Advanced Machine Learning Pipeline Utilizing Generative and Explainable Artificial Intelligence for Reliable Intrusion Detection in Internet of Things.

Chandrani Mukherjee [1]

[1]Mphasis AI, 7239 Riverside Station Blvd. Secaucus NJ

ABSTRACT

Recent research on intrusion detection systems (IDSs) in the internet of things (IoT) has explored various models, including deep neural networks, classical classifiers, explainable artificial intelligence (XAI), and dimensionality reduction methods such as principal component analysis (PCA). However, few studies offer a comprehensive AI pipeline that systematically integrate data preprocessing, class imbalance handling (e.g., using the synthetic minority oversampling technique (SMOTE)), advanced feature engineering (e.g., PCA and linear discriminant analysis), multimodel selection paradigms, and modern XAI techniques. This study fills that gap by proposing a unified IDS framework that integrates these elements and introduces generative AI and large language models (LLMs), such as Gemini, to automate dynamic feature extraction from unstructured network logs. Data visualization tools like tdistributed stochastic neighbor embedding (t-SNE) and Shapley additive explanation (SHAP) are employed to analyze feature distributions before and after dimensionality reduction. Experimental results confirm that the SMOTE significantly improves model accuracy, whereas dimensionality reduction has limited effect on model performance. Among evaluated classifiers, XGBoost achieves the highest accuracy (99.99%). For explainability, TreeSHAP is preferred due to its computational efficiency, and t-SNE visualizations based on SHAP values reveal distinct clusters of benign and malicious network traffic. This integration of data processing, automated feature extraction using LLMs, model selection, and interpretable machine learning offers a novel approach to IoT security. In addition to advancing IDS methodology via robust and transparent decision-making, this study exemplifies the potential of integrating automated data engineering and XAI in cyber–physical system research..

Keywords: generative artificial intelligence, internet of things, intrusion detection system, local interpretable modelagnostic explanations, principal component analysis, Shapley additive explanation, transmission control protocol

How to Cite: Chandrani Mukherjee , (2026) Advanced Machine Learning Pipeline Utilizing Generative and Explainable Artificial Intelligence for Reliable Intrusion Detection in Internet of Things... Journal of Carcinogenesis, Vol.25, No.1, 67-88

1. INTRODUCTION

The internet of things (IoT) represents a profound advancement in modern connectivity; it unifies a multitude of autonomous devices—from sensors and smart locks to surveillance cameras—within a dynamic digital ecosystem. The rapid adoption of IoT technologies, particularly during the COVID-19 pandemic, has led to the widespread deployment of IoT-enabled devices that minimize direct human contact in tasks such as door access direct human involvement in opening doors. However, it has also expanded the attack surface for cyber threats. Despite its evident societal and industrial benefits, the heterogeneous and resource-constrained nature of IoT devices leaves them highly vulnerable to an evolving array of cyberattacks, including high-profile malware threats such as Mirai, Troii, and Hakai botnets, which threaten network reliability, data security, and operational continuity. Current research primarily employs conventional machine learning (ML) models, deep neural networks, and explainable artificial intelligence (XAI) to improve IoT intrusion detection systems (IDSs). However, these methods often face challenges related to interpretability, scalability, and high-dimensional network traffic data. To address these gaps, this study introduces two novel innovations: (1) the application of t-distributed stochastic neighbor embedding (tSNE) for the intuitive visualization and analysis of high-dimensional IoT network packet data, enabling the identification of subtle patterns linked to various forms of cyberattacks, and (2) the integration of large language models (LLMs) to autonomously extract and contextualize critical features from raw network logs, simplifying feature engineering and improving model robustness.

This article presents an end-to-end IDS pipeline that integrates advanced ML classifiers optimized via rigorous data balancing and hyperparameter tuning. We thoroughly assessed classifier decisions by comparing a suite of XAI methods. The incorporation of LLMs for feature extraction, coupled with the visualization power of t-SNE, represents a pioneering approach for enhancing the transparency and performance of IDSs in IoT ecosystems. This article initiates with an extensive literature review, identifies research gaps, outlines the experimental methodology, and concludes by presenting comprehensive results and key insights for developing next-generation, interpretable IDS solutions tailored to IoT environments (Figure 1).

Figure 1: IoT ecosystem and security intrusion overview.

2. RELATED WORK

2.1. XAI

The opacity of modern ML, particularly deep learning, has prompted research on interpretability. Doshi-Velez and Kim [13] emphasized foundational approaches toward interpretable AI and the inherent accuracy–interpretability trade-off. Adadi and Berrada [1] systematically classified XAI methods into ante-hoc and post-hoc methods. Popular post-hoc techniques, such as local interpretable model-agnostic explanations (LIME) [20] provide localized, modelagnostic explanations. Guidotti et al. [18] categorized black-box model explanation strategies relevant to the finance and healthcare domains. Explainability is crucial for user trust, as reported by Amarasinghe and Manic [6] in their research on neural-network-based IDSs, highlighting the need for transparency in critical systems.

2.2. IDSs in IoT Environments

IoT proliferation has increased the demand for efficient IDSs. Aung and Min [7] demonstrated that K-means clustering effectively identifies network intrusions via unsupervised learning. Alrashdi et al. [4] presented AD-IoT, a supervised system achieving high accuracy with low false positives in detecting network attacks on real IoT data. In smart environments, ensemble methods are often preferred over individual models [3]. Integrating SDN further enhances model scalability and performance, as exemplified by hierarchical approaches [5]. In addition, edgecentric lightweight IDSs, such as Passban [14], have achieved resource-efficient anomaly detection.

2.3. ML Techniques for Security

ML-driven security frameworks integrate anomaly detection, and data integrity [8]. For intrusion detection, XGBoost excels due to its gradient boosting and regularization abilities [12]. Further, deep learning models effectively capture temporal patterns in IoT traffic [16]. To handle traffic variability and class imbalance during intrusion detection, researchers have employed unsupervised and dimensionality reduction techniques such as principal component analysis (PCA) [10] and the synthetic minority oversampling technique (SMOTE) [17].

2.4. Privacy, Compliance, and Legal Considerations

The intensifying scrutiny of regulations, exemplified by the General Data Protection Regulation (GDPR), necessitates robust data access and transparency mechanisms. Alizadeh et al. [2] identified gaps between regulatory expectations and current technical capabilities. Hausken and Mohr [19] applied the game theory to elucidate information sharing and utility, achieving fair and accountable AI system design.

