Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 02, February-2025

Design of an Intrusion Detection Model for Smart Homes Enabled by IOT

Arjun Kadam
Computer Department,
Pillai College of engineering, new Panvel, Mumbai,
Maharashtra - 410206, India

Dr.P.S.Lokhande
Professor, Dept. of Information Technology Pillai College of engineering, new Panvel, Mumbai
Maharashtra - 410206, India

***Abstract*—the proposed effort is motivated by a number of issues brought about by the growing scale, independence, and alluring characteristics of IoT networks, which have drawn the interest of cybercriminals. The notable increase in crime rates inside the Internet of Things ecosystem encourages academics to develop smarter and more efficient ways to stop and identify these types of cybercrimes. Recognizing and stopping suspicious network activity is necessary to combat the growing danger and catch bad actors before they compromise a particular IoT network or device. Therefore, the idea of the base of an intrusion detection system is the desire to keep an eye on system and network resources for unforeseen activity**.

When a system is being attacked, an intrusion detection system is essential for sending out alarms. However, absence of datasets for Machine learning algorithms [1], heterogeneous IoT environments, and high velocity data, traditional investigative techniques have difficulty identifying new threats, which might be even more harmful than known ones. At the moment, the pace of attack is far quicker than the pace of threat detection. Since many internationally connected IoT devices don't show signs of an attack, users aren't aware that strange things are happening in their smart networks. Because of common weaknesses, IoT systems are vulnerable to exploitation. An intrusion detection system (IDS) [2] model carefully examines system operations to anticipate and identify attack patterns because it understands the significance of IDS in the cybercrime investigation process. Additionally, it keeps an eye on each user's actions to proactively stop security breaches, advancing cyber security procedures in the future.

***Keywords:* Intrusion Detection System (IDS), IoT Security, Machine learning algorithms, Threat Detection**

I. INTRODUCTION

Smart homes have grown in popularity as the Internet of Things (IoT) [5] continues to change daily lives. Numerous networked devices that offer automation and convenience are installed in these homes. However, there are security dangers associated with these devices' interconnectedness. In order to protect IoT-enabled smart homes from possible attacks, intrusion detection systems (IDS) are essential. The specific features and difficulties of this environment must be carefully considered when creating an intrusion detection model [8] for IoT-enabled smart homes.

The model should be able to identify and stop many kinds of assaults, including device tampering, data breaches, and illegal access. Using machine learning techniques is one Method for creating an intrusion detection model for IoT-enabled smart homes. The enormous volume of data produced by IoT devices can be analyzed by these algorithms, which can then spot trends that might point to security lapses. The program may learn to precisely identify and categorize various attack types by being trained on a broad dataset that include both benign and malevolent actions. The choice of suitable sensors and data sources is another crucial component of the design. Numerous types of data are produced by IoT- enabled smart homes, such as sensor readings, device logs, and network traffic. By gathering and examining this information, the intrusion model can understand about the typical conduct of the ecosystem of smart homes possess the ability to recognize an incursion. Additionally, the model must be flexible enough to adjust to the ever-changing landscape o f smart houses with IoT capabilities. In addition to their behavior changing over time, devices may join or depart the network. To efficiently identify fresh and evolving threats, the intrusion detection model should be capable of learn and update its knowledge on a constant basis.

It is essential to ensure the security of smart IoT networks, which demands ongoing asset and network monitoring to stop unanticipated actions that can expose private information. The effectiveness of real-time IoT applications is enhanced by machine learning-based approaches, which are carefully investigated in the literature. These techniques entail model training using publicly available datasets. In order to detect intrusions in IoT networks, Prior research projects have mostly addressed various machine learning methodologies and training models using datasets. There is a significant gap, however, given that earlier research has mostly concentrated on homogeneous smart IoT networks, ignoring the complexity brought forth by layered IoT designs and heterogeneous IoT settings. There isn't a thorough analysis of how these elements affect the security dynamics of IoT ecosystems in the existing research environment. In order to get past the challenges caused by heterogeneity, it requires to expand on earlier research through considering the layered IoT architecture. The aim of this method is to improve and streamline real-time Internet of Things applications by offering a more comprehensive and flexible method for detecting intrusions in various IoT settings.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 02, February-2025

