & Preprocessing

The WSN-DS dataset underwent preprocessing steps to improve data quality and consistency. These steps included:

- Normalization: Ensuring uniform data scales.
- Feature Extraction: Identifying relevant attributes for model training.
- Splitting: Dividing the dataset into 80% training and 20% testing sets.

This is done to have reliable information supplied on the wireless sensors fully secured in the cloud environment. Both real-time and historical data are used. The process of loading the dataset and preprocessing for the wireless sensors is presented based on the script in Figure 4.



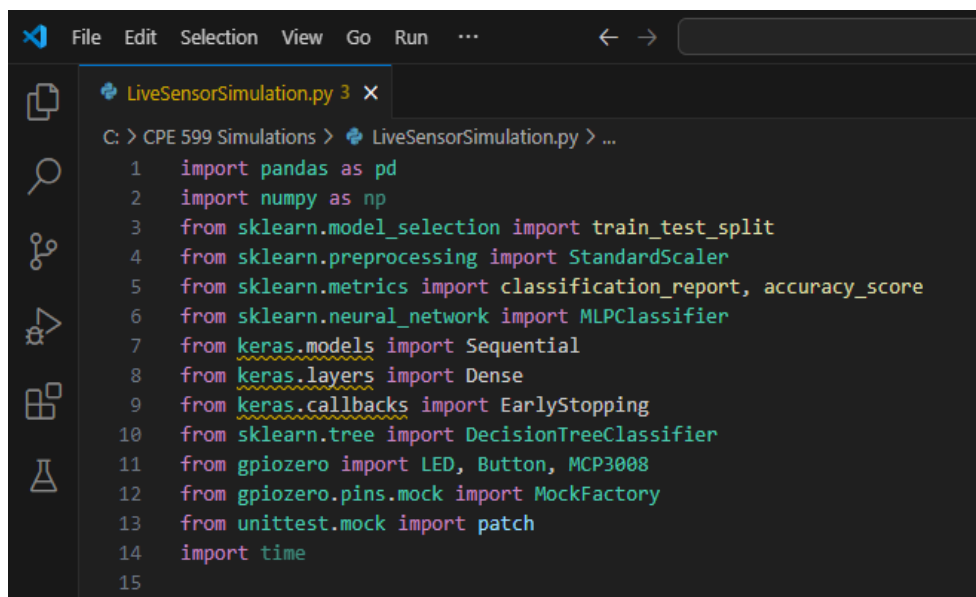
```
26
27 # Load and preprocess the dataset
28 df = pd.read_csv("C:\\CPE 599 Simulations\\WSN-DS.csv")
29
30 # Sampling 2% of the data
31 df = df.sample(frac=0.02, random_state=42)
32
33 # Separate features and labels
34 X = df.drop('Attack type', axis=1)
35 y = df['Attack type']
36
37 # Split the data into training and testing sets
38 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
39
40 # Standardize the data
41 scaler = StandardScaler()
42 X_train = scaler.fit_transform(X_train)
43 X_test = scaler.transform(X_test)
44
45 # Convert labels to categorical values
46 y_train = pd.get_dummies(y_train)
47 y_test = pd.get_dummies(y_test)
48
49 # Split training data into training and validation sets
50 X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
51
```

Figure 4. Loading the WSN-DS dataset and preprocessing

The historical data helps understand patterns, while real-time data detects ongoing threats in the cloud environment. Data preprocessing includes cleaning, filling gaps, removing duplicates, correcting inconsistencies, and normalizing data for machine learning. Based on cybersecurity techniques and cloud services, extracting relevant features from raw data involves various algorithm choices and identifying potential threats or anomalies.

2.2.3. AI-Based Detection and Classification for Cloud Intrusions

An AI-driven intrusion detection and classification system safeguards Industry 4.0 WSNs. Three different AI models, namely, Decision Tree, Multilayer Perceptron (MLP), and Autoencoder, are implemented using a publicly available WSN dataset. Cybersecurity intrusions are detected and classified with a particular emphasis on flooding, scheduling, black hole, and gray hole attacks. Each selected model has a distinct function in recognizing and categorizing cyberattacks, as shown in Figure 5. The Python library and dynamic AI libraries were leveraged due to their strong AI modeling support, particularly (sklearn, Tensorflow, Pandas, Numpy, and Keras).

