Journal of Engineering and Technology for Industrial Applications

ISSN ONLINE: 2447-0228

ITEGAM-JETIA

Manaus, v.12 n.58, p.1276-1286. March/April, 2026.
DOI: https://doi.org/10.5935/jetia.v12i58.3310

RESEARCH ARTICLE OPEN ACCESS

EFFICIENT CYBER THREAT DETECTION IN SMART HOME IOT NETWORKS USING MACHINE LEARNING AND EXPLAINABLE AI

Pritimayee Satapathy*[1], Prafulla Kumar Behera[2]

[1, 2]Utkal University, Bhubaneswar, Odisha, India

[1]https://orcid.org/0000-0002-1295-0981, [2]https://orcid.org/0000-0001-5016-0692

Email:*pritimayees@gmail.com, pkbehera.cs@utkaluniversity.ac.in

ARTICLE INFO

Article History
Received: January 14, 2026
Reviewed: February 18, 2026
Accepted: March 28, 2026
Published: April 30, 2026

Keywords:
Smart home,
Explainable AI,
IoT Security,
Machine Learning,
Hybrid Feature Selection.

ABSTRACT

Smart homes are increasingly common, making it more important than ever to safeguard them against cyber threats. In this work, we develop an improved anomaly-based Intrusion Detection System (IDS) for smart home IoT networks by combining machine learning (ML) techniques with advanced Feature Selection (FS) and Explainable Artificial Intelligence (XAI). We propose an FS approach that finds the intersection of important features identified by Recursive Feature Elimination with Cross-Validation (RFECV) using two different base learners: a Decision Tree (DT) and a Light Gradient Boosting Machine (LGBM). This yields a compact subset of 12 network traffic features. Using this significantly reduced feature set, our lightweight LGBM model achieves 99.86% detection accuracy, an F1-score of 99.93%, and a recall of 99.96% on the IoTID20 dataset. The false positive rate is greatly reduced, and computational cost is also minimized. To enhance model transparency, we integrate XAI tools, Local Interpretable Model-Agnostic Explanations (LIME) provide clear instance-level explanations, and Shapley Additive Explanations (SHAP), which provide global explanations of the classifier's decisions. Experimental results demonstrate that combining FS with XAI can improve both the efficacy and the usability of IoT IDS. Our hybrid model outperforms several recent deep learning approaches in precision and recall, while being far more interpretable. Overall, the proposed method yields high detection rates with improved transparency, making it well-suited for resource-constrained smart home environments.

I. INTRODUCTION

The Internet of Things (IoT) connects numerous everyday devices, including smart thermostats, cameras, voice assistants, sensors, and more, within smart home environments. However, increased connectivity also broadens the attack surface available to malicious actors. An Intrusion Detection System (IDS) monitors network traffic to identify abnormal behavior that may signal cyberattacks. Traditional signature-based IDS methods struggle to detect novel or evolving threats, making anomaly-based detection more suitable for the dynamic context of smart home IoT installations [1]. These technologies enable greater automation and convenience in smart homes, but they also generate vast amounts of complex data. Each device can produce hundreds of features (from network traffic, device logs, sensor readings, etc.), making it challenging to discover the small patterns indicative of malicious activity [2].

In other words, anomaly-based IDS models can potentially catch previously unseen attacks, but they face challenges like the high dimensionality of IoT data and a lack of explainability in complex models [3]. Smart home networks generate a vast amount of heterogeneous data from sensors, cameras, appliances, etc., which typically results in high-dimensional feature spaces. Many features may be redundant or irrelevant, slowing down learning algorithms and potentially degrading detection performance [4]. Moreover, complex models such as Deep Neural Networks (DNN) that achieve high accuracy often act as "black boxes," leaving administrators unsure why a certain alert was raised. This lack of interpretability can reduce trust in the IDS and make it harder to refine the system.

Therefore, there is a need for an attack detection approach that not only attains high accuracy but also provides clarity about its decision process. Recent research has applied Explainable Artificial Intelligence (XAI) techniques to improve the transparency of ML models in IoT security [5], [3]. According to [6], also emphasized on advanced feature engineering using DCNN method for enhancing IoT security. In this work, we address the dual challenges of dimensionality and lack of interpretability by integrating an efficient FS strategy with XAI methods. Research on enhancing IoT security has grown rapidly in recent years, yielding a variety of approaches. Many works leverage traditional ML, DL, or hybrid techniques to improve detection accuracy in IoT environments.

For example, by [7] proposed a hybrid DL method using Convolutional Neural Network- Long Short-Term Memory (CNN-LSTM) for enhancing the security of Internet of Medical Things (IoMT). They implemented the IDS on a Raspberry Pi device using a fog computing architecture. In [8] suggested an approach to enhance the IDS using the FS method SHAP-based REFCV with extreme Gradient Boosting (XGBoost) model on the UNSW-NB15 dataset. According to [9] proposed a hybrid FS approach using statistical methods and a metaheuristic algorithm to identify the most relevant features. They applied an LSTM-based model on two publicly available datasets, NF-BoT-IoT-v2 and IoTID20, with a reduced feature set.

For [10] suggested a framework for anomaly intrusion detection using RFE integrated with ML paradigms, and a ridge regression merged into DL models. They used a real-time IoMT dataset, WUSTL-EHMS. This framework achieved a training accuracy of 99 % and a testing accuracy of 97.85 % using the RFE-DT with a False Alarm Rate(FAR) of 0.03. In the past year, there have been promising developments incorporating Graph Neural Networks (GNN) and attention mechanisms into IDS. According to [11] employed a GNN for IoT intrusion detection, achieving superior accuracy and outperforming other methods on benchmarks. These advanced systems typically require significant computational resources and lack straightforward interpretability.

By contrast, our approach shows that a carefully engineered classical ML pipeline can reach comparable performance to GNN or attention-based DL models, while being lighter-weight and inherently more interpretable. Specifically, we employ an RFECV technique [12] using two different base learners [13], [14] to identify a robust subset of features. Intersecting the important features from these two models yields a drastically reduced feature set that still preserves the crucial information needed to detect attacks in IoT networks. This significant reduction in dimensionality lowers computational overhead and training time, which is crucial for real-time detection in IoT settings.