2.5. Challenges and Research Gaps

Despite advancements in IoT IDSs, key challenges persist. These include the absence of standardized benchmarks, trade-offs between model complexity and explainability, and limited generalizability across heterogeneous IoT contexts [1,13,14]. Moreover, inherent security vulnerabilities of ML algorithms necessitate the development of models robust against adversaries [9]. Notably, gaps in data preprocessing and feature engineering methods for IoT IDSs remain insufficiently addressed in current literature.

2.5.1. Imbalanced Data Handling

Advanced balancing techniques like the SMOTE and hybrid methods remain underutilized, despite their efficacy in mitigating skewed class distributions.

2.5.2. Algorithmic Sampling

Research exploring sampling strategies for class balancing informed by algorithmic or data-driven insights remains limited.

2.5.3. Dimensionality Reduction

PCA and linear discriminant analysis (LDA) predominate while nonlinear dimensionality reduction methods like t-SNE are seldom employed, despite t-SNE demonstrating advantages in terms of performance.

2.5.4. Integrated Methodologies

Existing literature lacks a systematic analysis that jointly considers class balancing, feature selection, and dimensionality reduction.

2.5.5. Challenges in Intrusion Detection in IoT Environments

Diversity and Scale of Threats: The proliferation of IoT devices has driven an increase in diverse attack types (e.g., distributed denial of service, MITM, spoofing, replay attacks), necessitating robust, real-time intrusion classification.

2.5.6. Resource Constraints

Owing to stringent resource limitations, IoT applications require computationally lightweight IDS solutions with minimal latency.

2.5.7. Classification Models

Conventional ML models (decision trees, Naïve Bayes models, and support vector machines (SVMs)) are prevalent; however, their evaluation lacks uniformity across studies.

2.5.8. Deep Learning Techniques

Deep learning architectures (CNNs, RNNs, and LSTM models) exhibit high performance, with activation and output functions (e.g., ReLU and softmax) demonstrating effectiveness for multiclass detection tasks.

2.5.9. Clustering-Based IDS Approaches

Unsupervised Anomaly Detection: Clustering algorithms, including LOF, INLOF, COF, BIRCH, and agglomerative methods, are increasingly utilized for detecting anomalies in unlabeled data.

2.5.10. Identified Limitations

Few studies have reported the comparative assessments of clustering techniques or have employed visual interpretability tools like scatter plots.

2.5.11. Explainability and Interpretability

Regulatory and Practical Imperatives: Owing to regulatory frameworks (e.g., the GDPR) and the demand for explainable predictions, transparent AI models are imperative.

2.5.12. Interpretability Taxonomy

Intrinsic methods (e.g., decision trees and linear regression models) are interpretable by design. Post-hoc techniques (e.g., LIME and the permutation feature importance technique) provide model-agnostic explanations. Global and local explanations address the entire model or individual predictions, respectively.

2.5.13. Real-Time Constraints

The feasibility of IDSs like SNORT, BRO, and SURICATA on low-power edge devices is currently being explored, with SNORT demonstrating superior efficiency on platforms like Raspberry Pi.

2.5.14. Hyperparameter Optimization

Deep learning approaches frequently overlook the systematic tuning of model parameters and network architectures.

2.5.15. Evaluation Metrics

Performance metrics adapted for evaluating imbalanced data (e.g., F-score, recall, kappa, and K–S statistics) are insufficiently reported.

2.5.16. Ensemble Methods

Ensemble classifiers such as AdaBoost and XGBoost garner less attention than linear approaches.

2.5.17. LLMs in Log Preprocessing

No current studies on IDSs used in IoT environments have integrated LLMs (e.g., Gemini Flash 2.0) for feature extraction from system logs.

3. RESEARCH METHODOLOGY

Research methodologies encompass the systematic design and execution of investigations, including explicit procedures for data collection, analysis, and interpretation. The three key components of every investigation are data collection, data analysis, and problem modeling.

Research is commonly classified based on the nature of used data (qualitative or quantitative), nature of data sources (primary or secondary), degree of variable manipulation (descriptive or experimental), and objectives (applied or fundamental). Qualitative research focuses on patterns and interpretations, and quantitative research relies on statistical analysis and hypothesis testing. While primary research generates data firsthand, secondary research utilizes pre-existing datasets. Descriptive research investigates phenomena without manipulating variables; in contrast, experimental research involves deliberate intervention. Finally, applied research addresses specific practical problems, and fundamental research focuses on advancing theoretical understanding.

Utilizing the IoT-23 dataset, this study developed a prediction model for intrusion detection in IoT. Hence, by definition, this research is secondary, applied, and quantitative. The investigation focused on model explainability, compared LIME and Shapley additive explanation (SHAP) algorithms for interpretability analysis, and adopted descriptive research strategies. This investigation further involved the analysis of various feature engineering methods and their effect on model prediction performance. The methodology involved a systematic evaluation and statistical comparison of various techniques, ranging from SMOTE sampling to SHAP and LIME explainability methods, as well as feature extraction from network logs. Furthermore, this study incorporated LLMs for feature extraction from network logs.

Subsequent sections elucidate the proposed research methodology and its underlying rationale.

3.1. Research Approach Overview

The main contributions of this study are as follows:

Identification of the optimal workflow for IDSs in IoT environments.

Investigation, selection, and comparison of the most effective class-balancing techniques, including the SMOTE as well as undersampling, oversampling, and hybrid sampling approaches.

Comparative analysis of diverse dimensionality reduction and feature selection methods, specifically filterbased, wrapper-based, and t-SNE techniques.

Evaluation and comparative analysis of various ML algorithms using multiple performance metrics to identify the most effective model.

Explanation of model predictions using intrinsic and model-agnostic techniques like LIME and SHAP.

Comparison of the computational time requirements of these explanation methods.