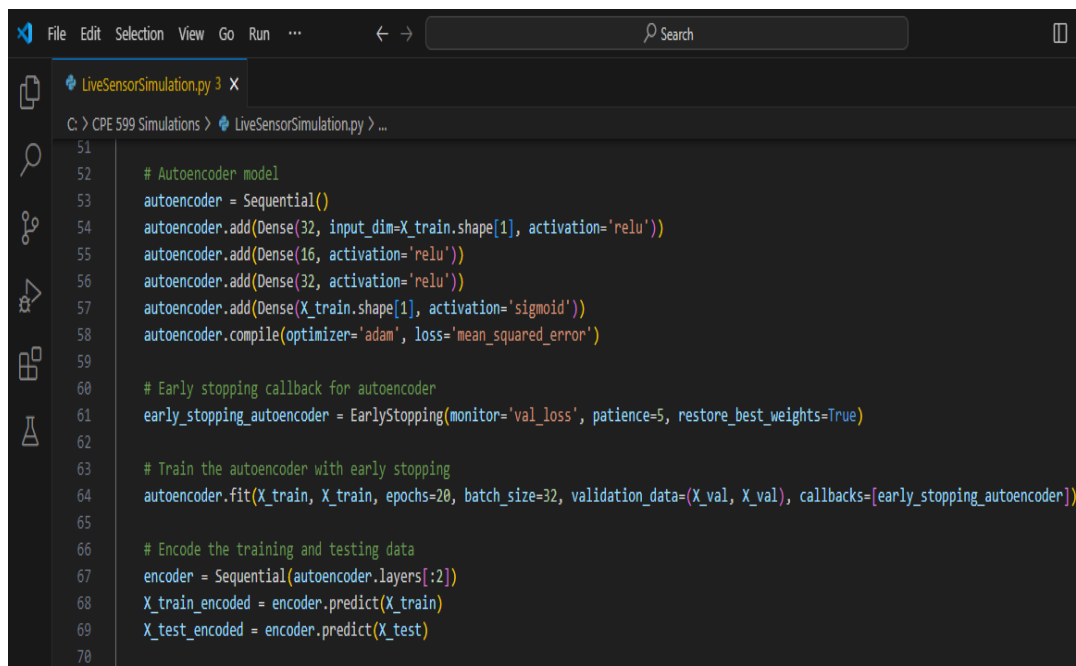


```
File Edit Selection View Go Run ...
LiveSensorSimulation.py 3 x
C: > CPE 599 Simulations > LiveSensorSimulation.py > ...
1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.metrics import classification_report, accuracy_score
6 from sklearn.neural_network import MLPClassifier
7 from keras.models import Sequential
8 from keras.layers import Dense
9 from keras.callbacks import EarlyStopping
10 from sklearn.tree import DecisionTreeClassifier
11 from gpiozero import LED, Button, MCP3008
12 from gpiozero.pins.mock import MockFactory
13 from unittest.mock import patch
14 import time
15
```

Figure 5. Importing AI and Machine Learning Libraries

2.2.4. The Autoencoder

This unsupervised learning approach is applied to data compression and feature learning. It is a powerful feature that presents adequate results for anomaly detection in the cloud environment. It also recognizes anomalies or deviations from the learned patterns by recreating the input data, as adopted in [1].



```
51
52 # Autoencoder model
53 autoencoder = Sequential()
54 autoencoder.add(Dense(32, input_dim=X_train.shape[1], activation='relu'))
55 autoencoder.add(Dense(16, activation='relu'))
56 autoencoder.add(Dense(32, activation='relu'))
57 autoencoder.add(Dense(X_train.shape[1], activation='sigmoid'))
58 autoencoder.compile(optimizer='adam', loss='mean_squared_error')
59
60 # Early stopping callback for autoencoder
61 early_stopping_autoencoder = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
62
63 # Train the autoencoder with early stopping
64 autoencoder.fit(X_train, X_train, epochs=20, batch_size=32, validation_data=(X_val, X_val), callbacks=[early_stopping_autoencoder])
65
66 # Encode the training and testing data
67 encoder = Sequential(autoencoder.layers[:2])
68 X_train_encoded = encoder.predict(X_train)
69 X_test_encoded = encoder.predict(X_test)
70
```

Figure 6. The Autoencoder Model Definition and Training

2.2.5. Autoencoder Model Definition:

`Sequential()` initializes a linear stack of layers.

`Dense()` layers are fully connected layers.

`input_dim=X_train.shape[1]` specifies the input dimension for the first layer based on the number of features in `X_train`.

`activation='relu'` uses Rectified Linear Unit activation function for hidden layers.

`activation='sigmoid'` uses Sigmoid activation function for the output layer to reconstruct input data between 0 and 1.

`compile(optimizer='adam', loss='mean_squared_error')` configures the model for training with Adam optimizer and Mean Squared Error loss function.

`EarlyStopping` is a callback that stops training when a monitored metric has stopped improving.

`monitor='val_loss'` monitors validation loss.

`patience=5` waits for 5 epochs after the validation loss has stopped improving.

`restore_best_weights=True` restores model weights from the epoch with the best value of the monitored quantity.

2.2.6. Training the Autoencoder:

`fit()` trains the autoencoder model on `X_train` with itself (`X_train`) as both input and output.

`epochs=20` specifies the number of training epochs.

