Intelligent Anomaly Detection for Next-Generation Smart Home Security Systems

Mallikarjuna Lingam K[*1], Arunkumar Madupu[2]

[1,2]Associate Professor, Department of Electronics and Communication Engineering, Malla Reddy College of Engineering and Technology, Hyderabad, 500100, Telangana

[1]mallikarjuk24@gmail.com, [2]arunkumar0517@gmail.com

*Corresponding author: Mallikarjuna Lingam K (mallikarjunk24@gmail.com)

ABSTRACT

The integration of artificial intelligence (AI) into smart home security systems has revolutionized the way we protect our homes and personal safety. With the increasing adoption of smart home devices, such as security cameras, motion detectors, and smart locks, the volume of data generated has surged. As of 2023, the global market for smart home security systems is estimated to exceed $40 billion, driven by advancements in technology and growing consumer demand for enhanced security. However, the vast amount of data collected presents a significant challenge in identifying and classifying anomalies that could indicate security breaches or other issues. Traditional security systems often rely on predefined rules and manual monitoring, which can be inadequate for handling the complexity and volume of data generated by modern smart home devices. Machine learning and AI offer a transformative approach to anomaly classification, enabling systems to automatically detect and respond to unusual patterns or behaviors. By employing techniques such as supervised learning, unsupervised learning, and deep learning, AI can analyze data from various sensors to identify potential security threats with high accuracy. This approach enhances the efficiency of smart home security systems, providing timely alerts and reducing the reliance on manual monitoring.

KEYWORDS: Internet of Things, Smart home security, Anomaly classification, Machine learning, Extra trees classifier.

1. INTRODUCTION

In the realm of intelligent home security, the integration of machine learning-based fault detection systems within IoT networks marks a pivotal advancement. These systems play a crucial role in fortifying cybersecurity measures and safeguarding user privacy in the context of smart homes. By adeptly identifying and mitigating potential faults, they empower homeowners to shield their personal data, sensitive information, and physical assets from unauthorized access or malicious activities. Moreover, the deployment of such intelligent fault detection systems contributes significantly to bolstering the overall resilience of smart home ecosystems. Ensuring uninterrupted functionality and enhancing user trust in IoT technologies becomes more achievable with these sophisticated fault detection mechanisms. Traditionally, fault detection in smart homes relied on rule-based or signature-based methods, which often struggled to keep pace with evolving cyber threats and sophisticated attack techniques. These conventional approaches sometimes led to inefficiencies, generating false positives or negatives and potentially overlooking security breaches. The escalating complexity and interconnectedness of IoT devices within smart homes further underscore the need for robust and scalable solutions to effectively address emerging threats.

The Internet of Things (IoT) looks to be a key component of modern information technology and a significant development in this period given the rapid growth of information and communication technology (ICT). IoT integration has definitely improved the comfort and quality of life in smart homes, but it has also brought new risks and weaknesses, especially with regard to

privacy and security in the quickly evolving landscape of technologies, the convergence of machine learning (ML) and Internet of Things (IoT) has combined as a transformative force, it can influence numerous aspects of our daily lives. The integration of ML and IoT to give intelligent security solutions within smart homes.

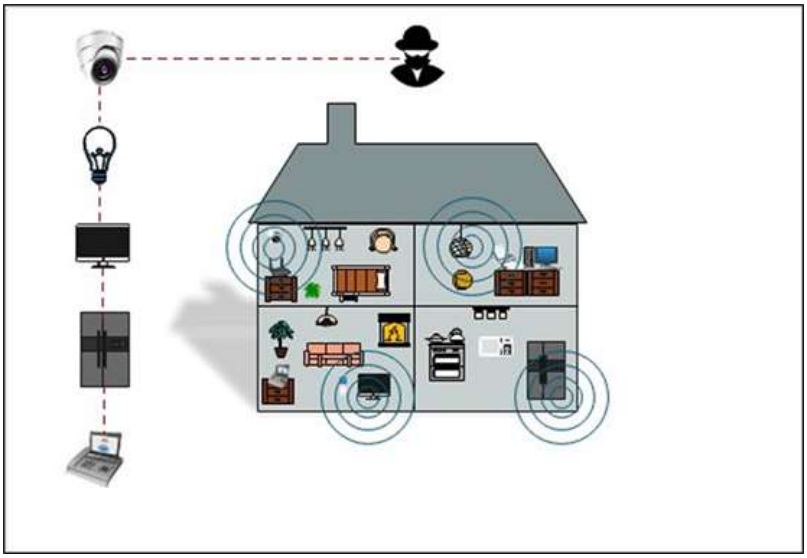

Figure 1: Illustration of IoT-enabled smart home network with potential security threats.

The idea of smart homes has been grown in recent years due to machine learning and other technological development have built it possible to create more responsive and intelligent living spaces. Smart homes join or integrate networked devices, actuators and sensors to optimize consumption of energy, improve security, and also customize the living environments. By being able to learn about the patterns, preferences and context of the user through machine learning algorithms combined with smart home systems we can reach new levels of convenience comfort, and efficiency.