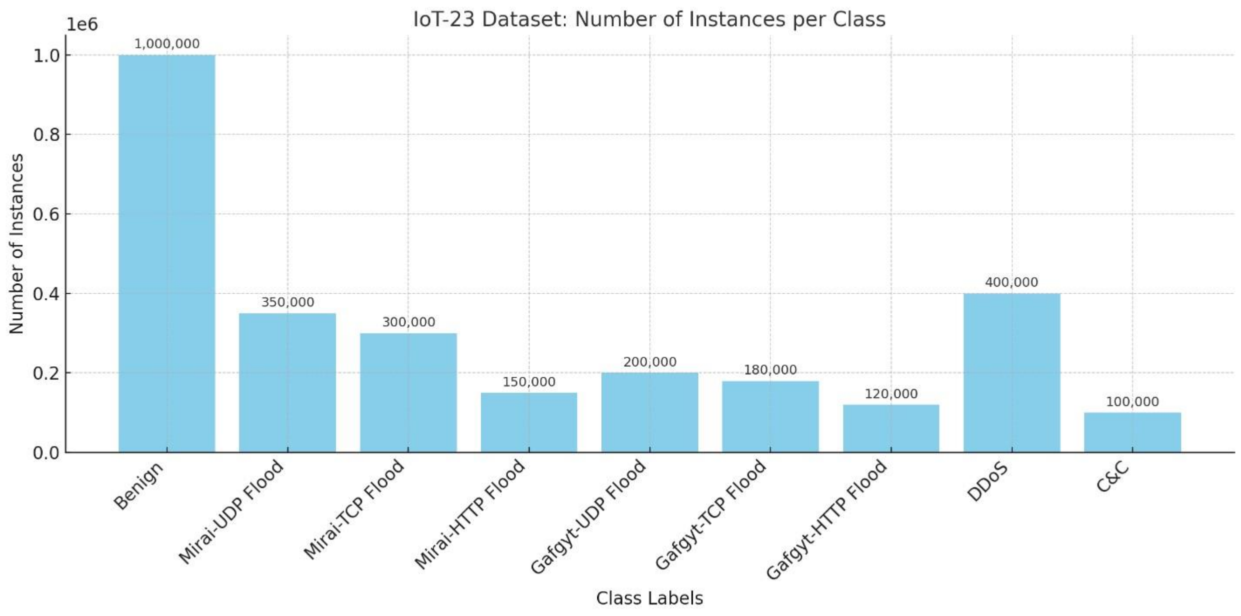


Figure 2: Class of attacks vs. total numbers of network logs.

Evaluation and utilization of an LLM to identify feature values from a simulated network log.

3.2. Data Analysis Approach

This research employed the IoT-23 dataset [15], and its specifications are presented in Table 1. Figure. 2 shows target labels vs. number of network flows recorded.

Table 1: Dataset Specification

Metric	Value
Total Number of Data Files	23
Number of Malicious Captures	20
Number of Benign Captures	3
Total Labeled Rows	325 million
Timestamp of Data	2018–2019
Target Variable	Label

The dataset, comprising 21 features with both binary and multiclass targets, underwent extensive preprocessing informed by exploratory data analysis (EDA). The process included normalization, class balancing, encoding, and data quality checks such as handling missing values, removing duplicates, detecting outliers, and addressing multicollinearity using variance inflation factor.

Model validation employed data splitting and K-fold cross-validation to ensure unbiased performance assessment. Feature engineering involved appropriate encoding, transformation, and scaling techniques. Class imbalance was systematically addressed using resampling methods (e.g., SMOTE), hybrid strategies, and algorithm-level solutions such as boosting. To improve interpretability and computational efficiency, dimensionality reduction was performed using filter, wrapper, and projection-based approaches (e.g., PCA, recursive feature elimination).

3.3. Proposed Modeling Methods

Model building, leveraging historical and current data to forecast future outcomes, is crucial in artificial intelligence. Although ML encompasses supervised, unsupervised, and reinforcement learning paradigms, this study focused on supervised learning, wherein models are trained on labeled input–output pairs to identify data relationships and enhance predictive accuracy. Specifically, this research examined binary classification, distinguishing between “normal” and “default” states of IoT devices for analytical clarity.

The evaluated classification algorithms included logistic regression models, classification and regression trees (CARTs), K-nearest neighbor (KNN) models, Naïve Bayes modes, SVMs, and ensemble approaches like random forest models, bagging models, AdaBoost, and XGBoost. Logistic regression models utilize the sigmoid function to model probabilities, CARTs construct hierarchical decision trees, KNN models conduct classification based on proximity to neighboring data points of network log, Naïve Bayes models obtain probabilistic inferences, and SVMs separate classes based on optimal hyperplanes. Ensemble methods aggregate multiple base classifiers to improve predictive performance and address the bias–variance trade-off.

Hyperparameter optimization—which involves the careful tuning of parameters, such as the regularization strength in logistic regression models, kernel functions and gamma values in SVMs, K values in KNN models, and the tree depth or estimator count in ensemble methods—is essential for model efficacy.

Enhancing ML model interpretability is a crucial research domain, with explanation techniques classified as intrinsic or model-agnostic techniques. Model-agnostic techniques, such as LIME and SHAP, are well known due to their broad applicability. LIME explains model predictions locally by fitting interpretable surrogate models via sampling and weighting. Meanwhile, SHAP employs Shapley values from the cooperative game theory to identify feature contributions using a principled approach. Although both techniques target instance-level explanations, LIME emphasizes local perturbations, whereas SHAP provides coalition-based attributions. This study compared the computational efficiency of both techniques.

Visualization is crucial for interpreting explanations. Principal visualizations include the following: (1) LIME feature plot, which presents feature impacts per instance; (2) LIME explanation heatmap, which aggregates feature importance across instances; (3) SHAP dependency plot, which reveals feature interaction effects; (4) SHAP force plot, which decomposes individual predictions into additive contributions; and (5) SHAP summary plot, which ranks global feature importance and distributions. These visualizations collectively promote model transparency by clarifying feature contributions to the output. Table 2 summarizes the distinguishing characteristics of LIME and SHAP.

Table 2: LIME vs. SHAP

LIME	SHAP
Local explanations	Local and global explanations
Fast, linear surrogate models	Slower, modelagnostic explanations
Measures prediction alterations after feature removal	Quantifies feature contributions to predict output differences

3.4. LLMs for Feature-Value Extraction

• The use of Gemini Flash or any other LLM to extract feature values from network logs provided the proposed IDS with a dynamic approach for detecting frauds.