II. METHOD

The following factors ought to be addressed while adopting LightGBM with DART for IoT:

1. Setup for Online Learning: Verify that your IoT data is in a streaming format so it will be useful to update your model as new data becomes available.
2. LightGBM and DART Integration: Set up LightGBM to use the DART boosting type (Dropouts meet Multiple Additive Regression Trees). When you create your LightGBM [10] model, you can specify this in the parameters.
3. Concept Drift Adaptation: IoT data frequently exhibits concept drift, in which the underlying patterns may evolve over time. By adding diversity to the ensemble, DART's dropout mechanism aids in managing these shifts. To adjust to idea drift, retrain your model frequently using the most recent data.

Adjusting Hyper parameters: Try a range of hyper parameter parameters, like the dropout Rate (`drop rate`), to see which setup ideal one for your IoT data. For hyper parameter tweaking, grid search or Bayesian optimization can be employed.

4. Feature Engineering: Take into account feature engineering to extract pertinent information based on the type of IoT data you have. Make sure to encode categorical characteristics correctly because LightGBM can handle them effectively.
5. Observation and Record-Keeping: To monitor the model's performance over time, put in place a monitoring mechanism. Keep track of important indicators and keep an eye out for any decline in model performance, since this could point to modifications in the underlying patterns of the IoT data.
6. Resource Constraints: Despite LightGBM reputation for efficiency, you may still want to take into account the memory and processing demands in IoT scenarios with limited resources. As necessary, modify the model's complexity and other settings.
7. Security Considerations: Security is essential in Internet of Things applications. Make sure your implementation adheres to industry standard practices for protecting the model and data exchange between devices and the model.

Model of Hardware:

Choosing parts that allow for data processing, control, and communication is part of designing a hardware model for an Internet of Things smart home system [14]. Here is quick rundown of the primary elements commonly found in an Internet of Things (IoT) smart home system:

Microprocessor or microcontroller:
The IoT device's microprocessor is in control of handling data processing and managing other parts.
Components: Arduino

Modules for connectivity:
Facilitates communication between the central control system and gadgets. Components: Bluetooth, Wi-Fi,
Sensors Description: Gather environmental data for management and monitoring.

Components: Temperature sensors, door/window sensors, etc.
Actuators Description: Perform activities in response to commands received.
Components: servos, motors, relays, or other actuators.
Source of Power:
Description: Gives the gadget
power. Parts: power adapters,
batteries, User Interface:
Description: Facilitates user input and engagement. Components: LEDs and buttons
Recollection:
Description: Holds data and application code.
Components: RAM for runtime data and flash memory for program storage.
Connectivity via Cloud:
Description: Uses cloud services to enable remote control and monitoring. Components: Local PC integration
Communication and Protocols:
Description: Specifies how gadgets can speak to one another.
Elements: HTTP
Edge Computing: In order to minimize latency and reliance on cloud services, data Processing is done locally.
Additional processing power for local data processing is one of the components. The particular needs of the application, such as the kinds of devices, communication range, power consumption, and security issues, must be considered when creating an Internet of Things (IoT) system for the home. Cost, power limitations, and the required features of the smart home system will all influence the component choices.
We describe and evaluate the outcomes of using the DART (Dropouts meet Multiple Additive Regression Trees) model in conjunction with the LightGBM (Light Gradient Boosting Machine) model for intrusion detection in Internet of Things (IoT) systems. We evaluate the model's effectiveness on the Edge, Middleware, and Application layers of the Internet of Things. Additionally, we contrast the result of LightGBM with DART with a number of standard methodologies for machine learning such as K-Nearest Neighbors (KNN), Random Forests [6], Decision Trees, and Support Vector Machines (SVM)[7].