`batch_size=32` determines the number of samples per gradient update.

`validation_data=(X_val, X_val)` uses `X_val` for validation during training.

`callbacks=[early_stopping]` applies early stopping during training to prevent overfitting.

2.2.7. Encoding Data:

`Sequential(autoencoder.layers[:2])` creates an encoder model using the first two layers of the trained autoencoder.

`encoder.predict(X_train)` and `encoder.predict(X_test)` encode `X_train` and `X_test` data into compressed representations (`X_train_encoded` and `X_test_encoded`) using the trained encoder model.

2.2.8. Multilayer Perceptron (MLP)

A Multilayer Perceptron is a type of neural network that is highly accurate at finding complicated patterns and connections in data. Its ability to learn from both structured and unstructured data makes it useful for finding simple and complicated attack patterns [3]. As shown in Figure 7, the MLP Classifier Implementation is explained as follows.

2.2.9. MLP Classifier Definition:

`MLPClassifier` is a multi-layer perceptron classifier from `sklearn.neural_network` module.

`hidden_layer_sizes=(64, 32)` defines two hidden layers with 64 and 32 neurons, respectively.

`max_iter=200` sets the maximum number of iterations for training.

`activation='relu'` uses Rectified Linear Unit activation function for hidden layers.

`solver='adam'` uses Adam optimizer for training.

`random_state=42` sets the random seed for reproducibility.

2.2.10. Training the MLP Classifier:

`mlp.fit(X_train_encoded, y_train)` trains the MLP classifier on the encoded training data (`X_train_encoded`) and corresponding labels (`y_train`).

2.2.11. Evaluating the MLP Classifier:

`mlp.predict(X_test_encoded)` predicts labels for the encoded test data (`X_test_encoded`).

`accuracy_score(y_test, mlp_predictions)` computes the accuracy of predicted labels (`mlp_predictions`) compared to true labels (`y_test`).

`classification_report(y_test, mlp_predictions)` generates a detailed classification report including precision, recall, F1-score, and support.

2.2.12. Decision Tree

We use this model to organize data into a tree-like structure to conduct AI-based decisions on the smart sensors using predetermined conditions. The decision tree classification model is intuitive and can handle numerical and categorical data, making it suitable for classifying different types of cloud intrusion and detection attacks based on the specific parameters and characteristics we mentioned. This research presents the decision tree for this cloud service as follows.

```
70
71 # MLP Classifier
72 mlp = MLPClassifier(hidden_layer_sizes=(64, 32), max_iter=200, activation='relu', solver='adam', random_state=42)
73 mlp.fit(X_train_encoded, y_train)
74
75 # Evaluate the MLP on the test set
76 mlp_predictions = mlp.predict(X_test_encoded)
77 print("MLP Test Accuracy:", accuracy_score(y_test, mlp_predictions))
78 print(classification_report(y_test, mlp_predictions))
79
80 # Decision Tree Classifier
81 dt = DecisionTreeClassifier(random_state=42)
82 dt.fit(X_train, y_train)
83
84 # Evaluate the Decision Tree on the test set
85 dt_predictions = dt.predict(X_test)
86 print("Decision Tree Test Accuracy:", accuracy_score(y_test, dt_predictions))
87 print(classification_report(y_test, dt_predictions))
88
```

Figure 7. MLP and Decision Tree Model Implementation

The Decision Tree Classifier Implementation in Figure 7 is explained below:

2.2.13. Decision Tree Classifier Definition:

DecisionTreeClassifier is a classifier from sklearn.tree module.
random_state=42 sets the random seed for reproducibility.

2.2.14. Training the Decision Tree Classifier:

dt.fit(X_train, y_train) trains the Decision Tree classifier on the training data (X_train) and corresponding labels (y_train).

2.2.15. Evaluating the Decision Tree Classifier:

dt.predict(X_test) predicts labels for the test data (X_test).
accuracy_score(y_test, dt_predictions) computes the accuracy of predicted labels (dt_predictions) compared to true labels (y_test).
classification_report(y_test, dt_predictions) generates a detailed classification report including precision, recall, F1-score, and support.