3.5. SUMMARY

This study developed a novel IDS and dedicated workflow optimized for IoT environments. The workflow integrated multiple classification methodologies, encompassing deep learning models, decision trees, clustering algorithms, and ensemble techniques. Subsequently, XAI techniques (specifically SHAP and LIME) were adopted to enhance model interpretability. The study incorporated LLMs for feature extraction from network logs. The anticipated outcomes are summarized in line with the study's objectives, providing a clear framework for future investigations.

4. . ANALYSIS AND DESIGN

This section presents the robust workflow of the proposed IDS, as illustrated in Figure 3. The dataset and code used in this research are presented in [21]. Here, only the adopted models for the IDS are described, including the preprocessing techniques employed to prepare the data for the models. Data sampling techniques, along with feature extraction processes and inferences, are presented. Model prediction results are analyzed and compared via LIME and SHAP and different explanation plots are discussed. Furthermore, we demonstrate how to extract feature values to draw inferences from the models using unstructured network logs.

4.1. Data Description

The IoT-23 dataset was used and subsampled to accelerate execution, with performance measured in terms of the time taken to run the IDS pipeline. Table 3 describes the subdataset. The dataset contains 23 columns and 1774329 instances of benign and malicious data. The individual column descriptions are presented in Supplementary Table S1.

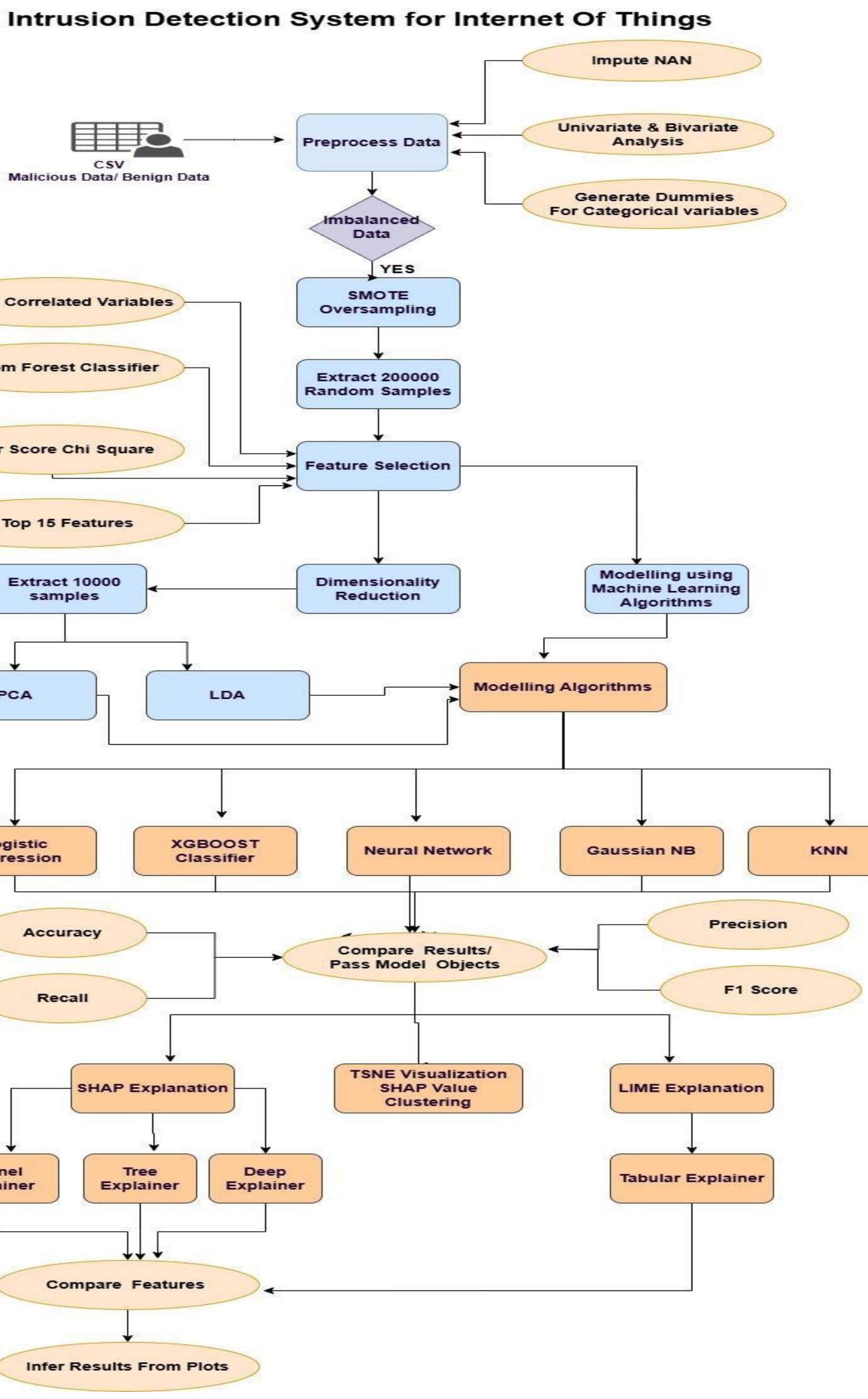


Figure 3: Flow diagram of the IDS

4.2. Data Cleaning

Rigorous data cleaning is essential for reliable predictions. Hence, the following sequential procedures were employed with minimal data loss:

Removal of missing values: Sparse NaNs in key features were mean-imputed; entirely null columns (*local orig*, *local resp*, and *tunnel parents*) were discarded.

Removal of duplicates: None were detected due to unique identifiers (UIDs).

Feature Engineering: Replaced *Timestamp* with the extracted hour to capture temporal effects.

Irrelevant Features: Discarded noninformative fields (e.g., *UIDs*).

Text Preprocessing: Eliminated hyphens for numeric conversion.

Type Conversion: Encoded IP addresses as binary values.

Categorical Data: Performed one-hot encoding for *conn state*, *history*, *proto*, and *service*.

4.3. EDA

Comprehensive EDA-guided feature selection provided insights into variable distributions, relationships, and multicollinearity.

4.3.1. Univariate Analysis

Univariate analysis was adopted to summarize each feature using descriptive statistics (minimum, maximum, quartiles, median, and mean). Consequently, significant class imbalance was identified, with 1,434,245 malicious and 338,124 benign samples (Figure 4), necessitating minority-class oversampling.