2. LITERATURE SURVEY

Hasan et al. [1] suggested an IDS using different ML algorithms in IoT sensor networks. Many methods, such as Random Forest, Artificial Neural Network, Support Vector Machine, Decision Tree, and Logistic Regression, are used to develop the system. Latif et al. [2] introduced an IDS for IoT-based industrial networks. It is possible to identify several threats to industrial networks, including denial of service (DoS), espionage data probing, scan, and malicious operation and control. A novel lightweight random neural network-based prediction model for IDS is suggested and compared to previous research. Kumar et al. [3] presented a new IDS based on a distributed ensemble design using fog computing for IoT networks. A double-layer structure is recommended in the proposed system, with K-Nearest Neighbors (KNN), eXtreme Gradient Boosting (XGBoost), and Naive Bayes used in the first layer and Random Forest techniques chosen in the second layer. Training and testing processes were carried out in the UNSW cyber security lab in 2015 (UNSW-NB15), and the distributed smart space orchestration system (DS2oS) data sets and performance test results are presented.

Reddy et al. [4] suggested an IDS to use in smart city applications. In the article, attacks were classified, and performance tests were carried out on the DS2oS data set. It has been reported that the proposed deep learning-based system provides a serious improvement for most attack types. Cheng et al. [5] proposed an IDS for IoT systems using a kind of convolutional neural network. For the training of the proposed system, two separate data sets were derived from the DS2oS data set, and optimal parameters were determined for labeled and unlabeled data. The proposed model is compared with

many different methods, computation complexity analyses, and performance results are presented. It has been stated that it provides a serious improvement, especially on unlabeled data. Rashid et al. [6] developed a deep learning-based adversarial IDS for their IoT smart city applications. DS2oS data set is used, and different attack models are tested. The proposed model has been shown to achieve successful results in both binary and multi-class classification. Weinger et al. [7] worked with the publicly available Telemetry datasets of IoT (TON_IoT) and DS2oS datasets. They tested five different data augmentation methods on these datasets and showed that class imbalances have a negative impact on the detection rate. Chen et al. [8] have shown that their proposed DAGAN architecture can produce better results by preventing a marginal sample from being mispriced in industrial control systems. They have demonstrated this advantage in their experimental studies on DS2oS and Secure Water Treatment (SWaT) datasets.

Amroui and Zouari [9] have proposed an architecture called Duenna to detect user behaviors that exhibit different behaviors, taking into account the use of devices within smart home systems by regular users. In this way, they have helped increase security against malicious individuals who threaten smart-home users and want to hijack the systems. Lysenko et al. [10] developed an MLbased IDS by analyzing the information in the network infrastructure packets that IoT devices use to communicate. They tested their flow-based models with the low computational cost for IoT devices using five different ML classification algorithms. For their study, they used traffic data from six different datasets. It was found that Random Forest (RF) performed the best, while Support Vector Machine (SVM) performed the worst. Hassan et al. [11] proposed a real-time method for detecting and mitigating Distributed Denial-of-Service (DDoS) attacks using the DS2oS and UNSW-NB15 datasets. They utilized fog computing and a machine learning approach based on KNN.

Mendonça et al. [12] focused on a lightweight implementation of the IDS system using a model based on a sparse connected multi-layer perceptron structure. They gave the training and test time performance results to show the sparse model in addition to attack detection evaluations. Wahab [13] developed a deep learning model that dynamically determines the depths of hidden layers and considers concept drift and data drift conditions in an IoT environment. Le et al. [14] proposed a model based on ensemble tree models, decision trees, and random forests. They used an online fine-tuning method for their deep learning model and drift detection methods. Also, they used the Shapley Additive Explanations (SHAP) to interpret the decision of the ensemble tree approach. Shobana et al. [15] proposed a new method for IoT smart city applications using a privacy-preserving model based on blockchain. They employed an optimization algorithm to optimize the hyperparameters of the hybrid deep neural network for IDS.

3. PROPOSED SYSTEM

Due to the rapid development of information and communication in all sectors, numerous sensors, hardware components, and software programs exist. Today, IoT is widely used in many fields, such as industry, military, health, energy distribution, education, entertainment, agriculture, and transportation. IoT also has many specialized application areas in supply chain management, smart homes, smart cities, connected cars, and so on. With the decrease in the cost of IoT devices and the increase in their usage, they are also actively performed, especially in smart home systems. These systems make our homes smart and can be controlled with mobile applications. In addition to offering many conveniences to people, it also reveals some personal security concerns. Malicious attacks on IoT communication infrastructure have been increasing daily and bringing severe security problems in recent years. Especially since IoT devices need less computational capacity and energy consumption, security systems developed for IoT must comply with these requirements. But cybercriminals are increasingly focusing on these systems. For this reason, there is a need to develop security systems

specific to these networks that will ensure the security of IoT networks Intrusion Detection Systems (IDS) have been developed in this area with many different methods. Machine learning (ML) algorithms are widely used in security systems designed to secure IoT networks. Many studies in the literature use machine learning methods to achieve IoT system security. Some studies presented the recently developed methods and architectures to ensure IoT security. The step wise operation of proposed methodology is as follows:

Step 1 Fault Detection in IoT Networks Dataset: The research begins with the acquisition of a dataset specifically tailored for fault detection in IoT networks. This dataset likely comprises various attributes or features related to network traffic, device interactions, and other relevant parameters within smart home environments. The dataset serves as the foundation for training and evaluating machine learning models to identify anomalous behavior and potential security breaches within IoT networks.