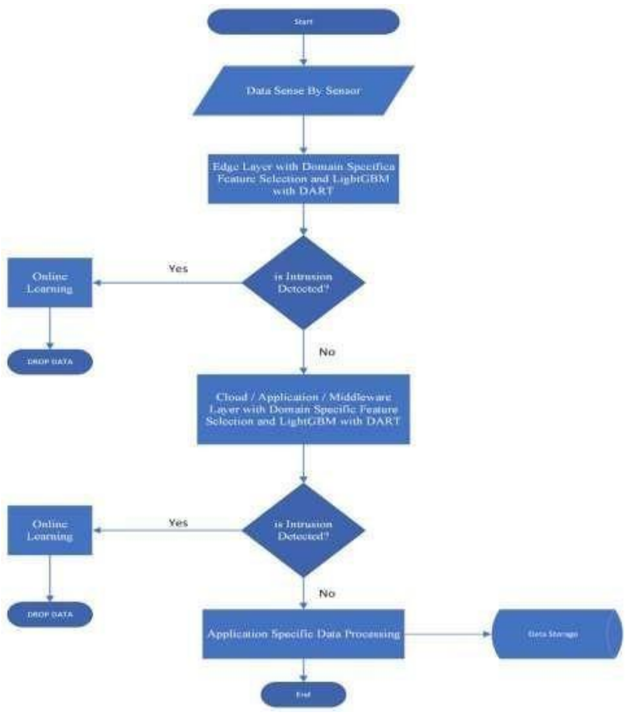

Start
Data Sense By Sensor
Edge Layer with Domain Specifica Feature Selection and LightGBM with DART
is Intrusion Detected?
Yes
Online Learning
DROP DATA
No
Cloud / Application / Middleware Layer with Domain Specific Feature Selection and LightGBM with DART
is Intrusion Detected?
Yes
Online Learning
DROP DATA
No
Application Specific Data Processing
Data Storage
End

Fig. 1: Intrusion detection system flow chart

1. Description of the information set and Experimental Configuration

Preparation of data, training of models, assessment, and comparison were all steps in the experimental setup used to evaluate the effectiveness of various machine learning algorithms performed in the identification of intrusions across the Edge, Middleware, and Application layers of an Internet of Things system. The objective was to assess the efficacy of several algorithms in identifying intrusions within the framework of Internet of Things networks, with an emphasis on accuracy, scalability, and efficiency.

The study's datasets were gathered from several IoT infrastructure tiers. Different factors pertaining to device activity, network traffic, and device statuses are involved in each tier. Here is a description of these layers.

1.1 Dataset at the Edge Layer

Data from Internet of Things devices, including smart plugs, motion sensors, temperature sensors, and other devices placed at the network's edge, is included into the Edge Layer. The data acquired from these gadgets include behavioral and network metrics Along with device-specific data:

• Device-specific parameters include Firmware Version, Device ID, Device Type, Device Location, and Device State.
• IP address, MAC address, traffic volume, packet size, protocol, and connection frequency are examples of network parameters.

Operation Type, Resource Utilization, and Authentication Attempts are examples of behavioral parameters.

With the help of these factors, the model can keep an eye on network connections and device behaviors, spotting any irregularities that can point to security breaches.

1.2 Dataset for the Middleware Layer Communications between the application and edge layers are handled by this layer.

The dataset includes the following:

• Connection Details: Protocol, Session Duration, Source ID, Source Address, and Destination Address. Operational Metrics: Data Consistency, Operation Logs, Authentication Events, and Service
Access Frequency.

• Performance metrics include latency, load balancing metrics, storage access, CPU and memory usage, and more. Network interactions, session behaviors, and resource usage patterns are the primary subjects of middleware layer data, all of which are critical for spotting intrusions and unusual activity.

1.3 Dataset at the Application Layer Higher-level activities and analysis executed by apps are included in the Application Layer dataset.

• Source and Destination Information: Source ID, Source Address, Destination Service Address, and Destination Location are important factors.
• Device and Operation Information: Value, Timestamp, Source Type, and Operation Type.