To ensure that the model's decisions remain transparent, we integrate [15], [16] tools to explain the contribution of each feature to the model's predictions. This combination of local (instance-level) and global (feature importance) explainability effectively transforms our IDS into a "glass box" system that can justify its reasoning for each alert, thereby boosting user trust and facilitating system refinement. In summary, the proposed methodology addresses the gaps identified in prior works by jointly improving detection accuracy, efficiency, and explainability in a single ML based framework for enhancing IoT security. Our main contributions can be summarized as follows:

- Hybrid Feature Selection: We propose a novel hybrid FS method that combines RFECV outputs from two different estimators (DT and LGBM). This approach effectively reduces the feature set from 83 down to 12 features without sacrificing detection capability. The resulting lightweight model yields high accuracy with a very low False Positive Rate (FPR) and shorter training time.
- Explainability: We integrate LIME and SHAP to provide both local and global explanations for the model's predictions. LIME offers instance-level interpretability by highlighting which features contribute most to a given classification, whereas SHAP summary plots provide an overview of feature importance and effect across all instances. By combining these XAI techniques, our framework produces not just predictions but actionable explanations for each alert, improving user trust and insight.
- Comprehensive Evaluation: We conduct extensive experiments on a real-world IoT intrusion dataset (IoTID20) to validate our framework. We compare our approach against several existing methods from the literature, including recent DL-based approaches. The results show that our hybrid model achieves competitive or superior detection rates while using far fewer features.
- Practicality for Smart Homes: We demonstrate the practicality of our solution for smart home deployment using lightweight ML algorithms. We analyse the system's runtime performance and resource usage, showing that it can be deployed on resource-constrained edge devices (such as a Raspberry Pi) with manageable inference latency. This makes our approach feasible for real-time attack detection in IoT home networks.

II. PROPOSED METHODOLOGY

This section describes our proposed attack detection framework in detail. We outline the dataset characteristics and preprocessing steps, the ML algorithms and FS techniques used, the hybrid FS approach we developed, the performance evaluation metrics, and the XAI tools integrated into the system. Figure 1 depicts the stages of the proposed methodology. In [17], the authors have utilized the IoTID20 dataset because of its significant citation rate in scholarly indexing repositories. Accordingly, we have also taken the IoTID20 dataset for our study. It addresses a wide range of attack types, guaranteeing their applicability and relevance in studies that focus on smart home environments.

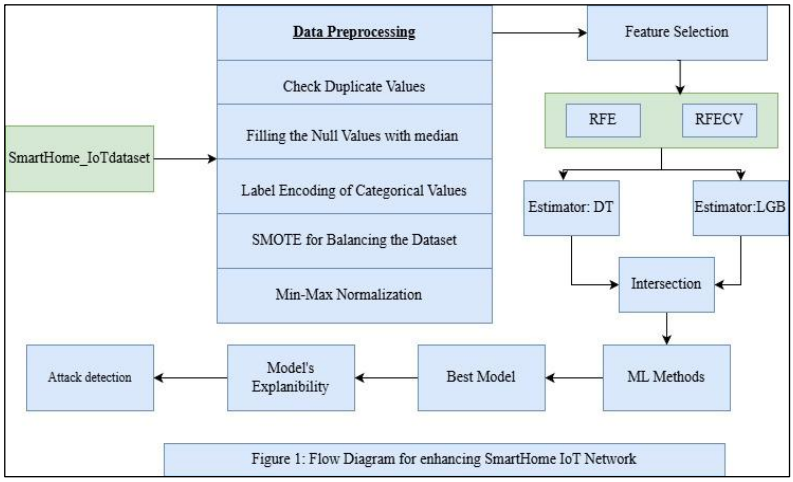

Figure 1: Flow for enhancing SmartHome IoT Network.
Source: Authors, (2026).

II.1 DATASET DESCRIPTION

We evaluate our approach using the IoTID20 dataset, which is a recent intrusion detection dataset specifically designed for smart home IoT environments [18]. IoTID20 captures network traffic from a typical smart home setup. In the data collection scenario, a standard smart home network was established, consisting of an SKT NGU smart home hub device and an EZVIZ Wi-Fi camera as the primary IoT devices, connected through a home router. Additional devices (computers, tablets, smartphones) were present in the network to generate background traffic and launch attacks. The SKT NGU and the camera acted as victims (targets of attacks), while the other devices functioned as attackers in various experiments [19]. The IoTID20 dataset provides labelled instances of both normal and attack traffic. It contains a total of 585,710 malicious traffic records and 40,073 benign records.

II.2 DATA PREPROCESSING

Before model training and FS, we performed comprehensive data preprocessing on the IoTID20 dataset. We utilized the Python built-in package (Pandas) [20] and NumPy [21] frameworks in the preprocessing step of our experiment. The following steps were applied.

II.2.1 Data Cleaning

We removed duplicate records and any irrelevant fields. In IoTID20, fields like Flow_ID, Dst_IP, Src_IP, and Timestamp were dropped, as they are identifiers or time indices that do not generalize to new data (and could leak information if not handled properly).

II.2.2 Handling Missing and Infinite Values

We replaced any occurrences of infinite values with NaN and then handled missing values. Specifically, for any feature with more than 5% missing values, we imputed the missing entries with the median of that feature.

II.2.3 Label Encoding

Label encoding has been used to convert the categorical variables into a numerical format so that ML algorithms can use them. We converted the labels into binary numeric form: 0 for Normal and 1 for Anomaly.

II.2.4 Train-Test Split

We partitioned the data into training and testing sets. Using a stratified split, we allocated 80% of the data for training and 20% for final testing. The stratification ensures that even minority attack categories are represented proportionally in both training and test sets.

II.2.5 Data Cleaning Class Imbalance Handling

The training data remained imbalanced after splitting. For balancing the dataset, we applied SMOTE (Synthetic Minority Over-sampling Technique) [22] on the training set. SMOTE generates synthetic examples of the minority class (normal traffic in this case) to balance the class distribution. The test set was kept at the original distribution for a realistic evaluation.

II.2.6 Normalization

Finally, we normalized the feature values using Min–Max scaling to the [0,1] range. This ensures that features on different scales (e.g., byte counts vs. time durations) all contribute proportionately in model training. After these preprocessing steps, the data were ready for FS and model training. By cleaning and resampling the data, we reduced noise and bias, providing a solid foundation for the subsequent FS phase.

II.3 MACHINE LEARNING TECHNIQUES

We implemented our ML models using the Scikit-Learn library in Python [23]. Our framework employs two primary algorithms as base learners: DT and LGBM. These were chosen for their complementary characteristics. DTs are fast and interpretable, whereas LGBM is a state-of-the-art gradient boosting method known for high accuracy and efficiency on large data.