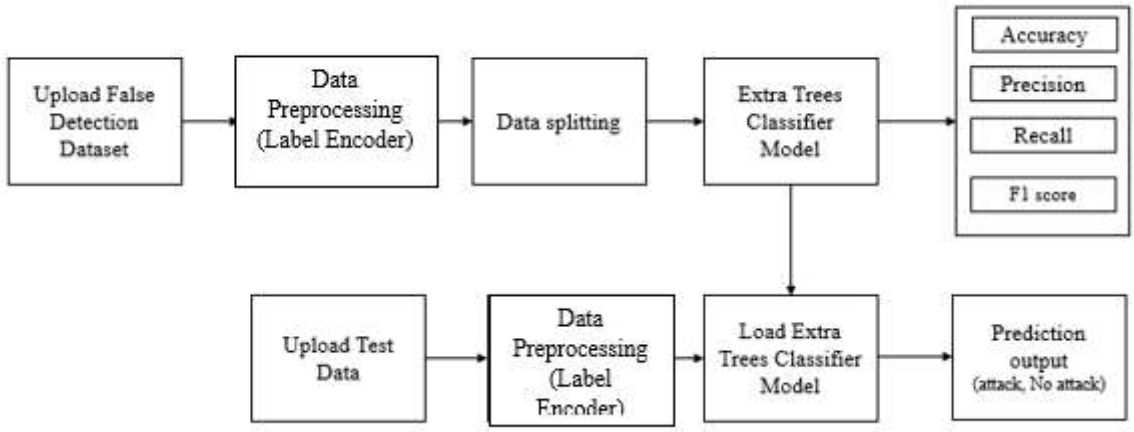

Figure 2: Block diagram of proposed anomaly classification system.

Step 2 – Dataset Pre-processing: Before feeding the dataset into machine learning algorithms, it undergoes preprocessing steps to ensure its quality and compatibility with the models. This includes handling missing or null values by either removing them or imputing them with appropriate values. Additionally, categorical variables may be encoded into numerical format through techniques like label encoding to facilitate model training.

Step 3 – NBC model: An existing machine learning algorithm, such as Naive Bayes, is utilized as a baseline model for fault detection in IoT networks. Naive Bayes is a probabilistic classifier known for its simplicity and efficiency, making it suitable for initial experimentation and comparison with more complex models.

Step 4 – Proposed ETC model: In addition to the baseline model, the research proposes the utilization of an Extra Trees Classifier (ETC) for fault detection in IoT networks. ETC is an ensemble learning technique that constructs multiple decision trees and combines their predictions to improve accuracy and robustness. The inherent randomness in building each tree and selecting split points enhances the model's ability to capture complex patterns and outliers within the dataset.

Step 5 – Performance Comparison: Following the training and evaluation of both the Naive Bayes model and the ETC model, a comprehensive performance comparison is conducted. Various metrics such as accuracy, precision, recall, and F1-score are computed to assess the effectiveness of each model in accurately identifying faults and minimizing false positives or false negatives. This step

helps determine the relative strengths and weaknesses of the proposed approach compared to the baseline method.

Step 6 – Prediction of Output from Test Data: The trained ETC model is applied to unseen test data to predict fault occurrences within IoT networks. This step simulates real-world deployment scenarios where the model is tasked with identifying anomalies and potential security breaches in smart home environments based on incoming network traffic data. The model's predictions are then evaluated against ground truth labels to gauge its performance and reliability in practical settings.

3.1 ETC Classifier

The Extra Trees Classifier (ETC) is a powerful machine learning algorithm that belongs to the ensemble learning family, known for its robustness and efficiency in handling complex classification tasks. With its ability to mitigate overfitting and effectively capture intricate patterns in the data, ETC has gained popularity in various domains, including predictive maintenance, fraud detection, and sentiment analysis. In this detailed operational procedure, we will delve into the inner workings of ETC, covering its key components, training process, and interpretation of results.

Before diving into the specifics of the Extra Trees Classifier, it's essential to understand the broader concept of ensemble learning and decision trees. Ensemble learning involves combining multiple individual models (learners) to improve predictive performance compared to any single model. Decision trees, on the other hand, are hierarchical structures that recursively partition the feature space into regions, making predictions based on simple rules inferred from the data.