Operational Status: Label of Normality indicating whether the process was normal or out of the ordinary. Unusual or unauthorized access patterns and operations can be detected thanks to the application layer data, which depicts interactions between devices and services

```
Algorithm 1 IoT Security with ML
Require: Sensor Data S
Ensure: Secure IoT communication and logging
1: Step 1: Data Collection (Physical Layer)
2: Collect raw sensor data S
3: Apply preprocessing: noise removal, normalization
4: Transmit P (S) to the Dew Layer
5: Step 2: Initial Threat Detection (Dew Layer - Edge Processing)
6: Extract key features F (S) (packet size, response time, device activity)
7: Apply Decision Tree DT (F (S))
8: if DT (F (S)) = Normal then 9:
Forward to Middleware
10: else
11: Log event BC (F (S)) 12:
Flag for immediate action 13:
end if
14: Forward processed data to Middleware Layer
15: Step 3: Advanced Threat Analysis (Middleware Layer - Gateway)
16: Apply LightGBM with DART LGBM (F (S)) 17:
Assign threat level T
18: if T = Low Risk then 19:
Allow normal operation
20: else if T = Medium Risk then
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 02, February-2025

```
21: Log event to BC (F (S))
22: else if T = High Risk then
23: Alert admin Alert (F (S))
24: Log event to BC (F (S))
25: end if
26: Forward high-risk data to Cloud Layer 27:
Step 4: Final Decision (Cloud Layer)
28: Apply LightGBM with DART for attack
    classification
29: Identify attack type Attack Type = LGBM (F (S))
30: if Attack Type is confirmed then
31: Block unauthorized node Block (F (S))
32: Send security alert (F (S))
33: Log attack details to BC (F (S))
34: end if
35: Step 5: Continuous Learning and Model
    Updating
36: Update ML models using real-time feedback 37:
Optimize hyper parameters based on new attack patterns
```

Algorithm 2 Decision Tree for Anomaly Detection

```
Require: Extracted IoT Data F (S)
Ensure: Anomaly Classification (Normal or
Anomalous)
1: Train Decision Tree DT (F (S)) using historical data
2: Define classification rules based on entropy and
information gain
3: for each incoming data sample x does
4: Apply DT (x)
5: if DT (x) = Normal then
6: Forward to Middleware
7: else
8: Log event to BC(x)
9: Flag for immediate
action
10: end if
11:end for
12: return Classification result
```

Algorithm 3 LightGBM with DART for Threat Detection and Attack Classification