2.2.16. Intelligent Prioritization and Prevention System

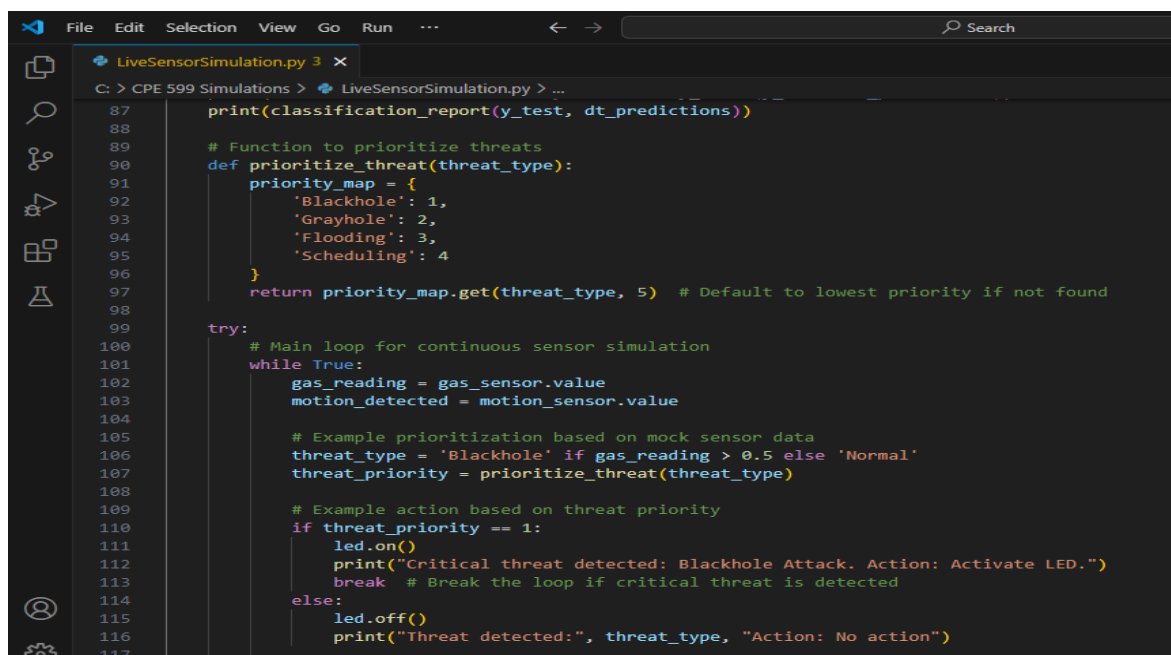
This paper implemented a smart prioritization and prevention system framework in a cloud environment, as shown in Figure 8. Based on their significance, this system categorizes various types of attacks (blackhole, grayhole, flooding, and scheduling). Blackhole attacks, for instance, are deemed critical in sectors like robotics control and energy management due to their potential to disrupt vital systems. Grayhole attacks, which selectively alter packets, are common in quality assurance and asset tracking, affecting data accuracy. Flooding attacks, prevalent in smart logistics and supply chain visibility, overload networks with excessive traffic. Scheduling attacks target manufacturing processes and healthcare equipment, impacting timing and scheduling systems. To proactively prevent such threats, attacks are prioritized based on their impact on Industry 4.0 scenarios. Prevention measures include setting up verification environments, utilizing validation tools, deploying traffic analysis and rate-limiting methods, and employing time synchronization procedures. These tailored prevention plans aim to enhance the safety and reliability of Wireless Sensor Networks (WSNs).

3. RESULTS AND DISCUSSION

This paper uses deep learning to implement a predictive framework for cloud-based intrusion detection and prevention in wireless sensor networks. The concept of the deep learning algorithm used has significantly improved the accuracy and efficiency of cloud intrusion detection and prevention in WSNs. This simulation will demonstrate three distinct machine learning models implemented based on the cloud-based intrusion and prevention system for wireless sensors. These are: Decision Tree, Multilayer Perceptron (MLP), and Autoencoder. The evaluation results are highlighted below:

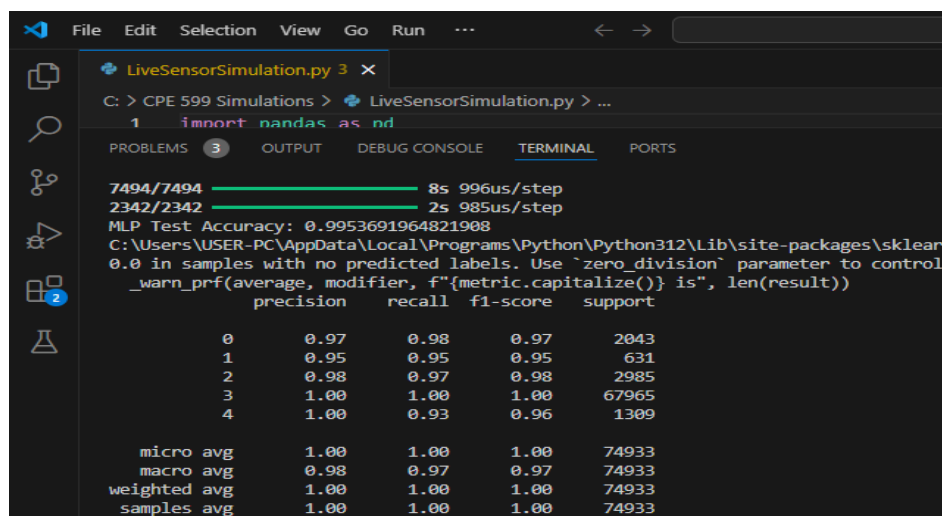
3.1. Multilayer Perceptron (MLP)