II.3.1 Decision Tree

A DT is a non-parametric supervised learning method used for classification (and regression). It learns decision rules inferred from the data features to predict the target label. Each internal node of the tree represents a feature test; each branch represents an outcome of the test, and each leaf node represents a class prediction (in our case, attack or normal). DTs are relatively fast to train and can handle datasets with many features, but a single tree can overfit if not pruned. DTs also provide an importance score for each feature, which we will leverage in FS [10], [13].

II.3.2 Light Gradient Boosting Algorithm

LGBM is a gradient boosting framework that uses tree-based learners. It is an optimized implementation of Gradient Boosted Decision Trees (GBDT), known for its speed and efficiency, especially on large datasets and high-dimensional data [24].

LGBM introduces techniques like histogram-based splitting and leaf-wise tree growth with depth constraints, which make it faster and sometimes more accurate than traditional XGBoost for certain tasks. In our context, LGBM serves both as a base estimator for FS and as the final classifier after feature reduction. We use default hyperparameters for LGBM, with a fixed random_state=42 for reproducibility.

II.4 FEATURE SELECTION

To minimize dimensionality and improve model generalizability, we apply multiple FS techniques after the basic preprocessing. Specifically, we experiment with Recursive Feature Elimination (RFE), RFE with cross-validation (RFECV), and our proposed hybrid FS method that combines RFECV results from two estimators (DT and LGBM).

II.4.1 Recursive Feature Elimination

RFE is a wrapper FS method that recursively removes the least important features until a desired number of features is reached. It works by training a model and, at each iteration, dropping the feature with the smallest importance as determined by the feature importance metric. We employed RFE with estimators such as DT and LGB to find sets of eight features to customize the FS in this study [10].

II.4.2 Recursive Feature Elimination

RFECV is an enhanced RFE that uses cross-validation at each stage of elimination to confirm the model's efficacy in choosing the optimal number of features. This approach uses cross-validation to assess model performance at each iteration and find the optimal feature set size by recursively eliminating features that are of the least importance based on estimator-specific coefficients or feature importance metrics. In our research, we have employed 5-fold cross-validation within the RFECV framework so that robust and unbiased FS is ensured. This process not only reduces overfitting and improves generalization but also enhances model interpretability by eliminating redundant or irrelevant attributes [12].

II.4.3 Proposed Hybrid FS

We propose a hybrid FS that utilizes RFECV by taking the common attributes of two base estimators. We have used two distinct tree-based ML methods, namely DT and LGBM. The dataset is first divided into training and testing subsets, and then each classifier independently applies RFECV to determine the most important features. The intersection of selected features from both models is computed to retain only the most significant features. If some common features are found, then the final classification model, which can be either LGBM or DT, is trained using these common features. The performance of the resulting model is evaluated by using standard metrics such as accuracy, precision, FPR, training time, testing time, recall, and F1-score. This algorithm is specified below:

Algoithm 1: Hybrid Feature Selection Using Two Estimators

- Input: All features post pre-processing
- Output: Common features obtained by using two estimators

1. *Split the dataset into training and testing sets:x_train, x_test, y_train, y_test ← train_test_split (x, y)*
2. *Initialize Decision Tree Classifier: dt ← DecisionTreeClassifier ()*
3. *Perform Recursive Feature Elimination with cross-validation using Decision Tree: rfecv_dt ← RFECV (estimator=DecisionTreeClassifier (), step=1, cv=StratifiedKFold (5), scoring='accuracy')*
4. *Selected_features_dt ←: Features selected by rfecv_dt*
5. *Initialize LightGBM Classifier: lgbm← LGBMClassifier ()*
6. *Perform Recursive Feature Elimination with cross validation using LightGBM: rfecv_lgbm ← RFECV (estimator = LGBMClassifier (), step=1, cv=StratifiedKFold (5), scoring='accuracy')*
7. *Intersection of selected features:*
   common_features ← selected_features_dt ∩ selected_features_lgbm

8. *if common_features ≠ Ø then*
    a. *Extract x_train_common ← x_train[common_features]*
    b. *Extract x_test_common ← x_test[common_features]*
    c. *Initialize final classifier:*
    d. *final_model ← LGBMClassifier () or DecisionTreeClassifier ()*
    e. *Train final_model on x_train_common and y_train*
    f. *Predict y_pred ← final_model. predict(x_test_common)*
    g. *Evaluate performance metrics*
9. *else*
10. *Output: "No common features found. Try selecting more features."*

Using this hybrid FS approach, we drastically reduced the feature count while preserving the most significant predictors. This not only lowers computational cost but can also improve detection performance by eliminating noisy features. Next, we discuss the metrics used to evaluate our models.

II.5 PERFORMANCE EVALUATION METRICS

Because accuracy alone is not enough for the IoT network in smart homes, a thorough evaluation of the ML technique is required. A confusion matrix is a useful evaluation metric for illustrating how well an ML algorithm performs. It presents a table that contrasts the model's predictions with the actual class labels. Below is an explanation of the significance of the four linked variables that were obtained from the confusion matrix: False Negative (FN), True Negative (TN), False Positive (FP), and True Positive (TP).

TP: When an IDS identifies a scenario as an attack, it means that a real attack has been found.
FP: When an instance is flagged as a danger by the IDS, even after it has been confirmed to be safe.
TN: While the IDS recognizes the usual scenario, additional study reveals that the occurrence is normal.
FN: Some cases were misclassified by IDS as malicious attacks, whereas they were later confirmed to be normal occurrences.

Our ML models are assessed using a variety of assessment metrics, including accuracy, precision, FPR, recall, and F1 score, in order to validate our suggested methodology. An overview of these five metrics is given below:

(a)Accuracy: Out of all the predictions, the percentage of true predictions is what determines a prediction system's accuracy.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

(b)Precision: The precision metric is determined by the ratio of correctly predicted true positive cases to the total predicted positive cases

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

(c)Recall: Recall, detection rate, or sensitivity denotes a model's ability to accurately identify and classify attacks. The recall for a specific label is defined as the ratio of true positives to the total number of actual positives.

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

(d)F1-Score: The F1-score is derived by computing the harmonic mean of precision with recall. The maximum value is achieved when precision equals recall.

$$F1 = 2.\frac{Precision*Recall}{Precision+Recall} \tag{4}$$

(e)FPR: The FPR is computed using the following formula.

$$FPR = \frac{FP}{TN+FP} \tag{5}$$

(f)Training Time: The amount of time needed to train ML models using a given dataset is measured in seconds (s). The time it takes to go from starting to finishing the training is the key factor in determining this period.
(g)Testing Time: By calculating the difference between the end of the training phase and the end of the prediction phase, the amount of time it took for machine learning models to predict the test data of a particular dataset was evaluated.