Table 3: Dataset Characteristics

Category	Instances	Attributes	Proportion (%)
Benign Hue	453	23	0.021
Benign Echo	1,375	23	0.066
Benign Door Lock	131	23	0.0063
Mirai (Malicious)	1,048,575	23	50.66
Hakai (Malicious)	10,404	23	0.50
Hide and Seek (Mal.)	713,391	23	40.20
Total	1,774,329	23	100.00

4.3.2. Categorical and Continuous Variable Distribution

The analysis of categorical variables revealed that *id.orig h* and *id.orig p* were dominated by two values each, indicating that most attacks originated from two primary IP addresses and ports. The variable *id.resp h* was distributed across numerous IP addresses, reflecting a wide range of affected devices, while *id.resp p* clustered around four response ports. TCP was the predominant protocol while the HTTP was the most affected protocol, followed by the DNS. S0 was the prevailing connection state, implying connection attempts without response. In addition, request histories were primarily SYN packets without the ACK bit.

For continuous variables, request durations were concentrated between 10 and 20 s. Both *orig bytes* and *resp bytes* were mostly within the 45–50 byte range. As indicated by NaN values for *local orig* and *local resp*, all connections were remote. The original IP header bytes mostly ranged between 40 and 45 bytes, response bytes were in the range of 0–40, and response header bytes lied between 40 and 45 bytes. Furthermore, tunneling was not observed (tunnel was consistently NaN). These distributions are illustrated in Figure 5.

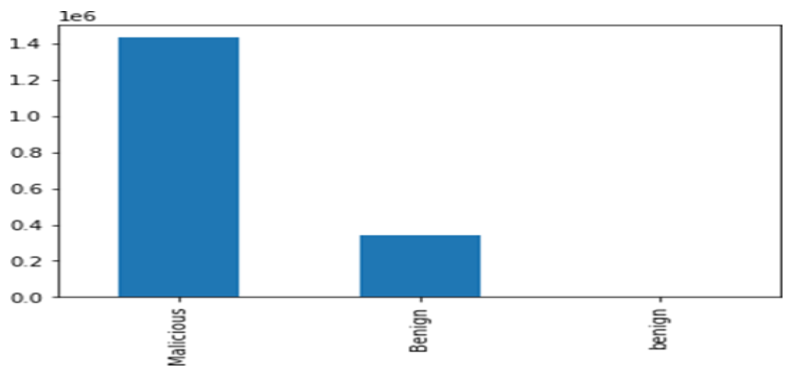


Figure 4: Target variables in univariate analysis

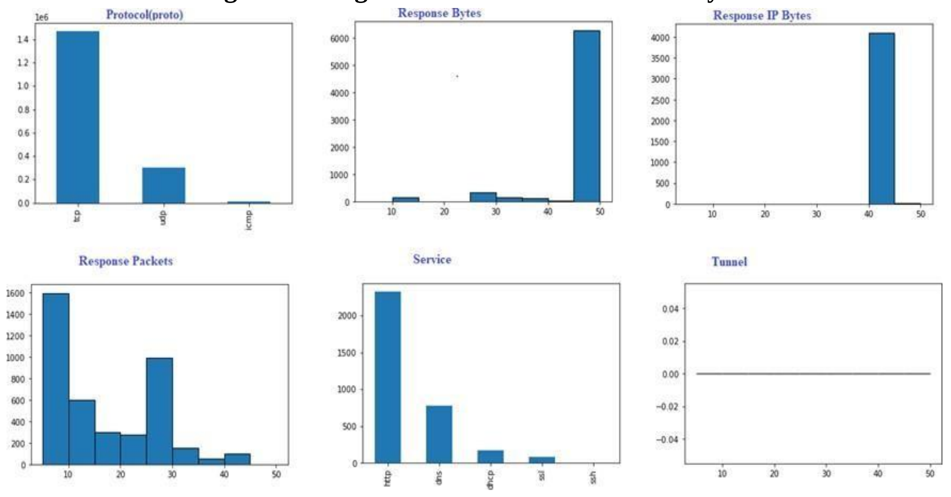


Figure 5: Univariate analysis of features

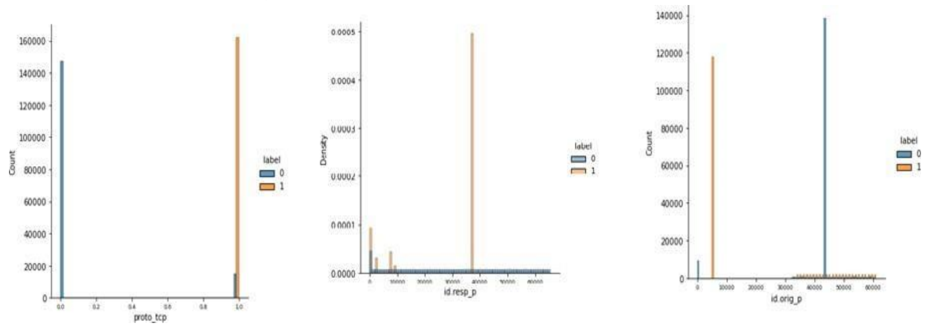


Figure 6: Bivariate analysis results

4.3.3. Bivariate Analysis

Performing bivariate analysis while investigating the relationship between the independent and target variables is crucial which is important to know which variables are likely to influence the target variable. Bivariate analysis was performed on unprocessed and processed data after oversampling. It was also performed after preprocessing to check the distribution of independent variables with the target variable. Findings were then compared with those obtained from model explanation techniques and their output features for a particular classification. Figure 6 shows the results of a few bivariate analyses.

4.3.4. Multivariate Analysis

Multivariate analysis reveals the relationship between feature variables. Figure 7 illustrates the relationship between all independent variables and the target variable.