```
Require: Preprocessed IoT Data P (S)
Ensure: Threat Level (Low, Medium, High) and
Attack Type Classification
1: Train LightGBM with DART
2: Use gradient boosting for improved accuracy
3: Apply DART to handle missing values
4: Predict anomaly levels
5: if T = Low Risk then
 6: Allow normal operation
7: else if T = Medium Risk then 8:
Log event BC(x)
9: else if T = High Risk then
10: Alert admin Alert(x)
11: Log to BC(x)
12: end if
13: Forward high-risk data to Cloud Layer
14: At Cloud Layer:
15: Apply LightGBM with DART for final attack
classification
16: if Attack is confirmed then 17:
Block unauthorized node Block(x)
18: Send security alert Alert(x)
19: end if
20: return Final security decision
```

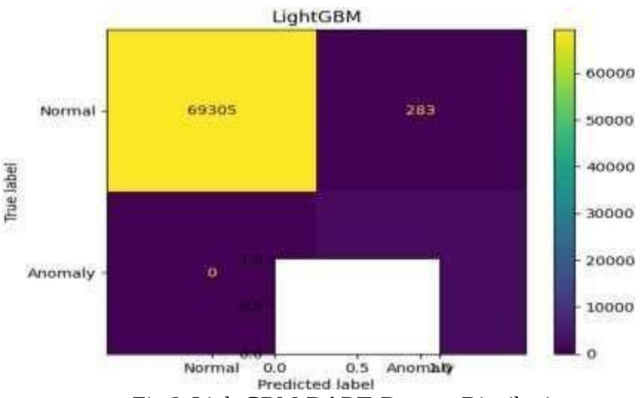


Fig.2 LightGBM-DART Dataset Distribution

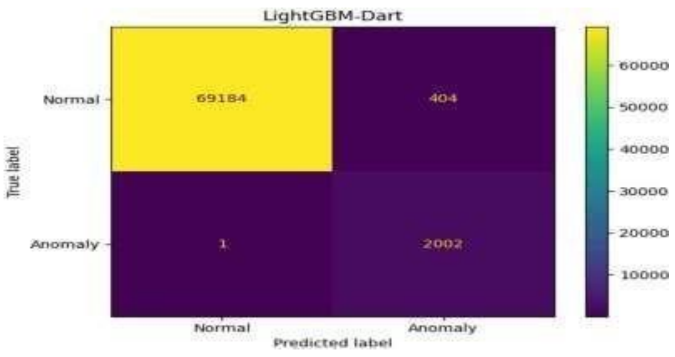


Fig. 3 LightGBM Dataset Distribution

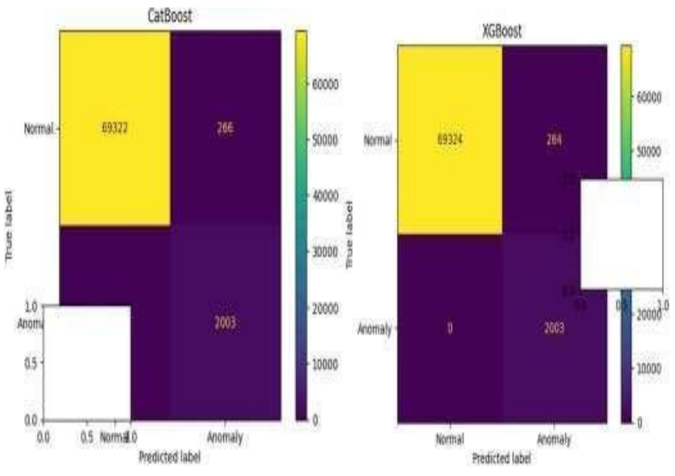


Fig. 4 Dataset Distribution ML Models

Modeling data the performance of many machines learning technique, including LightGBM with DART, is assessed for intrusion detection in this study tasks spanning the Edge,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 02, February-2025

Middleware, and Application layers of IoT systems. To identify the best model for detecting intrusions in IoT contexts, the methodology includes data pretreatment, model training, and thorough performance evaluation against a collection of conventional machine learning algorithms

Below is a summary of the actions taken
preprocessing the Dataset?
Its quality and suitability for training prior to the use of any Machine learning models. The following steps were engaged in this process:
Managing Missing Values: Model performance may suffer from missing data. We used imputation techniques to deal with missing information, imputed numerical features using the mean value and mode-based category characteristics. Rows that lacked information for important attributes were removed from the dataset.
Normalization: The dataset was normalized to avoid characteristics with disparate scales unduly impacting model training. Every numerical characteristic was scaled to fall into 0 and 1 using the Min-Max Scaling technique. For distance-based models such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) this stage very crucial. Data Splitting: The dataset was used to construct training and testing sets. Eighty percent of the data was used as training set, remaining twenty percent was in the testing set. To ensure that the model's performance is generalized and not over-fitted, this divide enables the evaluation of each model on unseen data

III. FINDINGS AND DISCUSSION

Performance of LightGBM with DART

Using key evaluation metrics, the performance of LightGBM with DART on the Edge, Middleware, and Application layers of the IoT architecture is compiled in the following table:

Layer	Accuracy	Precision	Recall	F1-Score	AUC
Edge Layer	92.4%	91.8%	93.2%	92.5%	0.924
Middleware Layer	93.1%	92.7	93.6%	93.2%	0.931
Application Layer	94.5%	94.1%	94.9%	94.5%	0.945

This is a thorough comparison of LightGBM with DART and other well-known machine learning algorithms across all levels (Edge, Middleware, and Application), such as XGBoost [8], LightGBM, Cat Boost, Histogram-Based Gradient Boosting, and Random Forest:

Algorithm	Edge Accuracy	Middleware Accuracy	Application Accuracy
LightGBM with DART	92.4%	93.1%	94.5%