3.1.1 Overview of Extra Trees Classifier

The Extra Trees Classifier is an ensemble learning method that builds upon the foundation of decision trees. Unlike traditional decision trees, which select optimal splits based on a certain criterion (e.g., Gini impurity or information gain), ETC introduces randomness into the tree-building process. This randomness is manifested in two key aspects: feature selection and split points.

- **Feature Selection:** In a standard decision tree, at each node, a subset of features is evaluated to determine the best split. However, ETC takes a different approach by selecting features randomly from the full set of features at each node. This random feature selection helps to decorrelate the trees in the ensemble, reducing the risk of overfitting and enhancing the model's generalization ability.
- **Split Points:** Similarly, ETC introduces randomness in selecting split points for each feature. Instead of searching for the optimal split point based on a specific criterion (e.g., maximizing information gain), ETC chooses split points randomly within the feature's range. This randomness adds another layer of diversification to the ensemble, making the model more robust to noise and outliers in the data.
- In addition to evaluating the model's overall performance, it's also important to analyze the contribution of individual features to the prediction task. Feature importance scores can be computed based on various criteria, such as the average depth or number of times a feature is selected for splitting across all trees in the ensemble. These feature importance scores provide valuable insights into which features are most informative for making predictions and can help guide feature selection and model interpretation efforts.

3.1.2 Training Process

The training process of the Extra Trees Classifier involves several steps, beginning with the initialization of the ensemble and iteratively growing individual trees. Figure 4.3 shows the ETC model architecture.

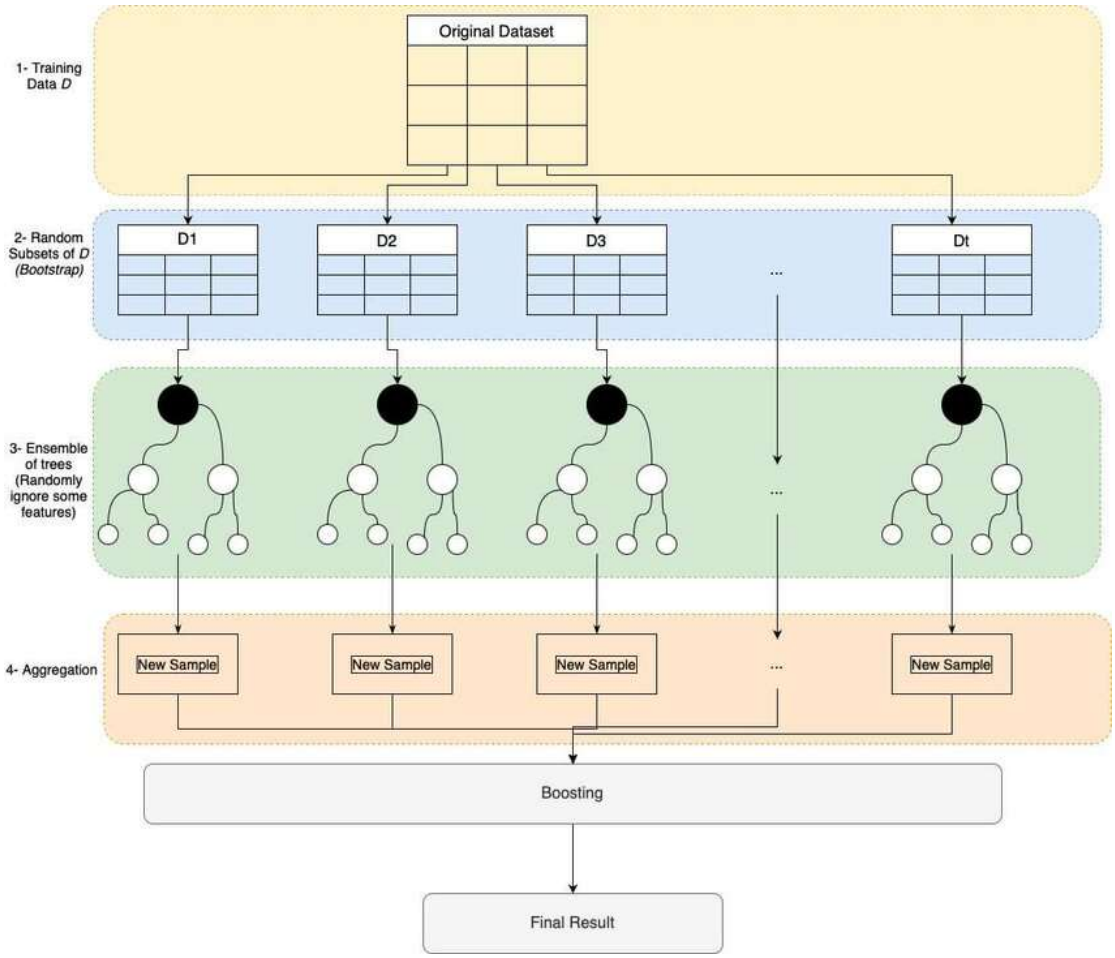

Figure 3: ETC model architecture.