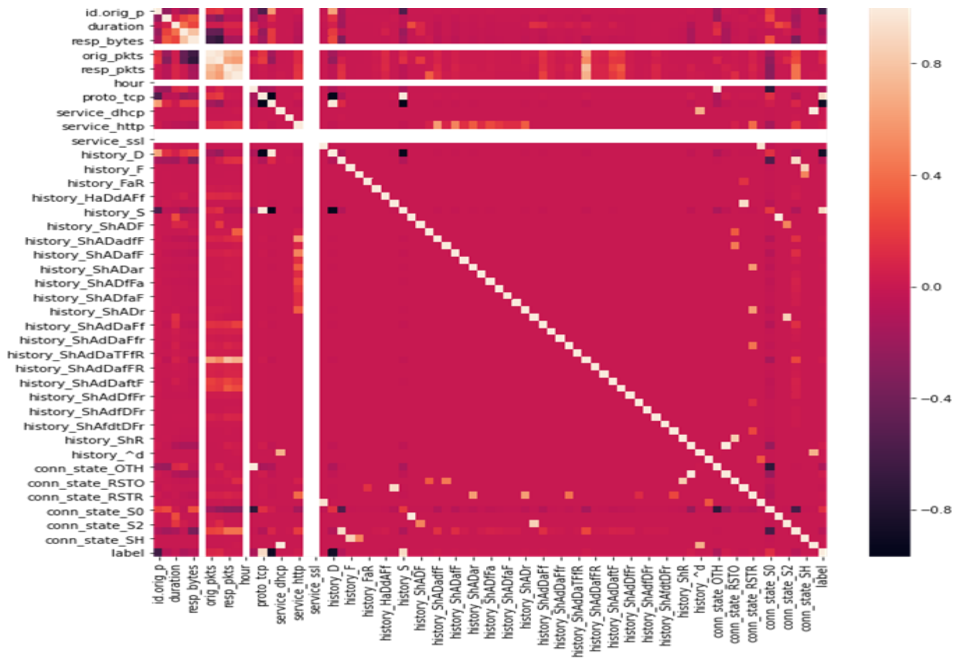


Figure 7: Heatmap obtained from multivariate analysis

The analysis of the heatmap revealed that the target variable "label" was highly correlated to *history S*, *proto tcp*, *history ShR*, *orig pkts*, and *resp pkts* as well as to *duration* to some extent. The label was negatively correlated to a few independent variables (*id.orig p* and *history D*), and 15 of such correlations were discarded before dimensionality reduction or modeling. After discarding these correlated features, only 61 features remained.

4.4. Data Transformation

Categorical variables (*proto*, *service*, *history*, and *conn state*) were converted into the numerical form via one-hot encoding using the pandas.get_dummies function, increasing feature dimensionality from 23 to 76. The target variable (label) was encoded using a label encoder. In addition, the dataset was partitioned into training and testing subsets with a 70/30 split using train_test_split() inScikit-learn.

4.4.1. Class Balancing and Feature Extraction

Owing to class imbalance (malicious ¿ benign), the SMOTE was employed to oversample the minority class, constituting the first adoption in the IoT IDS for data balancing. After SMOTE implementation, both classes contained 161,727 instances. Malicious and benign classes were labeled as "1" and "0," respectively.

4.4.2. Feature Selection

Feature selection was performed to mitigate overfitting and improve model accuracy. Four feature selection methods were applied to the oversampled data, as presented in Supplementary Table S2. After the successful elimination of correlated variables and feature extraction, the extracted features were combined and integrated for modeling.

4.4.3. Feature Scaling and Dimensionality Reduction

To ensure comparability across features, numerical variables were standardized utilizing StandardScaler from Scikit-learn. Dimensionality reduction was achieved using PCA and LDA. For PCA, the top eight components were retained. Meanwhile, LDA—constrained by the binary class structure—yielded a single component. Preliminary results indicated that LDA with a logistic regression model achieved high classification performance. Owing to computational constraints, analyses were conducted on a stratified random sample of 10,000 instances. Additional classifiers, including KNNs, neural networks, GaussianNB, and XGBoost, were evaluated to assess the impact of dimensionality reduction on classification accuracy. Table 4 presents detailed results.

Table 4: Dimensionality Reduction Techniques

Dimensionality Reduction Method	Model	Accuracy	Precision	I Recall	F1-Score	Support

PCA	Logistic regression	98%	0:0.99, 1:0.97	0:0.97, 1:0.99	0:0.98, 1:0.98	0:1519, 1:1481
LDA	Logistic regression	98%	0:1.00, 1:0.96	0:0.96, 1:1.00	0:0.98, 1:0.98	0:1519, 1:1481

4.5. ML Model Implementation

4.5.1. Selection of the Sampling Technique

The SMOTE was employed for data oversampling because the minority classes required augmentation. After sampling, 200,000 instances were extracted for model inputs.

4.5.2. Evaluating and Selecting the Best Feature Extraction Technique

Feature selection: Features extracted via the four feature selection methods (correlation-based method, random forest classifier, gradient boosting classifier, and Fisher-score-based method) were combined as a union of all features selected after discarding correlated variables. The resultant features were as follows. The analysis utilized a subset of sixteen key features: *proto tcp*, *id.orig p*, *id.resp p*, *resp bytes*, *duration*, *orig bytes*, *proto icmp*, *orig pkts*, *conn state S0*, *history Dd*, *resp pkts*, *history Sr*, *orig ip bytes*, *service http*, *history ShAdDafF*, and *history ShADadfF*. These features were selected due to their relevance to network behavior characterization and attack detection, while ensuring that dimensionality remained tractable for one-column presentation.

4.5.3. Dimensionality Reduction

Dimensionality reduction was performed after feature extraction. Two process flows were considered. In one of the two flows, the end-to-end ML modeling process was conducted. In the other, feature selection and dimensionality reduction were omitted. PCA and LDA components were determined, and their performances were compared. Although LDA performed slightly better than PCA, we considered both for modeling.

4.5.4. Data Distribution Visualization

After dimensionality reduction, data distribution was visualized using t-SNE. Figure 8 shows that more than four clusters remained in the reduced data, which was due to more categories of malicious data present in the combined dataset.