II.6 EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI) INTEGRATION

To address the interpretability challenge in our IDS, we integrate two model-agnostic XAI techniques: LIME [15] and SHAP [16]. These tools help explain the predictions of our ML models, ensuring the IDS is not a “black box” and providing insight into why a given network instance is classified as an attack or normal.

II.6.1 Local Interpretable Model-Agnostic Explanations

LIME generates an explanation for individual predictions by learning a simple interpretable model (like a linear model) locally around the prediction. In practice, for a given instance, LIME perturbs the input features and observes how the classifier's prediction changes, then fits a sparse linear model to mimic those changes. The result is a set of feature contributions (weights) that approximately explain the original model's decision for that instance. LIME has been used in prior IoT security research. In our case, we apply LIME to instances of interest (e.g., alarms triggered by the IDS) to highlight which features (such as packet rate, port numbers, etc.) contributed most to the classification of "Attack" or "Normal" [15], [9].

II.6.2 Shapley Additive Explanations

SHAP is a unified framework based on cooperative game theory to explain model outputs. It assigns each feature a Shapley value representing its contribution to the difference between the actual prediction and the average prediction. SHAP values can be computed for each feature of each instance, and they have a solid theoretical foundation of fairness. We use the Tree SHAP algorithm (optimized for tree-based models like LGBM and DT) to compute feature attributions for our trained models. SHAP can provide both local explanations (for instance) and global insight via summary plots that show how features affect predictions across the dataset [16]. By incorporating LIME and SHAP, our proposed approach uses two levels of transparency: (a) local interpretability for specific events and (b) global interpretability.

We also performed a quantitative stability analysis, checking that the set of top features and their importance rankings remain consistent across different cross-validation folds and model variations – this increases confidence that our explanations are robust and not artifacts of a particular train-test split. In summary, the integration of XAI techniques ensures that our high-performing IDS does not operate as an inscrutable black box. Instead, it can explain its rationale both at the micro level (for instance) and macro level (across the dataset), thereby increasing user confidence and aiding in system diagnostics.

III. EXPERIMENTAL RESULTS AND DISCUSSION

The experimental results are thoroughly analysed in this section, which is primarily split into two separate parts. The first part describes the results obtained from ML methods using different FS techniques. Models' explainability is discussed in the second part. IoTID20, a dataset generated by a smart-home environment, is utilized to assess the suggested method. The experiment is conducted on a personal computer workstation with a 64-bit Microsoft Windows 10 operating system, equipped with an Intel(R) Core (TM) i7-3777OS CPU, clocked at 3.10 GHz, and possessing an overall storage capacity of 16 GB.

The Scikit-Learn Python Framework is utilized to create, optimize, test, and validate ML models [23]. No GPU acceleration was used, as our models (DT and LGBM) are not computationally intensive. The entire training and FS process (including 5-fold cross-validation in RFECV) took approximately 2 hours on this machine, while training the final LGBM model on the selected 12 features required only a few minutes. Hyperparameters were kept at their defaults except for setting random_state=42 for reproducibility. The selected features performed by RFE, RFECV, and hybrid FS using estimator DT and LGBM are depicted in Table 1.

Table 1: Selected features per model using RFE, RFECV, and hybrid FS method.

Model	FS	Features Name
DT	RFE	Src_Port, Dst_Port, Flow_Duration, Bwd_IAT_Tot, Pkt_Len_Max, ACK_Flag_Cnt, Init_Bwd_Win_Byts, Idle_Max
LGBM	RFE	Src_Port, Dst_Port, Flow_Duration, Flow_Pkts/s, Flow_IAT_Max, Flow_IAT_Min, Pkt_Size_Avg, Init_Bwd_Win_Byts
DT	RFECV	Src_Port, Dst_Port, Flow_Duration, Flow_Byts/s, Flow_Pkts/s, Flow_IAT_Max, Flow_IAT_Min, Bwd_IAT_Tot, Bwd_Header_Len, Bwd_Pkts/s, Pkt_Len_Min, Pkt_Len_Max, Pkt_Len_Var, ACK_Flag_Cnt, Init_Bwd_Win_Byts, Idle_Max
LGBM	RFECV	Src_Port, Dst_Port, Flow_Duration, Tot_Bwd_Pkts, TotLen_Fwd_Pkts, TotLen_Bwd_Pkts, Bwd_Pkt_Len_Max, Bwd_Pkt_Len_Min, Bwd_Pkt_Len_Std, Flow_Byts/s, Flow_Pkts/s, Flow_IAT_Mean, Flow_IAT_Max, Flow_IAT_Min, Bwd_Header_Len, Bwd_Pkts/s, Pkt_Len_Min, Pkt_Len_Std, ACK_Flag_Cnt, Pkt_Size_Avg, Init_Bwd_Win_Byts, Idle_Mean
LGBM	Hybrid FS	Bwd_Header_Len, Init_Bwd_Win_Byts, Dst_Port, Src_Port, Flow_Pkts/s, Flow_IAT_Max, Pkt_Len_Min, ACK_Flag_Cnt, Bwd_Pkts/s, Flow_Byts/s, Flow_IAT_Min, Flow_Duration
DT	Hybrid FS	Pkt_Len_Min, Bwd_Header_Len, ACK_Flag_Cnt, Bwd_Pkts/s, Flow_IAT_Max, Flow_IAT_Min, Src_Port, Dst_Port, Flow_Duration, Flow_Byts/s, Init_Bwd_Win_Byts, Flow_Pkts/s

Source: Authors, (2026).

III.1 PERFORMANCE ANALYSIS OF REF, RFECV, AND PROPOSED HYBRID FS METHODS

We first compare the performance of the models using different FS strategies: (1) RFE with DT, (2) RFECV with DT, (3) RFE with LGBM, (4) RFECV with LGBM, (5) Hybrid FS with LGBM as the final model, and (6) Hybrid FS with DT as the final model. For each, we evaluate the metrics described in Section 3.5. Table 2 summarizes the results.

Table 2: Performance results for ML models with various FS methods.

Models	Train Set Accuracy	Test Set Accuracy	Acc of 10-FCV	Precision	Recall	F1-Score	Training Time (Seconds)	Testing Time (Seconds)	FPR
RFE_DT	99.98%	99.82%	99.82%	99.93%	99.87%	99.90%	2.40	0.02	0.006
RFECV_DT	99.99%	99.82%	99.81%	99.93%	99.88%	99.91%	34.38	0.36	0.006
RFE_LGBM	99.81%	99.80%	99.79%	99.88%	99.91%	99.89%	49.54	0.76	0.011
RFECV_LGBM	99.83%	99.92%	99.79%	99.88%	99.93%	99.91%	47.28	0.37	0.011
Hybrid_FS_LGBM	99.97%	99.86%	99.91%	99.89%	99.96%	99.93%	3.66	1.21	0.011
Hybrid_FS_DT	99.99%	99.80%	99.82%	99.92%	99.86%	99.89%	4.24	0.10	0.007

Source: Authors, (2026).