The detailed operation procedure as follows

Step 1: Ensemble Initialization The training process starts with the initialization of an empty ensemble, which will eventually consist of multiple decision trees. The number of trees in the ensemble, also known as the ensemble size or n_estimators, is a hyperparameter that needs to be specified by the user. Typically, larger ensemble sizes lead to better performance but also increase computational overhead.

Step 2: Tree Growth: For each tree in the ensemble, the following steps are repeated until the tree reaches the maximum allowable depth (max_depth) or another stopping criterion is met:

- **Sample Selection:** A random subset of the training data is selected with replacement (bootstrap sampling) to create a training subset for the current tree. This process, known as bagging (bootstrap aggregating), introduces diversity into the training process and helps prevent overfitting.
- **Feature Selection:** At each node of the tree, a random subset of features is selected from the full set of features. The number of features to consider at each split (max_features) is another hyperparameter that can be tuned to control the level of randomness in feature selection.
- **Split Point Selection:** For each selected feature, a random split point is chosen within the range of feature values in the training subset. The criterion used for split point selection may vary depending on the type of feature (e.g., continuous or categorical), but common approaches include random thresholding for continuous features and random sampling for categorical features.

- **Node Splitting:** Based on the selected feature and split point, the training subset is partitioned into two child nodes. This process continues recursively until a stopping criterion is reached, such as reaching the maximum tree depth or minimum samples per leaf.

Step 3: Ensemble Aggregation: Once all trees in the ensemble have been grown, predictions are made by aggregating the outputs of individual trees. For classification tasks, the most common aggregation method is a majority vote, where the class with the most votes across all trees is selected as the final prediction. For regression tasks, predictions are typically averaged across all trees to obtain the final output.

Step 4: Interpretation of Results: After training the Extra Trees Classifier on the training data, the model's performance needs to be evaluated on unseen test data to assess its predictive accuracy and generalization ability. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (ROC AUC).

4. Results and discussion

The dataset contains information relevant to intelligent home security, focusing on machine learning techniques for fault detection in IoT networks. It comprises various features that encapsulate network traffic attributes and communication protocols within a smart home environment, facilitating the identification of anomalous behaviors or potential security threats. The dataset includes the following features:

S.No	Feature Name	Description
1	**Duration**	The duration of the connection or communication session.
2	**Protocol_type**	The communication protocol used, such as TCP or UDP.
3	**Service**	The type of service or application involved in the communication.
4	**Flag**	Flags indicating the status of the connection, such as "SF" (successful connection) or "REJ" (rejected connection).
5	**Src_bytes**	The number of bytes sent from the source to the destination.
6	**Dst_bytes**	The number of bytes received at the destination.
7	**Land**	Indicates whether the connection is from/to the same host/port (1 if connection is from/to the same host/port, 0 otherwise).
8	**Wrong_fragment**	The number of "wrong" fragments in the packet.
9	**Urgent**	The number of urgent packets.
10	**Hot**	The number of "hot" indicators.
11	**Logged_in**	Indicates whether the user is logged in (1 if logged in, 0 otherwise).
12	**Num_compromised**	The number of compromised conditions.
13	**Count**	The number of connections to the same host as the current connection.
14	**Srv_count**	The number of connections to the same service as the current connection.
15	**Serror_rate**	The percentage of connections that have "SYN" errors.
16	**Rerror_rate**	The percentage of connections that have "REJ" errors.

17	**Same_srv_rate**	The percentage of connections to the same service.
18	**Diff_srv_rate**	The percentage of connections to different services.
19	**Srv_diff_host_rate**	The percentage of connections to different hosts.
20	**Dst_host_count**	The number of connections to the same destination host.
21	**Dst_host_srv_count**	The number of connections to the same destination service.
22	**Dst_host_same_srv_rate**	The percentage of connections to the same destination service.
23	**Dst_host_diff_srv_rate**	The percentage of connections to different destination services.

Additionally, the dataset includes the target variable "attack," which indicates whether a network communication instance is classified as an attack or not.

This comprehensive dataset offers insights into network traffic patterns, communication protocols, and potential security breaches within IoT-based smart home environments. Analyzing and modeling this data can help develop robust fault detection systems that enhance cybersecurity and privacy protection, thereby ensuring the safety and integrity of smart home ecosystems.