MLP presented the best classification approach for cloud-based WSN intrusions. As evidenced in Figure 9, this research achieves 99.37% simulation accuracy.



```
87 print(classification_report(y_test, dt_predictions))
88
89 # Function to prioritize threats
90 def prioritize_threat(threat_type):
91     priority_map = {
92         'Blackhole': 1,
93         'Grayhole': 2,
94         'Flooding': 3,
95         'Scheduling': 4
96     }
97     return priority_map.get(threat_type, 5) # Default to lowest priority if not found
98
99 try:
100     # Main loop for continuous sensor simulation
101     while True:
102         gas_reading = gas_sensor.value
103         motion_detected = motion_sensor.value
104
105         # Example prioritization based on mock sensor data
106         threat_type = 'Blackhole' if gas_reading > 0.5 else 'Normal'
107         threat_priority = prioritize_threat(threat_type)
108
109         # Example action based on threat priority
110         if threat_priority == 1:
111             led.on()
112             print("Critical threat detected: Blackhole Attack. Action: Activate LED.")
113             break # Break the loop if critical threat is detected
114         else:
115             led.off()
116             print("Threat detected:", threat_type, "Action: No action")
117
```

Figure 8. Intelligent Prioritization and Prevention Logic



```
7494/7494 ██████████ 8s 996us/step
2342/2342 ██████████ 2s 985us/step
MLP Test Accuracy: 0.9953691964821908
C:\Users\USER-PC\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn
0.0 in samples with no predicted labels. Use `zero_division` parameter to control
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
precision recall f1-score support
0 0.97 0.98 0.97 2043
1 0.95 0.95 0.95 631
2 0.98 0.97 0.98 2985
3 1.00 1.00 1.00 67965
4 1.00 0.93 0.96 1309

micro avg 1.00 1.00 1.00 74933
macro avg 0.98 0.97 0.97 74933
weighted avg 1.00 1.00 1.00 74933
samples avg 1.00 1.00 1.00 74933
```

Figure 9. Accuracy, precision, recall, and F1 score for the MLP model

Based on Figure 9, a simulation is conducted for accuracy, precision, recall, and F1 score for different attack types in the cloud computing environment. In the MLP model, Normal and TDMA classes were examined. This is to indicate high accuracy in positive predictions. However, the TDMA class has a slightly lower recall rate, suggesting that some real TDMA instances are missed. The Blackhole, Flooding, and Grayhole classes show a good balance between precision and recall, reflected in their high F1-Scores. The 'Macro Avg' and 'Weighted Avg' rows highlight the MLP model's consistently high performance across the dataset. This indicates the robust classification capabilities of the MLP model in this multiclass problem. In addition to the MLP model's precision, recall, and F1 scores per class, Figure 9 presents the Normal class with an exceptional precision of 99.7%, and the TDMA class also achieves 99.7%, demonstrating the model's effectiveness in reducing false positives. The Grayhole class has a precision of 94%, while the Blackhole and Flooding classes show strong precisions of 93% and 99.9%, respectively, indicating effective reduction of false positives.

3.2. Decision Tree (DT)

DT in this simulation presents effective accuracy during attack identification and prediction in cloud-based networks, which is crucial for cybersecurity (see Figure 10). The macro average and weighted average metrics demonstrate our model's overall high performance across the dataset, underscoring the robust capability of DT handling multiclass classification problems.