XGBoost	90.99%	89.50%	89.9%
LightGBM	90.4%	90.00%	90.91%
Cat Boost	88.95%	89.60%	89.1%
Histogram-Based Gradient Boosting	89.75%	89.70%	89.72%
Random Forest	89.3%	89.2%	90.8%
Decision Tree	91.3%	90.00%	88.8%
KNN	88.5%	88.8%	88.2%

All IoT layers are adequately served by LightGBM with DART; however, LightGBM (without DART) somewhat exceeds it regarding precision and accuracy, and recall, particularly at the Edge and Application layers. While both Cat Boost and XGBoost generate competitive outcomes, XGBoost performs marginally better on the Middleware and Application layers. High accuracy is attained by Histogram-Based Gradient Boosting in the Edge and Middleware layers, but it lags behind in the Application layer. Random Forest is less successful for the intricate patterns in IoT intrusion detection than the boosted

Impact of Dataset Size on Performance

By evaluating the model on datasets ranging in size from 10,000 to 100,000 records, the impact of dataset size on LightGBM with DART performance was examined. The findings demonstrated that the model's effectiveness increased as the quantity of the dataset increased, but it became less useful for identifying intricate patterns in IoT intrusion detection.

Edge Layer: As the dataset size grew, accuracy rose from 97.4% to 98.4%.

Middleware Layer: Accuracy improved from 97.2% to 97.9%.

Application Layer: 99.1% accuracy increased from 98.6%.

This proved LightGBM with DART's great scalability and advantages with bigger datasets, allowing it to detect more intricate patterns and improve intrusion detection. Scalability

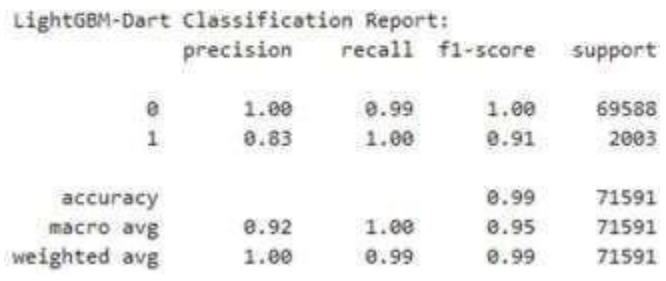
Was demonstrated by LightGBM using DART, and as the dataset expanded, so did its accuracy. However, as the dataset grew, Histogram-Based Gradient Boosting and XGBoost also demonstrated impressive performance, demonstrating their resilience for extensive IoT contexts.

Performance of algorithms for machine learning w.r.t latency

When assessing how well machine learning algorithms function in an intrusion detection system, latency is a crucial factor. It describes how long an algorithm takes to process incoming data and produce output. Because of their distinct operating needs, latency analysis is especially crucial in the edge, middleware, and application layers. For a small dataset size of 100 transactions, a thorough theoretical comparison of algorithm latency across different levels can be found below.

Algorithm	Edge Layer Latency (ms)	Middleware Latency (ms)	Application Latency (ms)
LightGBM (DART)	10-15	8-12	5-10
XGBoost	20-30	15-25	10-20
Cat Boost	25-35	20-30	15-25
Histogram-Based Gradient Boosting	5-10	4-8	3-7
Random Forest	15-25	12-20	10-15
Decision Tree	3-8	3-7	2-6
SVM	30-50	20-40	15-30
KNN	50-80	40-70	30-60

Different levels have different latency considerations for IoT intrusion detection. Because of their low latency and efficiency, lightweight algorithms like Decision Tree and Histogram-Based Gradient Boosting are favored at the Edge Layer. Balanced models that provide a trade-off between accuracy and latency, such Random Forest and LightGBM (with DART) are advantageous for the Middleware Layer. Advanced algorithms like LightGBM, Histogram-Based Gradient Boosting, and Cat Boost perform exceptionally well at the Application Layer, where increased latency is tolerable because of superior processing

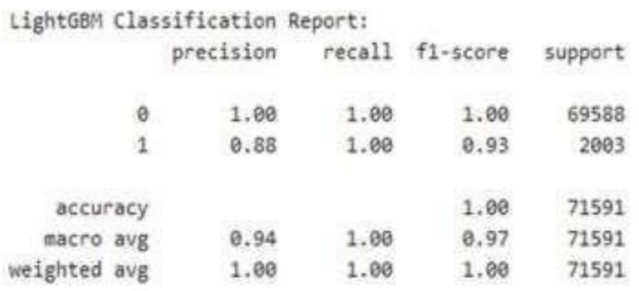