Acc of 10-FCV: Accuracy of 10-Fold Cross Validation, FPR: False Positive Rate, RFE_DT: RFE with DT as base estimator, RFECV_DT: RFECV with DT as base estimator, RFE_LGB: RFE with LGB as base estimator, RFECV_LGBM: RFECV with LGB as base estimator, Hybrid_FS_LGB: Hybrid feature selection LGB as final model, Hybrid_FS_DT: Hybrid feature selection DT as final model. The performance of different FS methods on ML techniques was evaluated using popular evaluation metrics, including training accuracy, testing accuracy, precision, recall, F1-score, training time, testing time, and FPR. The results obtained from experimental analysis are summarised in Table 2. Significant generalization capabilities were demonstrated by all models with impressive testing accuracies of 99.80% to 99.92%.

With a recall of 99.93% and the best testing accuracy of 99.92%, RFECV_LGBM can accurately detect real positive cases. It also shows a low rate of false alarms with an FPR of 0.0110 and a balanced F1-score of 99.91%. In contrast, models RFE_DT and RFECV_DT obtained the highest training accuracies, 99.99% and 99.98%, respectively, indicating that they correctly comprehended the training data. Model RFE_LGBM outperformed the others with a recall value of 99.91%, demonstrating its efficacy in recognizing positive cases, despite having a slightly lower training accuracy of 99.81% and a testing accuracy of 99.80%. The Hybrid_FS_LGBM and Hybrid_FS_DT models consistently performed well, with F1-scores of 99.90% and 99.89% respectively, and testing accuracy of 99.81% and 99.80%, respectively. Although Hybrid_FS_DT had the best training accuracy of 99.99% among all the models, it had a little higher FPR (0.007) than RFE_DT and RFECV_DT.

With minimal variation across metrics, all six models demonstrated outstanding predictive performance. The RFECV_LGBM model is the most suitable for use in actual smart home settings where detection reliability and accuracy are essential, since it is the most balanced in terms of testing accuracy, recall, and F1-score, and has a low FPR with a training time of 3.66 seconds. In order to validate our methodology, we have additionally evaluated our models for test accuracy using a 10-fold cross-validation process, where the data is randomly divided into 10 subsets, out of which nine subsets of the data are allocated for training and the tenth subset is reserved for testing. This process is carried out ten times until every subset has been used for testing once. Our hybrid FS methodology uses the important 12 characteristics and LGBM for model prediction, achieving a 10-fold cross-validation accuracy of 99.91%. Based on these results, the RFECV_LGBM and Hybrid_FS_LGBM models emerge as top contenders. RFECV_LGBM slightly edges in test accuracy and recall, while Hybrid_FS_LGBM is almost as good and uses a smaller feature set.

We carried out an additional 10-fold cross-validation on the training data for our Hybrid_FS_LGBM model, which yielded an average accuracy of 99.91%, confirming the model's stability across different data splits. We also performed a statistical significance test: a paired t-test comparing the F1-score of Hybrid_FS_LGBM versus a baseline LGBM using all 83 features indicated that our improvements are statistically significant ($p < 0.01$). This gives confidence that the hybrid FS is genuinely beneficial and not just fitting the idiosyncrasies of one train/test split. In practical terms, the Hybrid_FS_LGBM model offers an excellent balance of high detection rates, low false positives, and computational efficiency, using only a small subset of features. In the following, we will use Hybrid_FS_LGBM as our primary model (alongside RFECV_LGBM) for further comparisons and for generating explanations.

III.2 COMPARISON OF THE PROPOSED MODEL WITH OTHER RELATED MODELS

We benchmarked our approach against several recent IDS models from the literature, using the described metrics on either the same IoTID20 dataset or similar IoT intrusion datasets. Table 3 provides a detailed comparison of our best model with other existing state-of-the-art methods. These include ML, DL models, and hybrid frameworks: for example, by [25] who used a deep CNN on IoTID20, by [26] with autoencoders, and a Hybrid CNN-LSTM approach by [7] on an IoMT dataset.

Table 3: Comparison of our approach with other related work.

Reference	Models	Test Set Accuracy	Precision	Recall	F1-Score
[25]	DCNN	99.84%	99.67%	99.02%	99.34%
[7]	Hybrid CNN-LSTM	99.92%	99.91%	99.99%	99.95%
[27]	SA-DCNN	98.05%	95.35%	95.96%	95.64%
[17]	EE	94.14%	96.72%	97.03%	96.87%
[28]	PCC-CNN	99%	97%	95%	96%
Proposed approach	Hybrid_FS_LGBM	99.86%	99.89%	99.96%	99.93%
Proposed approach	RFECV_LGBM	99.92%	99.88%	99.93%	99.91%

Source: Authors, (2026).

SA-DCNN: Self-Attention-based Deep Convolutional Neural Network, EE: Elliptic Envelope, PCC-CNN: Pearson-Correlation Coefficient - Convolutional Neural Networks. Our Hybrid_FS_LGBM model achieves comparable or better performance than these state-of-the-art methods. Compared to the DCNN model by [25], which reported 99.84% accuracy and an F1-score of 99.34% on IoTID20, our model achieves an accuracy of 99.92%. More importantly, our recall (99.96%) is substantially higher than their 99.02%. This means our approach misses fewer attacks.

An attention-based IDS was proposed by [29] achieved ~99.9% accuracy on a large IIoT dataset. We match this performance level on our dataset, but importantly, we also provide full model interpretability, which those deep models lack. The CNN-LSTM hybrid model by [7] achieved 99.9% accuracy on their dataset. However, their approach relies on a DNN with significantly more parameters and complexity. According to [27] self-attention CNN for IIoT had significantly lower metrics (98.05% accuracy), which our model vastly outperforms. This highlights the effectiveness of focusing on feature engineering rather than solely on complex architectures. The unsupervised EE approach by [17] reached ~96% precision and recall.

Our supervised approach, unsurprisingly, does much better in a fully labelled scenario. When a selected dataset, such as IoTID20, is available, a supervised learner with FS can obtain significantly better results than unsupervised algorithms for new attack detection without labels. In summary, our proposed approach is on par with the latest research in terms of accuracy, and actually outperforms many methods in recall and precision, while using a much simpler and more interpretable model. These results validate our premise that focusing on FS and XAI can allow us to avoid the complexity of DNNs and still achieve excellent outcomes for identifying attacks in IoT networks.