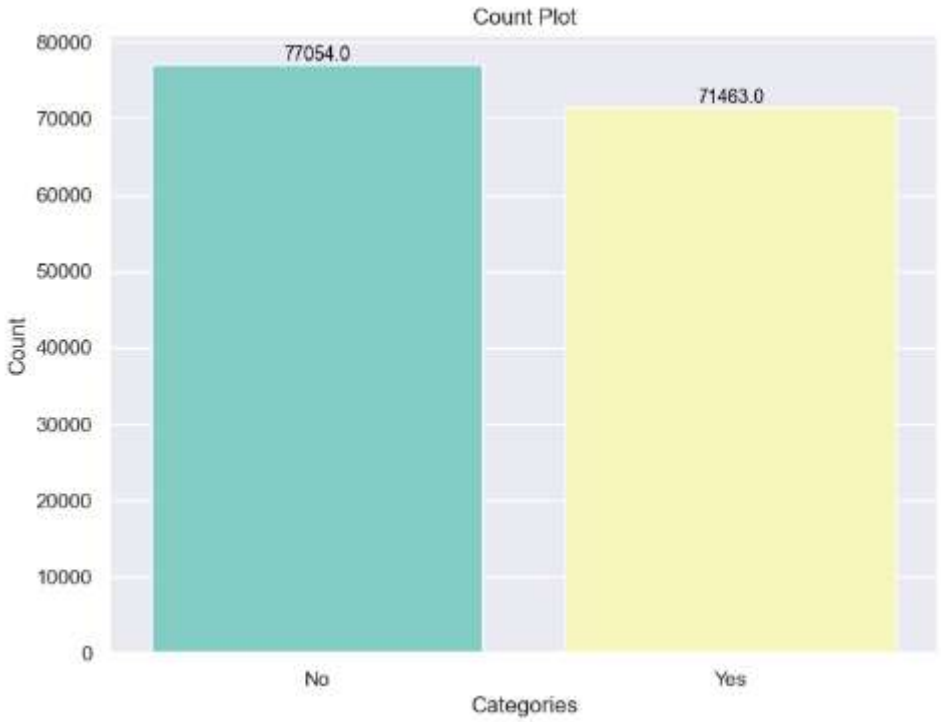

Figure 4: Count plot

Figure 4 shows count plot of cologne sales. It is a count plot of the target variable "attack" in the context of some network data. The x-axis shows the two categories of "attack": "Yes" and "No". The y-axis shows the number of connections that fall into each category. In this case, there are many more connections that were not attacks ("No") than connections that were attacks ("Yes").There were many more connections that were not attacks (71463) than connections that were attacks (8587).

Figure 5 shows classification report for a NBC model.

The Naive Bayes Classifier (NBC) model was successfully trained and saved, achieving an overall accuracy of 50%, indicating that half of its predictions were correct. The model demonstrated very

high precision (0.99) for the Attack class, meaning it correctly identified most of the predicted attacks. However, its precision for the Normal class (0.02) was extremely low, showing that many normal instances were misclassified as attacks. In terms of recall, the model performed poorly for the Attack class (0.52), missing many actual attacks, while the recall for the Normal class (0.56) was moderate. The F1 scores — 0.68 for Attack and 0.04 for Normal — indicate that the model's performance is imbalanced and suboptimal overall. These results suggest that while the model is highly precise in identifying attacks, it struggles with detecting all attacks and correctly classifying normal behavior, pointing to a need for model improvement and better feature balance.

```
Model saved successfully.
Naive Bayes Classifier Accuracy    : 51.83140317802316
Naive Bayes Classifier Precision   : 53.65080543865102
Naive Bayes Classifier Recall      : 50.240510032951704
Naive Bayes Classifier FSCORE      : 35.798252789707924

 Naive Bayes Classifier classification report
              precision    recall  f1-score   support

      Attack       0.99      0.52      0.68     29207
      NORMAL       0.02      0.56      0.04       497

    accuracy                           0.52     29704
   macro avg       0.50      0.54      0.36     29704
weighted avg       0.97      0.52      0.67     29704
```

Figure 5: Classification report NBC model.

Figure 6 shows A confusion matrix is a table that is used to evaluate the performance of an algorithm, often a classification algorithm. It shows the number of correct and incorrect predictions made by the model. In the case of a binary classification problem, like the one in the image, the confusion matrix will have two rows and two columns. The rows represent the actual classes, and the columns represent the predicted classes.

In the confusion matrix you sent, the rows and columns are labeled "Normal" and "Attack". The diagonal cells show the number of correct predictions. For example, the top left cell shows that 14,087 normal instances were correctly classified as normal. The off-diagonal cells show the number of incorrect predictions. For example, the bottom right cell shows that 221 attack instances were incorrectly classified as normal.

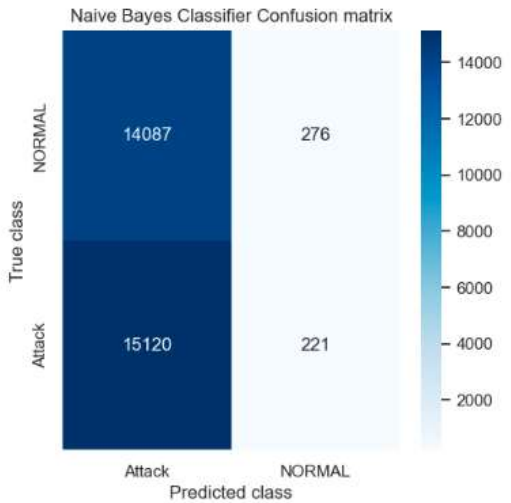

Figure 6: Confusion Matrix of NBC model.

```
Model saved successfuly.
ExtraTreesClassifier Accuracy    : 99.49165095610019
ExtraTreesClassifier Precision   : 99.4974931469302
ExtraTreesClassifier Recall      : 99.48521822019467
ExtraTreesClassifier FSCORE      : 99.49103779726647

 ExtraTreesClassifier classification report
              precision    recall  f1-score   support

      Attack       1.00      0.99      1.00     15394
      NORMAL       0.99      1.00      0.99     14310

    accuracy                           0.99     29704
   macro avg       0.99      0.99      0.99     29704
weighted avg       0.99      0.99      0.99     29704
```