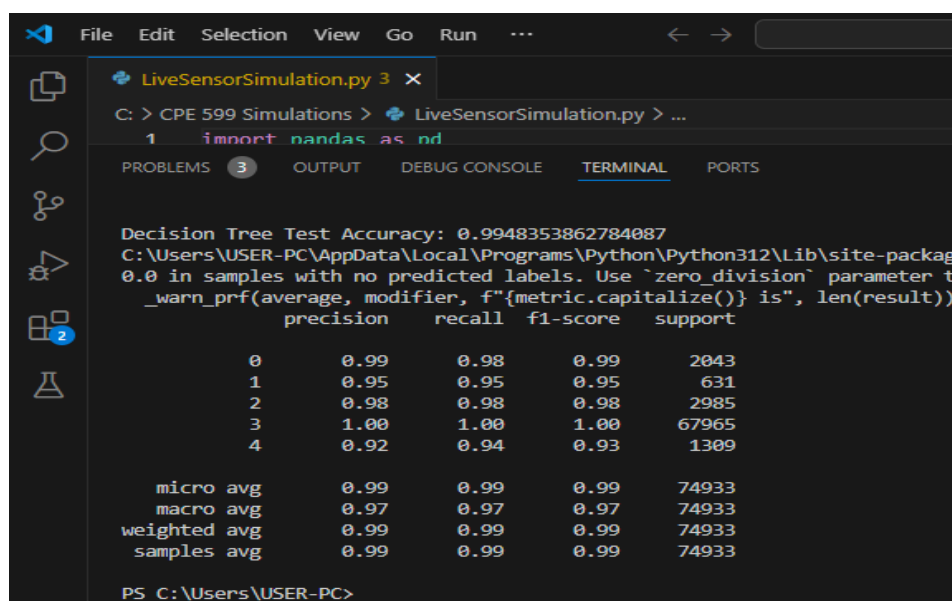


Figure 10. Impact of Decision Tree Classifier

Figure 10 shows a simulation based on accuracy, precision, recall, and F1 score for different attack types in the cloud environment. The results measure the precision score per class, where the Decision Tree is robustly classified in all instances. For precision, 99% is achieved for the "Normal" class. This indicates high accuracy in detecting typical occurrences and reducing false positives. For Blackhole and Grayhole classes, we achieved precisions of 95.82% and 98.89%. This shows its effectiveness in classifying all kinds of intrusions. The flooding class achieves a precision of 92%. This suggests a slightly higher chance of false positives. Overall, the Decision Tree model shows strong performance across all classes. In addition, the recall score per class is presented to indicate the model's ability to classify each class accurately. With high recall values for "Normal" (98%) and "Blackhole" (95%), the model

effectively captures most real occurrences in these classes. For the Grayhole class, we achieved a recall of 98%. That means the model minimizes false negatives by efficiently identifying instances of network intrusions.

3.3. Autoencoder

As shown in Table 2, Autoencoder results highlight the performance metrics of a binary classification model that distinguishes between normal and anomalous instances. This simulation's precision for normal and anomalous instances is high, at 0.95 and 0.88. This indicates the model predicts an instance as normal or anomalous. The recall for normal and anomalous instances is 0.88 and 0.95, showing the model captures 88% of actual normal instances and 95% of actual anomalous instances. The F1-score for the Autoencoder model is 0.91 for normal instances and 0.92 for anomalous instances. This indicates enhanced balance between precision and recall. Therefore, the Autoencoder model outperforms other models during instance identification. Overall accuracy of the Autoencoder model is 0.91. This means it can correctly identify 91% of the total instances. For sensitivity, the Autoencoder model is 88%, and thus performs well in identifying positive instances out of all positive ones. For specificity, the Autoencoder model, being 95%, performs well in identifying negative instances out of all actual negative instances. Considering sensitivity and specificity, the Autoencoder model effectively identifies positive and negative instances. These results are presented in Table 2.

Table 2. Training and Validation Results of Autoencoder

Class	Precision	Recall	F1-Score	Accuracy	Sensitivity	Specificity
Normal	99%	88%	91%			
Anomaly	88%	95%	92%			
Macro Average	92%	91%	91%	91%	88%	95%
Weighted Average	92%	91%	91%			

As shown in Table 2, the training and validation of the Autoencoder model are conducted across 20 epochs, along with the reconstruction error report. An enhancement is noted for accuracy from 84.90% in the first epoch to 87.81% in the final epoch, with corresponding decreases in loss values. The validation accuracy mirrors this trend, reaching 87.81% by the end of the training period (this is only one of many iterations). This indicates that the model learns effectively and generalizes well to unseen data. The mean reconstruction error is 0.250247, which is presented using a standard deviation of 0.238110, suggesting that most reconstruction errors are low, though there is some variability. The minimum error is 0.038404, and the maximum error is 26.071553. The 25th, 50th, and 75th percentiles are 0.094400, 0.128891, and 0.233322, respectively, indicating that most reconstruction errors are below 0.233322. Furthermore, the Autoencoder Reconstruction Error Report is presented in Figure 11.

The performance of the Multilayer Perceptron (MLP) model significantly enhances the effectiveness of the Decision Tree model, resulting in improved accuracy, precision, recall, and F1 scores. This improvement is particularly evident in detecting and classifying cloud-based cybersecurity intrusions, underscoring the models' potential for robust threat identification. The findings indicate optimizing the MLP model yields better precision in detecting cloud-based intrusions within wireless sensor networks. Additionally, the results for the Decision Tree model are analyzed using various averaging methods, including micro, macro, weighted, and sample averages, to comprehensively evaluate its performance. These insights highlight the

complementary strengths of the MLP and Decision Tree models in addressing cybersecurity challenges in cloud and wireless sensor network environments.