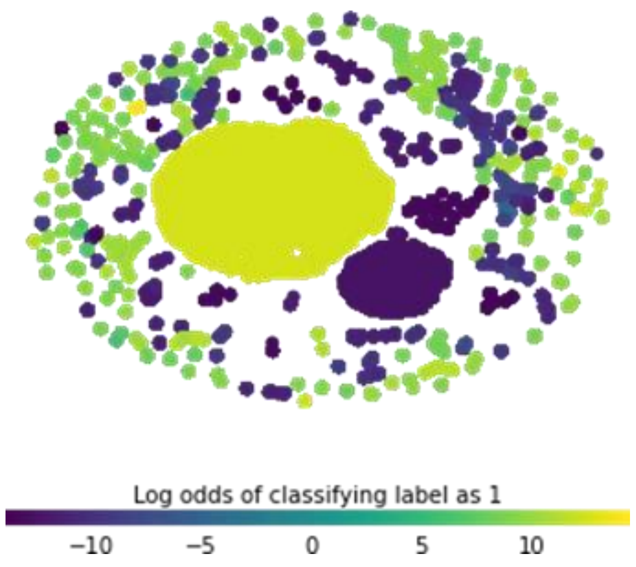


Figure 8: t-SNE visualization of LDA-reduced training data

4.5.5. Classifier Training, Tuning, and Evaluation Setup

This section outlines the model training procedures and adopted algorithms. Two process flows are compared: (1) sampling, feature selection, dimensionality reduction, and model tuning (a novel contribution of this study) and (2) direct modeling on sampled data, omitting feature selection and dimensionality reduction. Hyperparameter optimization was performed to identify the optimal variant of Naïve Bayes models and explore neural network architectures that would achieve maximal accuracy.

4.6. Model Evaluation

Model performance was assessed using cross-validation scores. Metrics pertaining to the algorithms derived from confusion matrices, including accuracy, precision, recall, and F1-score, were compared. Table 5 summarizes the results from both process flows.

4.7. Model Explainability

This section elucidates the decision-making process of the models for specific predictions, employing explainability techniques such as SHAP and LIME. Both methods, implemented via appropriate libraries, elucidated model prediction logic via diverse visualizations, including waterfall, force, and summary plots. Given that certain explainers were restricted to specific model types (e.g., tree-based or linear models), the discussion is organized by the plot type rather than the explainer. This approach facilitates clearer identification of plots compatible with corresponding explainers and model categories, constituting a novel contribution, as summarized in Supplementary Table S3.

SHAP offers local and global interpretability, providing rapid explanations for tree-based models via TreeSHAP, significantly reducing the computation time compared to legacy Kernel SHAP. Herein, relevant SHAP plots addressing the research problem were generated and analyzed for different models. Specifically, summary plots obtained using Kernel SHAP (with logistic regression) and TreeSHAP (with XGBClassifier) exhibited similar key features, although TreeSHAP demonstrated markedly superior efficiency, as presented in Table 6. All plotted results are comprehensively discussed.

A SHAP waterfall plot (Figure 9) displays SHAP values on the X-axis, with corresponding features and their values shown along the Y-axis, highlighting feature interactions beyond individual effects. This plot was essential to analyze the results of the summary plots in detail. The plot indicated that higher source port values are associated with a greater likelihood of malicious packets, as indicated by higher SHAP values.

In a force plot, the individual contributions and directions (positive or negative) of each feature to a specific prediction are visualized for a deep learning model (three-layer sequential network). This enables the identification of features exerting the most influence in classifying a TCP packet as malicious, such as shorter duration, fewer response bytes, and fewer original bytes.

Table 5: Model Performance With/Without Feature Selection and Dimensionality Reduction

Method	Dimensionality Algorithm Reduction		Accuracy (%)	F1 Score
FS + FS	PCA	Logistic Regression	98.00	1.00
FS + FS	LDA	Logistic Regression	98.00	1.00
FS + FS	PCA	XGBoost	99.93	1.00
FS + FS	LDA	XGBoost	99.65	1.00
FS + FS	PCA	Gaussian NB	98.00	1.00
FS + FS	LDA	Gaussian NB	98.00	1.00
FS + FS	PCA	KNN	97.85	1.00
FS + FS	LDA	KNN	97.85	1.00
FS	NA	GaussianNB	84.94	0.90
FS	NA	MultinomialNB	72.13	0.70
FS + FS	PCA	Neural Network	98.94	–
FS + FS	LDA	Neural Network	97.95	–
FS + FS	NA	Neural Network	99.38	–

Abbreviations: FS = Feature Selection; FS + FS = Feature Selection and Scaling; PCA = Principal Component

Analysis; LDA = Linear Discriminant Analysis; KNN = K-Nearest Neighbors; NB = Naïve Bayes; NA = Not Applicable. Only test results are shown. F1 values are not reported for neural networks.