III.3 MODELS' EXPLAINABILITY

One of the key benefits of our framework is that it not only produces high metrics but also provides human-understandable explanations for its decisions. In this section, we analysed the explainability of our trained models using LIME and SHAP.

The explanation process of ML models boosts confidence in their decision-making process. By eliminating the "black-box" aspect of classifiers, it ensures that the cause of the high accuracy of the classifier can be comprehended [5]. We analysed the explainability of our trained model using both SHAP and LIME to increase the credibility of the proposed ML-based framework for enhancing security in smart home IoT networks. The Tree explainer is the SHAP explanation that we used in our experimental study [9].

III.3.1 LIME Explanations

We applied LIME to generate explanations for representative instances of each class. Figure 2(a) and(b) depict the bar plots generated by LIME for the two different models, the Hybrid_FS_LGBM and Hybrid_FS_DT. These bar plots show the contributions of each feature for specific data occurrences. Orange-shaded features contribute to the "Attack" predictions, while blue-shaded features support the "Normal" class. The LIME explanation for the Hybrid_FS_LGBM model depicts that features such as Flow_Duration, Dst_Port, and Flow_Pkts/s significantly contribute to the attack classification. On the other hand, features with higher values, such as ACK_Flag_Cnt and Src_Port, contributed more to the "Normal" class but were insufficient to affect the final decision.

The LIME explanation of the Hybrid_FS_DT model shows that features such as Pkt_Len_Min, Flow_IAT_Min, and Dst_Port are the primary contributors towards the identification of attack instances. However, standard indicators such as ACK_Flag_Cnt and Bwd_Pkts/s settled for the normal class but were not sufficient to eliminate the attack indicators. These LIME explanations increase our confidence in the models. They show that the models are picking up on sensible features for attacks (e.g., long flows, particular ports, rapid packet rates, unusual packet sizes) rather than spurious correlations. Moreover, the differences highlight that the DT and LGBM models, despite similar accuracy, can base decisions on somewhat different sets of features or thresholds. Such insights could help a security analyst understand alerts better or refine the model further.

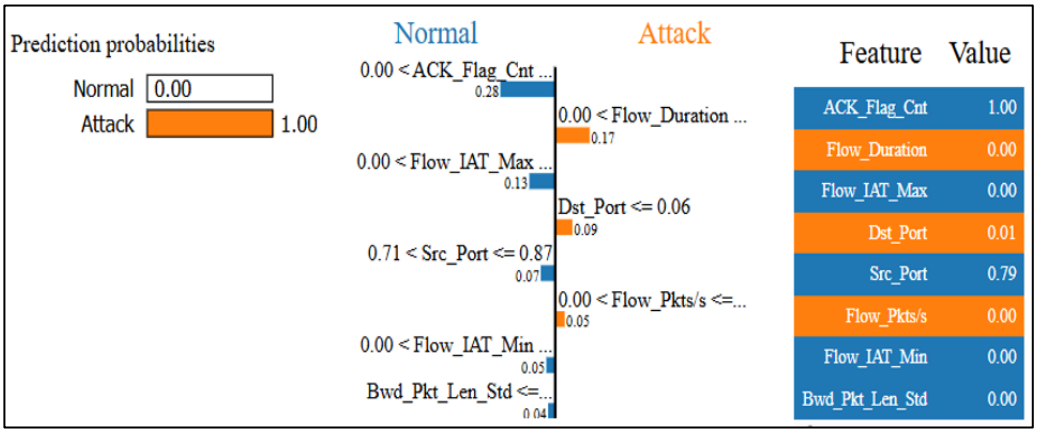

Figure 2(a): LIME explanation for the Hybrid_FS_LGBM.
Source: Authors, (2026).

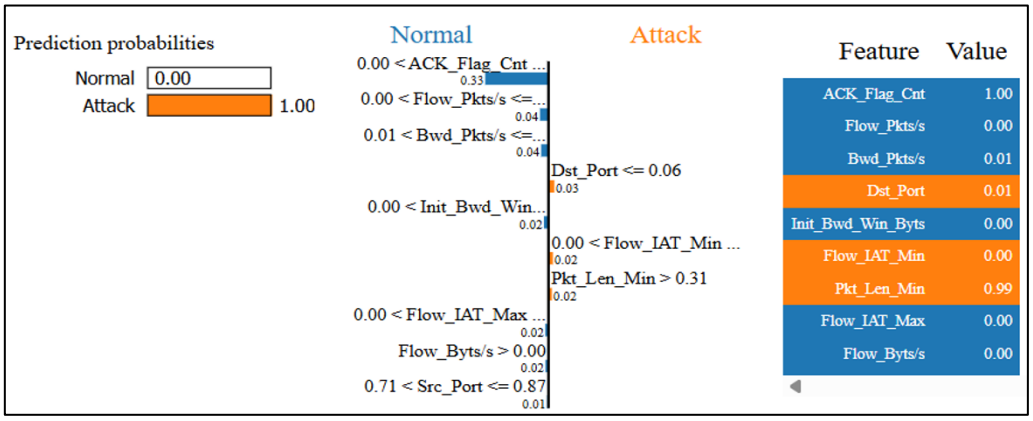

Figure 2(b): LIME explanation for the Hybrid_FS_DT.
Source: Authors, (2026).

III.3.2 Shapley Additive Explanations

While LIME is local, SHAP gives both local and global interpretability. Figure 3 presents the SHAP summary plots for the Hybrid_FS_LGBM and Hybrid_FS_DT models. In these plots, each point is a single instance's SHAP value for a feature, and points are coloured by the feature's value (red = high value, blue = low value).

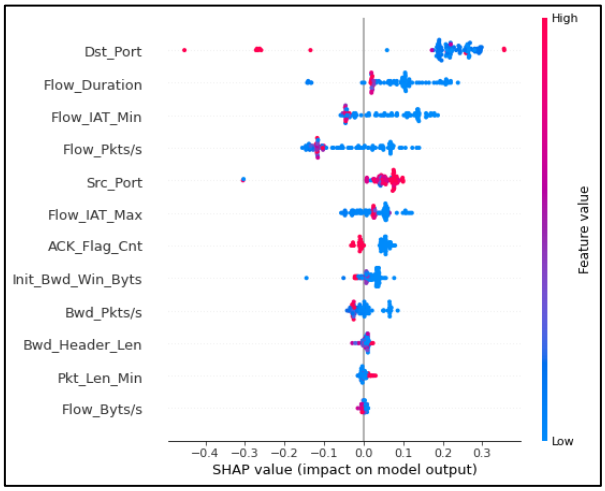

Figure 3(a): SHAP summary plot for the Hybrid_FS_LGBM.
Source: Authors, (2026).