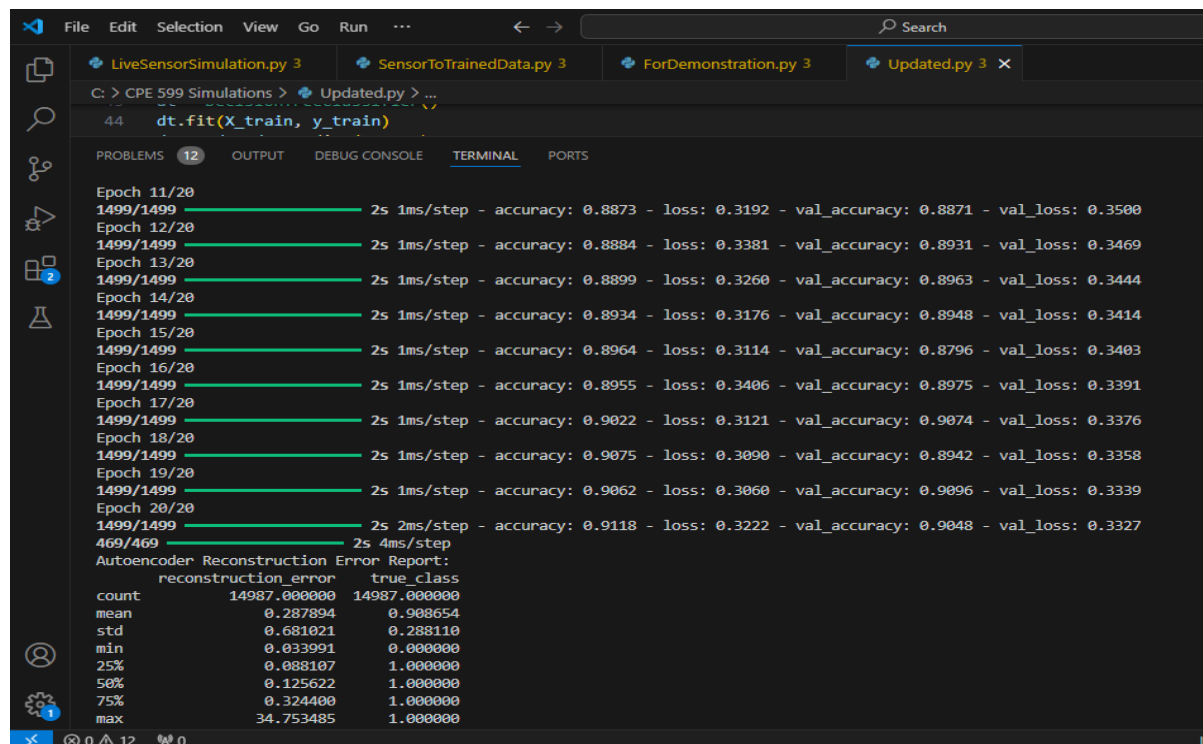


Figure 11. Autoencoder Reconstruction Error Report

3.4. Limitations

Observed limitations include increased computational delays on the Raspberry Pi during high-traffic scenarios. Addressing these limitations through hardware upgrades or algorithmic optimization will enhance the system's efficiency.

3.5. TESTING AND EVALUATION

Testing is conducted to evaluate the precision of the implemented predictive system. The evaluation considers a cloud computing scenario involving wireless sensor clusters in the cloud during intrusion detection and prevention actions. The technique involves, first, the use of an Internet cloud-based framework to secure and analyze real-time wireless sensor clusters. This includes detecting and classifying a wide range of cloud-based cybersecurity intrusions, such as data exfiltration, denial-of-service attacks, and unauthorized access attempts. Second, we perform intelligent prediction and prioritization to assess the framework's accuracy. This ensures that the model can accurately identify potential threats in the Internet cloud under specific algorithmic conditions. Finally, proactive measures are implemented to prevent attacks on the system's capacity, mitigating the impact of cloud-based breaches and enhancing real-time defense mechanisms through adaptive security approaches.

4. CONCLUSION AND FUTURE WORK

This paper outlined a comprehensive research study investigating cloud computing-based security analysis on wireless sensor node clusters using predictive techniques. It actively implements an intelligent and robust predictive framework for cloud-based intrusion detection

and prevention for Industry 4.0 using wireless sensor networks as follows. Firstly, an AI-based detection mechanism is implemented to recognize and classify wireless sensors using a cloud computing strategy and cybersecurity intrusions. Three distinct machine learning models (multilayer perceptron, autoencoder, and decision tree) are implemented for cybersecurity intrusion detection and classification within WSNs. Secondly, an intelligent prioritization model is implemented to give priorities to cyber threats based on their nature and impact. Finally, a prevention system is implemented to mitigate the impact of cybersecurity intrusions on the cloud effectively. The result presents significant advancements in WSN security, with implications for various applications. Leveraging advanced algorithms such as machine learning algorithms, intelligent prioritization mechanisms, and proactive safety measures enhances the framework's security and resilience against evolving cyber threats. Future research will explore advanced ensemble techniques to improve model performance, adaptive learning algorithms to enhance resilience against evolving threats, and real-time deployment scenarios to validate the framework's effectiveness in industrial environments.