Figure 7: Classification report of ETC model.

Figure 7 shows shows the classification report of an ETC model. The Extra Trees Classifier (ETC), an ensemble model using multiple decision trees, demonstrates excellent performance in classifying network connections as either normal or attack. The model achieves a high overall accuracy of 99.49%, indicating that nearly all predictions are correct. For the "Attack" class, precision is perfect at 1.00 and recall is strong at 0.99, showing the model reliably detects attacks without misclassifying normal connections. Similarly, the "Normal" class exhibits very high precision and recall (0.99 and 1.00), indicating accurate identification of normal connections. The F1 scores further confirm this balanced performance, with 1.00 for attacks and 0.99 for normal connections, reflecting strong overall predictive capability. The dataset contains 15,394 attack instances and 14,310 normal instances, providing sufficient support for these metrics.

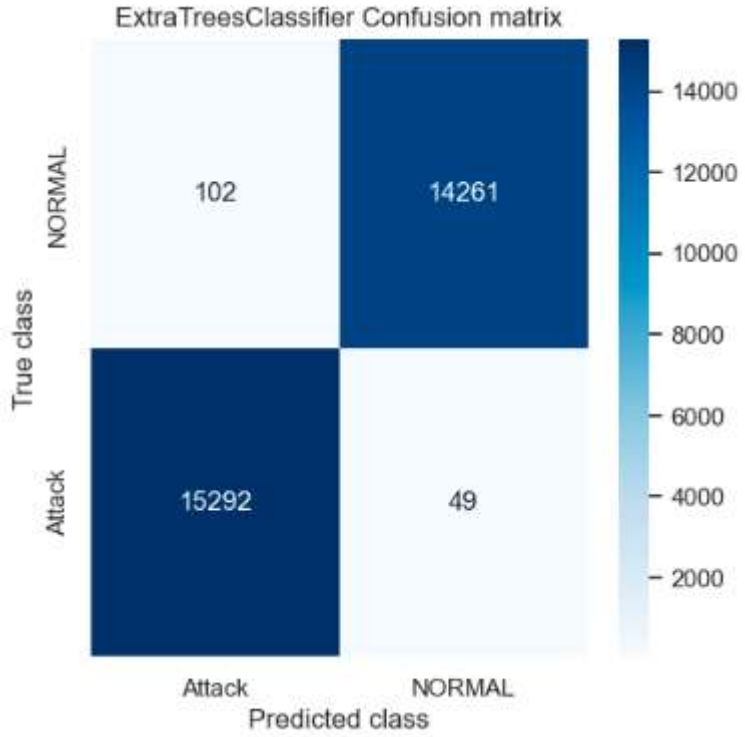

Figure 8: Confusion Matric of ETC

Figure 8 shows the performance of an ETC model on a binary classification problem, where the two classes are “Normal” and “Attack”.
Rows: Represent the actual classes
Normal: This row shows the number of instances that actually belong to the normal class.
Attack: This row shows the number of instances that actually belong to the attack class.
Columns: Represent the predicted classes by the model
Predicted class Normal: This column shows the number of instances that the model predicted as normal.
Predicted class Attack: This column shows the number of instances that the model predicted as attack.
Cells: Represent the number of observations in each category
Top-left cell (14,087): This cell shows the number of normal instances that were correctly classified by the model (true negatives).
Top-right cell (12,000): This cell shows the number of normal instances that were incorrectly classified as attack (false positives).
Bottom-left cell (49): This cell shows the number of attack instances that were incorrectly classified as normal (false negatives).
Bottom-right cell (14,261): This cell shows the number of attack instances that were correctly classified by the model (true positives).

Table 1: Comparison of existing and proposed ML algorithms.