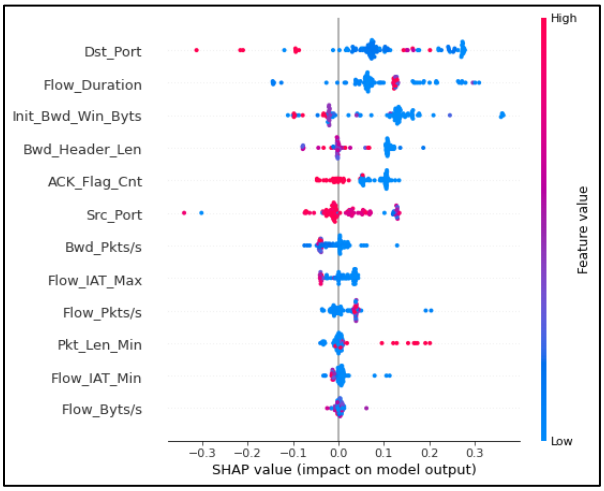

Figure 3(b): SHAP summary plot for the Hybrid_FS_DT.
Source: Authors, (2026).

SHAP is a useful method to understand the relationships between features and how they impact predictions [9]. Figure 3 (a) and (b) present the SHAP summary plots, illustrating feature distributions for the models Hybrid_FS_LGBM and Hybrid_FS_DT. In the Hybrid_FS_LGBM model, the most significant features affecting model output are Dst_Port, Flow_Duration, Flow_IAT_Min, and Flow_Pkts/s. According to the information obtained above, higher Dst_Port and Flow_Duration values significantly improve the model's ability to forecast the attack class, whereas lower Flow_IAT_Min and Flow_Pkts/s values imply an insignificant impact. However, the model Hybrid_FS_DT has a somewhat different feature attribution pattern.

Init_Bwd_Win_Byts, Bwd_Header_Length, and ACK_Flag_Cnt have become more significant in comparison to the Hybrid_FS_LGBM model, even though Flow_Duration and Dst_Port are still essential. Low values of Init_Bwd_Win_Byts and Bwd_Pkts/s are linked to attack predictions, based on the SHAP values. Thus, packet-level behaviour is more important to the DT model. Both these models adequately represent critical attack-related attributes, as can be seen from the SHAP visualizations. In summary, SHAP provides a global view that both models heavily utilize a handful of network features to make decisions. The DT and LGBM agree on many top features (port, flow duration, etc.), but DT places relatively more emphasis on certain packet-level features (window bytes, header lengths) than LGBM does. Overall, the explainability analysis confirms that our IDS behaves realistically.

III.4 EFFICIENCY AND SCALABILITY FOR DEPLOYMENT

Beyond detection performance, it is important to consider the practicality of deploying our IDS in real-world smart homes. Two major considerations are inference latency and resource usage, especially if the model will run on low-power or embedded devices (like a Raspberry Pi or a home IoT gateway).

III.4.1 Inference Latency

Our chosen algorithms are relatively lightweight and offer fast prediction times. In practice, the LGBM model can score a single network flow instance in just a few milliseconds on a CPU.

Even if we assume a modest Raspberry Pi-class CPU, the model can easily handle real-time streaming data in a home network, which typically would be on the order of tens or hundreds of flows per second without noticeable delay. Thus, the IDS can analyze packets or flows on-the-fly and raise alerts virtually instantaneously from the user's perspective.

III.4.2 Resource Footprint

Our proposed Hybrid_FS_LGBM with 100 trees and 12 features occupies only a few megabytes of memory. Since Raspberry Pi typically has a 1.2 GHz CPU and 1-2 GB of RAM, it is possible to employ a modest CPU and memory during inference. In contrast to DL models, which may need GPU support for effective operation, the proposed model uses minimal resources. Without quantization or acceleration, for instance, a CNN-LSTM IDS might not even fit or function well on a Raspberry Pi.

III.4.3 Training Complexity

Training our proposed ML-based framework (including feature selection) took a couple of hours on a standard PC. However, this process is done offline and can be performed infrequently (e.g., retraining the model when new data is available or when updating the IDS). It is simple to deploy the model to edge nodes after it has been trained. Additionally, there is very little cost on the gateway for feature extraction because we only need to compute 12 features for each incoming flow.

III.4.4 Scalability

By installing a small model at each local gateway or device, our solution can expand horizontally to handle larger networks or more devices. The model's input dimension stays the same even if more IoT devices are added because we limited the feature set.

III.4.5 Edge Deployment Considerations

Deploying edge nodes like a home router or an edge server has benefits for privacy and real-time response. Our model is well-suited for edge deployment due to its small size. Since LIME requires local models to train and SHAP requires numerous model evaluations, explainability tools like LIME and SHAP are a little too complex to execute on an edge device for every prediction. Nevertheless, those can be utilized offline or on occasion to audit the behavior of the model. In summary, our IDS is efficient and scalable enough for real-world use in smart homes. It can run on low-power hardware with minimal latency, making immediate detection and alerting feasible. We have deliberately chosen methods that ensure the solution is not just effective, but also practical.

IV. CONCLUSIONS

This study presents an anomaly-based IDS that combines a hybrid FS approach with XAI techniques, specifically tailored for IoT environments in smart homes. RFECV uses two complementary estimators to effectively reduce feature dimensionality, while still keeping the important discriminative features. This dimensionality reduction is essential for successful IoT deployments that require cost-effective models due to resource limitations. The system can provide understandable justifications for its predictions by integrating LIME and SHAP into the pipeline, shedding light on the "black-box" aspects of the ML model. We have utilized the IoTID20 smart house IoT dataset to assess our methodology. Our findings are consistent with recent research that shows that careful FS greatly improves detection performance. Moreover, the use of XAI yields detailed feature-level insights, which can improve operator trust and facilitate regulatory compliance in high-stakes smart home scenarios.

Overall, the proposed framework can able to achieve high detection rates while also providing transparency. Its robustness is demonstrated by achieving accuracy and recall metrics that are comparable to those of state-of-the-art IoTIDSs, which validates its impact on securing smart home networks. There are several promising directions to extend this research. First, we plan to investigate more sophisticated or automated FS techniques using DL algorithms. To automatically train feature representations that capture complex non-linear relationships between IoT features, for example, attention-based neural networks or autoencoder techniques could be used. Integrating these deep feature-learning strategies is likely to enhance the detection performance and robustness of complex DL models, particularly against evolving attack patterns that are difficult to capture with static feature sets.