ACKNOWLEDGEMENT

This work is supported by the Ministry of Higher Education (MOHE) Fundamental Research Grant Scheme (FRGS22-264-0873) (Grant No: FRGS/1/2022/ICT11/UIAM/01/1).

REFERENCES

- [1] Dash, Sanjit Kumar, Subasish Mohapatra, and Prasant Kumar Pattnaik. "A survey on applications of wireless sensor network using cloud computing." *International Journal of Computer Science & Emerging Technologies* 1, no. 4 (2010): 50-55.
- [2] John, Ayuba, Ismail Fauzi Bin Isnin, Syed Hamid Hussain Madni, and Muhammed Faheem. "Cluster-based wireless sensor network framework for denial-of-service attack detection based on variable selection ensemble machine learning algorithms." *Intelligent Systems with Applications* 22 (2024): 200381.
- [3] Sareen, Sanjay, Sandeep K. Sood, and Sunil Kumar Gupta. "An automatic prediction of epileptic seizures using cloud computing and wireless sensor networks." *Journal of medical systems* 40 (2016): 1-18.
- [4] Mohammed, Ahmed. "The Web Technology and Cloud Computing Security based Machine Learning Algorithms for Detect DDOS Attacks." *Journal of Information Technology and Informatics* 3, no. 1 (2024).
- [5] Gayathri, S., and D. Surendran. "Unified ensemble federated learning with cloud computing for online anomaly detection in energy-efficient wireless sensor networks." *Journal of Cloud Computing* 13, no. 1 (2024): 49.
- [6] Luo, Haonan, Jing Wang, Deyu Lin, Linghe Kong, Yufei Zhao, and Yong Liang Guan. "A Novel Energy-Efficient Approach Based on Clustering Using Grey Prediction in WSNs for IoT Infrastructures." *IEEE Internet of Things Journal* (2024).
- [7] Yadav, Santosh Kumar, and Rakesh Kumar. "Scalable energy optimization of resources for mobile cloud computing using sensor enabled cluster-based system." *Wireless Networks* (2024): 1-26.
- [8] Chellathurai Amirthabai, Subasini, Udit Malhotra, Saravanan Thapasimuthu Rajeswari, and Sudhahar Thachankurichy Natesan. "Healthcare security in cloud-based wireless sensor networks: Botnet attack detection via autoencoder-aided goal-based artificial intelligent agent." *Concurrency and Computation: Practice and Experience*: e8152.

-
- [9] Alawi, Mahmoud A., Rashid A. Saeed, and Aisha A. Hassan. "Cluster-based multi-hop vehicular communication with multi-metric optimization." In *2012 international conference on computer and communication engineering (ICCCE)*, pp. 22-27. IEEE, 2012.
- [10] Islam, Shayla, Aisha-Hassan Abdalla Hashim, Mohamed Hadi Habaebi, and Mohammad Kamrul Hasan. "Design and implementation of a multihoming-based scheme to support mobility management in NEMO." *Wireless Personal Communications* 95 (2017): 457-473.
- [11] Hassan, Mona Bakri, Rashid A. Saeed, Othman Khalifa, Elmustafa Sayed Ali, Rania A. Mokhtar, and Aisha A. Hashim. "Green machine learning for green cloud energy efficiency." In *2022 IEEE 2nd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, pp. 288-294. IEEE, 2022.
- [12] Badawi, Ahmed SA, Nurul Fadzlin Hasbullaha, Y. Yusoff, Sheroz Khan, Aisha Hashim, Alhareth Zyoud, and Mohammed Elamassie. "Evaluation of wind power for electrical energy generation in the mediterranean coast of Palestine for 14 years." *International Journal of Electrical and Computer Engineering* (2019).
- [13] Elagib, Sara B., Atahur Rahman Najeeb, Aisha H. Hashim, and Rashidah F. Olanrewaju. "Big data analysis solutions using MapReduce framework." In *2014 International Conference on Computer and Communication Engineering*, pp. 127-130. IEEE, 2014.
- [14] Khalifa, Othman O., Adil Roubleh, Abdelrahim Esgiar, Maha Abdelhaq, Raed Alsaqour, Aisha Abdalla, Elmustafa Sayed Ali, and Rashid Saeed. "An IoT-platform-based deep learning system for human behavior recognition in smart city monitoring using the Berkeley MHAD datasets." *Systems* 10, no. 5 (2022): 177.
- [15] Hashim, M. M., Mustafa Sabah Taha, Azana Hafizah Mohd Aman, Aisha Hassan Abdalla Hashim, Mohd Shafry Mohd Rahim, and Shayla Islam. "Securing medical data transmission systems based on integrating algorithm of encryption and steganography." In *2019 7th International Conference on Mechatronics Engineering (ICOM)*, pp. 1-6. IEEE, 2019.