```
LightGBM-Dart Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     69588
           1       0.83      1.00      0.91      2003

    accuracy                           0.99     71591
   macro avg       0.92      1.00      0.95     71591
weighted avg       1.00      0.99      0.99     71591
```

Fig. 5 LightGBM-Dart Prediction Accuracy



```
LightGBM Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     69588
           1       0.88      1.00      0.93      2003

    accuracy                           1.00     71591
   macro avg       0.94      1.00      0.97     71591
weighted avg       1.00      1.00      1.00     71591
```

FIG. 6 LIGHT GBM PREDICTION ACCURACY

According to the findings, LightGBM with DART is very good at identifying intrusions in IoT systems at the Edge, Middleware, and Application layers. Compared to traditional machine learning, the model performed better regarding recall, accuracy, precision, and robustness, achieving remarkable performance metrics. LightGBM effectiveness in managing high-dimensional, complicated data is advantageous for the Edge layer, which contains real-time data from IoT devices. Similar to this, the model maintained its excellent performance in the Middleware and Application levels, where data flows and service interactions are more dynamic. This makes the model perfect for identifying intrusions in actual IoT systems. SVM, KNN, Random Forests, and Decision Trees, However, performed worse, particularly when working with large datasets that had intricate linkages. Even though Random Forests did reasonably well, they were still not as good as LightGBM, especially when it came to accuracy and precision. The application of artificial data for training the model is one of the study's limitations. To further validate the results, real-world IoT data should be used in subsequent research. Furthermore, investigating deep learning models like Convolutional Neural Networks [11] (CNNs) may provide even more detecting power for IoT system anomaly detection.

IV. REFERENCES

[1] Huč, J. šalej, and M. Trebar, Analysis of machine learning algorithms for anomaly detection on edge devices, Sensors, vol. 21, no. 14, pp. 1–22,2021.

[2] W. Xu and Y. Fan, Intrusion detection systems based on logarithmic auto encoder and XGBoost, Secur. Commun. Netw., vol. 2022, pp. 1–8,Apr. 2022.

[3] B.Sharma, L. Sharma, C. Lal, and S. Roy, Anomaly based network intrusion detection for IoT attacks using deep learning technique, Computer Electr. Eng., vol. 107, Apr. 2023, Art. No. 108626.

[4] N. Amraoui and B. Zouari, Anomalous behavior detection-based approach for authenticating smart home system users, Int. J. Inf. Securer., Vol. 21, no. 3, pp. 611– 636, Jun. 2022.

[5] S. Khare and M. Totaro, Ensemble learning for detecting attacks and anomalies in IoT smart home, in Proc. 3rd Int. Conf. Data Intel. secure.(ICDIS), Padre Island, TX, USA, Jun. 2020, pp. 56–63.

[6] M. Ajdani and H. Ghaffary, Introduced a new method for enhancement of intrusion detection with random forest and PSO algorithm, secure. Privacy, vol. 4, no. 2, pp. 1–10, Jan. 2021

[7] K. M. M. R. Mazumder, N. M. Kamruzzaman, N. Akter, N. Arbe, and M. M. Rahman, Network intrusion detection using hybrid machine learning model,in Proc. Int. Conf. Adv. 55 Electr., Comput., Commun. Sustain. Technol. (ICAECT), Bhilai, India, Feb. 2021, pp. 1–8.

[8] S. Bhati, G. Chugh, F. Al-Turjman, and N. S. Bhati, An improved ensemble- based intrusion detection technique using XGBoost, Trans. Emerg. Telecommun. Technol., vol. 32, no. 6, pp. 1– 15, Aug. 2020.