Second, a future development might incorporate collaborative detection and FL due to the variety of data sources in smart homes. In FL, models in smart homes or multiple edge devices can be trained collectively without sharing raw data. Third, edge computing integration is becoming a natural progression for deploying IDS widely. Since our model is lightweight, it can run on edge devices. Future studies might examine a complete edge-based architecture in which an Internet of Things gateway detects intrusions in real time and perhaps works with other edge nodes. Managing distributed attacks and exchanging alert intelligence amongst edge network devices are part of this. In summary, our study demonstrated that an IoT IDS is efficient, effective, and transparent when integrating FS with XAI.

V. AUTHOR'S CONTRIBUTION

Conceptualization: Pritimayee Satapathy and Prafulla Kumar Behera.
Methodology: Prafulla Kumar Behera.
Investigation: Pritimayee Satapathy and Prafulla Kumar Behera.
Discussion of results: Pritimayee Satapathy and Prafulla Kumar Behera.
Writing - Original Draft: Pritimayee Satapathy.
Writing - Review and Editing: Prafulla Kumar Behera.
Resources: Pritimayee Satapathy and Prafulla Kumar Behera.
Supervision: Pritimayee Satapathy.
Approval of the final text: Pritimayee Satapathy and Prafulla Kumar Behera.

VI. REFERENCES

[1] A. G. Ayad, N. A. Sakr, and N. A. Hikal, "A hybrid approach for efficient feature selection in anomaly intrusion detection for IoT networks," The Journal of Supercomputing, vol. 80, no. 19, pp. 26942-26984, 2024.

[2] Y. Majib, M. Alosaimi, A. Asaturyan, and C. Perera, "Dataset for cyber–physical anomaly detection in smart homes," Frontiers in the Internet of Things, vol. 2, p. 1275080, 2023.

[3] A. Alabbadi and F. Bajaber, "An Intrusion Detection System over the IoT Data Streams Using eXplainable Artificial Intelligence (XAI)," Sensors (Basel, Switzerland), vol. 25, no. 3, p. 847, 2025.

[4] P. Waghmode, M. Kanumuri, H. El-Ocla, and T. Boyle, "Intrusion detection system based on machine learning using least square support vector machine," Scientific Reports, vol. 15, no. 1, p. 12066, 2025.

[5] M. M. Alani, "An explainable efficient flow-based Industrial IoT intrusion detection system," Computers and Electrical Engineering, vol. 108, p. 108732, 2023.

[6] B. Jayan, T. Ganesan, and B. B. Kurup, "Enhancing IoT network security through advanced data preprocessing and hybrid firefly-salp swarm optimized deep CNN-based intrusion detection," ITEGAM-JETIA, vol. 10, no. 47, pp. 73-82, 2024.

[7] A. Berguiga, A. Harchay, and A. Massaoudi, "HIDS-IoMT: A deep Learning-Based intelligent intrusion detection system for the internet of medical things," IEEE Access, 2025.

[8] C. E. Asry, I. Benchaji, S. Douzi, and B. E. Ouahidi, "Enhancing cybersecurity: A high-performance intrusion detection approach through boosting minority class recognition," PloS one, vol. 20, no. 3, p. e0317346, 2025.
[9] T. B. Ogunseyi and G. Thiyagarajan, "An Explainable LSTM-Based Intrusion Detection System Optimized by Firefly Algorithm for IoT Networks," Sensors, vol. 25, no. 7, p. 2288, 2025.

[10] G. Lazrek, K. Chetioui, Y. Balboul, and S. Mazer, "An RFE/Ridge-ML/DL based anomaly intrusion detection approach for securing IoMT systems," Results in Engineering, vol. 23, p. 102659, 2024.

[11] A. S. Ahanger, S. M. Khan, F. Masoodi, and A. O. Salau, "Advanced intrusion detection in internet of things using graph attention networks," Scientific Reports, vol. 15, no. 1, p. 9831, 2025.

[12] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," Machine learning, vol. 46, no. 1, pp. 389-422, 2002.

[13] J. R. Quinlan, "Induction of decision trees," Machine learning, vol. 1, no. 1, pp. 81-106, 1986.

[14] G. Ke et al., "Lightgbm: A highly efficient gradient boosting decision tree," Advances in neural information processing systems, vol. 30, 2017.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, "" Why should I trust you?" Explaining the predictions of any classifier," in Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135-1144.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," Advances in neural information processing systems, vol. 30, 2017.

[17] J. I. Iturbe-Araya and H. Rifà-Pous, "Enhancing unsupervised anomaly-based cyberattacks detection in smart homes through hyperparameter optimization," International Journal of Information Security, vol. 24, no. 1, p. 45, 2025.

[18] H. Kang, D. H. Ahn, G. M. Lee, J. Yoo, K. H. Park, and H. K. Kim, "IoT network intrusion dataset," IEEE Dataport, vol. 10, pp. q70p-q449, 2019.

[19] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in IoT networks," in Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33, 2020: Springer, pp. 508-520.

[20] W. McKinney, "Data structures for statistical computing in Python," scipy, vol. 445, no. 1, pp. 51-56, 2010.

[21] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, no. 7825, pp. 357-362, 2020.

[22] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, vol. 16, pp. 321-357, 2002.

[23] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," The Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[24] M. Saied, S. Guirguis, and M. Madbouly, "A comparative study of using boosting-based machine learning algorithms for IoT network intrusion detection," International Journal of Computational Intelligence Systems, vol. 16, no. 1, p. 177, 2023.

[25] S. Ullah et al., "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, 2022.

[26] Y. Song, S. Hyun, and Y.-G. Cheong, "Analysis of autoencoders for network intrusion detection," Sensors, vol. 21, no. 13, p. 4294, 2021.

[27] M. S. Alshehri, O. Saidani, F. S. Alrayes, S. F. Abbasi, and J. Ahmad, "A self-attention-based deep convolutional neural network for IIoT networks intrusion detection," IEEE Access, vol. 12, pp. 45762-45772, 2024.

[28] M. Bhavsar, K. Roy, J. Kelly, and O. Olusola, "Anomaly-based intrusion detection system for IoT application," Discover Internet of Things, vol. 3, no. 1, p. 5, 2023.

[29] K. Yang, J. Wang, and M. Li, "An improved intrusion detection method for IIoT using attention mechanisms, BiGRU, and Inception-CNN," Scientific Reports, vol. 14, no. 1, p. 19339, 2024.