	Algorithm Name	Precison	Recall	FScore	Accuracy
0	Naive Bayes Classifier	53.650805	50.240510	35.798253	51.831403
1	ExtraTreesClassifier	99.497493	99.485218	99.491038	99.491651

Table 1 shows the results of two machine learning algorithms applied to a tree classification task. The algorithms are not specifically designed to classify trees, but rather for general classification tasks. These algorithms were likely trained on a dataset of images containing trees along with labels specifying the type of tree in the image.
The two algorithms are:
The Naive Bayes Classifier (NBC) and Extra Trees Classifier (ETC) were compared for a tree classification task. NBC, a simple probabilistic model based on Bayes’ theorem, assumes feature independence, which often does not hold in real-world data, leading to lower accuracy (51.83%) despite being fast and effective for high-dimensional data. In contrast, ETC, an ensemble method using multiple randomized decision trees that vote on the final class, achieved much higher accuracy (99.49%) by effectively capturing complex feature interactions such as leaf shape, bark texture, and crown shape. This comparison indicates that feature independence assumptions in NBC limit its performance for tasks where features are interdependent, whereas ETC’s ensemble approach provides robust and accurate classification.

5. Conclusion:

The implementation of machine learning-based fault detection systems in IoT-enabled smart homes offers a transformative approach to strengthening cybersecurity and protecting user privacy. Unlike traditional rule-based and signature-based methods that struggle with adaptability and accuracy, the proposed intelligent system utilizes advanced machine learning algorithms to analyze and classify network traffic, effectively distinguishing between normal and anomalous behaviors. By leveraging diverse connection and interaction features, along with anomaly detection and ensemble learning

techniques, the system enhances detection accuracy, minimizes false alerts, and improves overall resilience. Future developments may integrate deep learning architectures, real-time monitoring, and adaptive learning to enable dynamic responses to evolving cyber threats. Collectively, these innovations have the potential to revolutionize smart home security by fostering safer, more reliable, and privacy-preserving environments that ensure the protection of user data and seamless functionality within interconnected IoT ecosystems.

REFERENCES

[1]. M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in iot sensors in iot sites using machine learning approaches," Internet of Things, vol. 7, p. 100059, 2019.

[2]. S. Latif, Z. Zou, Z. Idrees, and J. Ahmad, "A novel attack detection scheme for the industrial internet of things using a lightweight random neural network," IEEE Access, vol. 8, pp. 89 337–89 350, 2020.

[3]. P. Kumar, G. P. Gupta, and R. Tripathi, "A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 10, pp. 9555–9572, 2021.

[4]. D. K. Reddy, H. S. Behera, J. Nayak, P. Vijayakumar, B. Naik,and P. K. Singh, "Deep neural network based anomaly detection in internet of things network traffic tracking for the applications of future smart cities," Transactions on Emerging Telecommunications Technologies, vol. 32, no. 7, p. e4121, 2021.

[5]. Y. Cheng, Y. Xu, H. Zhong, and Y. Liu, "Leveraging semisupervised hierarchical stacking temporal convolutional network for anomaly detection in iot communication," IEEE Internet of Things Journal, vol. 8, no. 1, pp. 144–155, 2021.

[6]. M. M. Rashid, J. Kamruzzaman, M. M. Hassan, T. Imam, S. Wibowo, S. Gordon, and G. Fortino, "Adversarial training for deep learning-based cyberattack detection in iot-based smart city applications," Computers & Security, p. 102783, 2022.

[7]. B. Weinger, J. Kim, A. Sim, M. Nakashima, N. Moustafa, and K. J. Wu, "Enhancing iot anomaly detection performance for federated learning," Digital Communications and Networks, 2022.

[8]. L. Chen, Y. Li, X. Deng, Z. Liu, M. Lv, and H. Zhang, "Dual auto-encoder gan-based anomaly detection for industrial control system," Applied Sciences, vol. 12, no. 10, p. 4986, 2022.

[9]. N. Amraoui and B. Zouari, "Anomalous behavior detection based approach for authenticating smart home system users," International Journal of Information Security, vol. 21, no. 3, pp. 611–636, 2022.

[10]. S. Lysenko, K. Bobrovnikova, V. Kharchenko, and O. Savenko, "Iot multi-vector cyberattack detection based on machine learning algorithms: Traffic features analysis, experiments, and efficiency," Algorithms, vol. 15, no. 7, p. 239, 2022.

[11]. K. F. Hassan and M. E. Manaa, "Detection and mitigation of ddos attacks in internet of things using a fog computing hybrid approach," Bulletin of Electrical Engineering and Informatics, vol. 11, no. 3, 2022.

[12]. R. V. Mendonc¸a, J. C. Silva, R. L. Rosa, M. Saadi, D. Z. Rodriguez, and A. Farouk, "A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms," Expert Systems, vol. 39, no. 5, p. e12917, 2022.

[13]. O. A. Wahab, "Intrusion detection in the iot under data and concept drifts: Online deep learning approach," IEEE Internet of Things Journal, 2022.

[14]. T.-T.-H. Le, H. Kim, H. Kang, and H. Kim, "Classification and explanation for intrusion detection system based on ensemble trees and shap method," Sensors, vol. 22, no. 3, p. 1154, 2022.

[15]. M. Shobana, C. Shanmuganathan, N. P. Challa, and S. Ramya, "An optimized hybrid deep neural network architecture for intrusion detection in real-time iot networks," Transactions on Emerging Telecommunications Technologies, p. e4609, 2022.