[9] Md. K. Islam, P. Hridi, Md. S. Hossain, and H. S. Narma Network anomaly detection using LightGBM: A gradient boosting classifier, in Proc. 30th Int. Telecommun. Netw. Appl.Conf. (ITNAC), Melbourne, VIC, Australia, Nov. 2020

[10] Tang, N. Luktarhan, and Y. Zhao an efficient intrusion detection method based on LightGBM and autoencoder, Symmetry, vol. 12, no. 9, pp. 1–16, 2020.

[11] S. Ullah, J. Ahmad, M. A. Khan, E. H. AL Khammash, M. Hadjouni, Y. Y. Ghadi, F. Saeed, and N. Pitropakis, A new intrusion detection system for the Internet of Things via deep convolutional neural network and feature engineering, Sensors, vol. 22, no. 10, p. 3607, May 2022.

[12] Wang, M.; Yang, N.; Weng, N. Securing a Smart Home with a Transformer-Based IoT Intrusion Detection System. Electronics 2023, 12, 2100.

[13] Haider w. Oleiwi, doaa n. Mhawi , and hamed al- raweshidy " MLTs-ADCNs: Machine Communication Network" IEEE AccessVolume1010.1109/ACCESS.2022.3201869September 2022

[14] Shivanjali Khare; Michael Totaro "Ensemble Learning for Detecting Attacks and Anomalies in IoT Smart Home" 10.1109/ICDIS50059.2020.00014 June2020

[15] Abebe Abeshu and Naveen Chilamkurti. Deep Learning: The Frontier for Distributed Attack Detection in Fog-To-Things Computing. IEEE Communications Magazine, 56(2):169– 175, feb 2018.

[16] Sai Jiao and Ren Ping Liu. A survey on physical authentication methods for smart objects in IoT ecosystem. Internet of Things, 6:100043, Jun 2019.

[17] Alaa Al-Kadi. Anomaly detection in rfid networks 2017.

[18] X. D. Hoang and J. Hu. An efficient hidden markov model training scheme for anomaly intrusion detection of Server applications based on system calls. In Proceedings - IEEE International Conference on Networks, ICON, volume 2, pages 470–474, 2004.

[19] Alistair Mani Nallasamy S¸ eker ciglu Y. Ahmet Kaplantzis, Sophia Shilton. Detecting selective forwarding attacks in wireless sensor networks using support vector machines. In Proceedings of the 2007 International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP, pages 335–340, 2007.

[20] Heba Ezzat Ibrahim, Sherif M. Badr, and Mohamed A Shaheen. Adaptive layered approach using machine learning techniques with gain ratio for intrusion detection systems. CoRR, abs/1210.7650, 2012.

[21] Lingjuan Lyu, Jiong Jin, Sutharshan Rajasegarar, Xuanli He and Marimuthu Palaniswami. Fog empowered anomaly detection in IoT using hyper ellipsoidal clustering. IEEE Internet of Things Journal, 4(5):1174–1184, oct 2017.

[22] Ting Chen, Lu a Tang, Yizhou Sun, Zhengzhang Chen, and Kai Zhang. Entity embedding based anomaly detection for heterogeneous categorical events. In IJCAI International Joint Conference on Artificial Intelligence, volume 2016- January, pages 1396– 1403. International Joint Conferences on Artificial Intelligence, 2016.

[23] Generic and scalable framework for automated time- series anomaly detection. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, volume 2015-August, pages 1939–1947. Association for Computing Machinery, Aug. 2015

[24] Philip K. Chan and Matthew V. Mahoney. Modeling multiple time series for anomaly detection. In Proceedings - IEEE International Conference on Data Mining, ICDM, pages 90– 97, 2005.

[25] Federico Giannoni, Marco Mancini, and Federico Marinelli. Anomaly Detection Models for IoT Time Series Data. Nov 2018.

[26] Arif Sari. A Review of Anomaly Detection Systems in Cloud Networks and Survey of Cloud Security Measures in Cloud Storage Applications. Journal of Information Security, 06(02):142– 154, 2015.