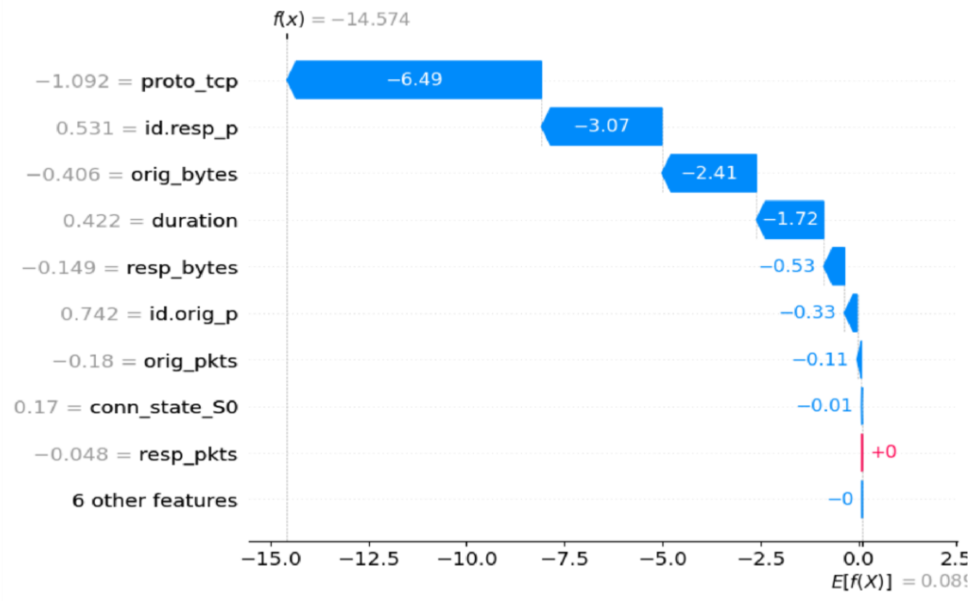


Figure 9: SHAP Waterfall Plot

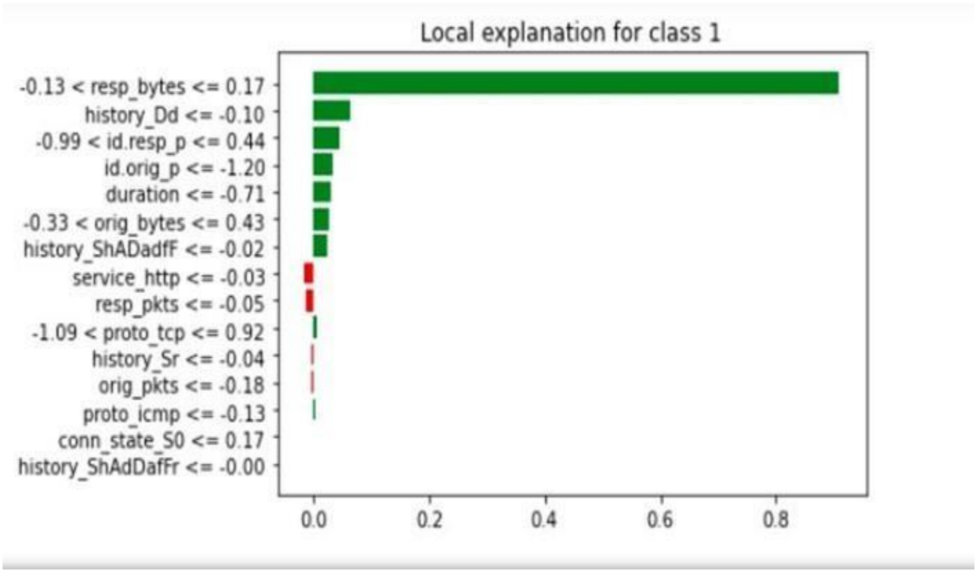


Figure 10: LIME plot B

A waterfall plot (Figure 9) provides a quantifiable, instance-level breakdown of feature contributions to the SHAP value. Unlike the force plot, it specifies the magnitude of each feature's impact; for example, a *duration* value of −0.409 reduces the SHAP value by −1.36, clarifying its effect on designating the class as “Benign.” LIME provides local, model-agnostic explanations by quantifying each feature's influence on an individual prediction (Figure 10), including class probabilities. It enables faster instance-level interpretation than force or waterfall plots. Herein, LIME classified an instance as malicious with 100% confidence, primarily influenced by *resp bytes*, *history Dd*, *id.resp p*, and *id.orig p*. Python packages used in the analysis were standard libraries used for data analysis and ML, including *imblearn*, *lime*, *matplotlib*, *mlxtend*, *numpy*, *pandas*, *python*, *scikit-learn*, *scipy*, *seaborn*, *shap*, and *xgboost*.

4.8. LLMs for Feature-Value Extraction

4.8.1. User Prompt

Figure 11 presents a prompt used to request an LLM (Gemini Flash 2.0) to extract the feature values.

Table 6: Kernel SHAP and TreeSHAP Time Difference

SHAP Method	Time Taken (Samples)	Time Take n (Instances)
Kernel SHAP	2 min (100 samples)	–
TreeSHAP	20 s (200,000 in- stances)	–

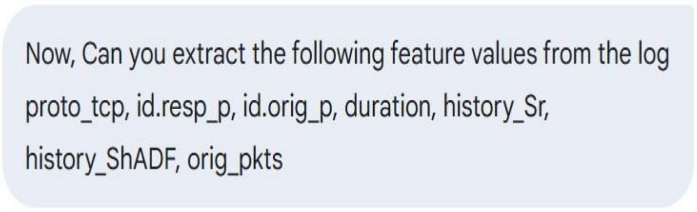

Now, Can you extract the following feature values from the log proto_tcp, id.resp_p, id.orig_p, duration, history_Sr, history_ShADF, orig_pkts

Figure 11: User Prompt For Feature-Value Extraction

4.8.2. Gemini Flash 2.0 Response

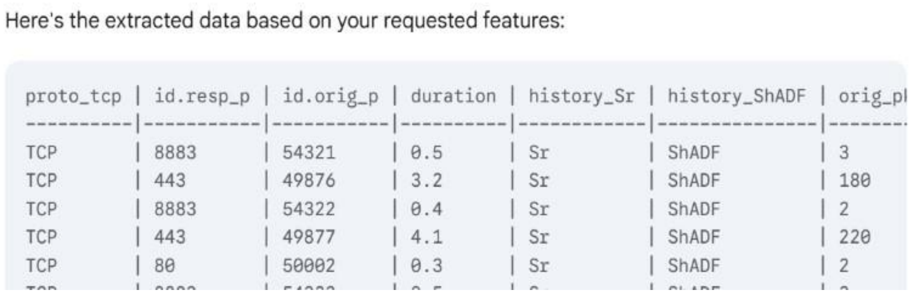
Figure 12 presents a response to the aforementioned prompt, demonstrating that it was successful in extracting the feature values. The link to the entire chat with Gemini can be found in [11].

4.9. Summary

This section presents a stepwise methodology for implementing and modeling the IDS in an IoT environment. The process involved IoT data preprocessing; SMOTE-based oversampling; univariate, bivariate, and multivariate (heatmap) analyses; and feature selection/dimensionality reduction using the chi-square test and PCA or LDA. First, five models were built with PyCaret. Further, SHAP and LIME were employed for explainability, with the application of t-SNE for visualizing SHAP clustering—a novel concept. Notably, an LLM was used for feature extraction from unstructured logs, further contributing to the study's novelty.

5. RESULTS AND DISCUSSION

This section details model results, SHAP and LIME interpretations, and performance discussion across target classes and instances. In addition, it examines algorithm suitability and appropriate metrics for IoT IDS modeling, with guidelines for model output interpretation.


Here's the extracted data based on your requested features:

```
proto_tcp | id.resp_p | id.orig_p | duration | history_Sr | history_ShADF | orig_pl
----------|-----------|-----------|----------|------------|---------------|--------
TCP       | 8883      | 54321     | 0.5      | Sr         | ShADF         | 3
TCP       | 443       | 49876     | 3.2      | Sr         | ShADF         | 180
TCP       | 8883      | 54322     | 0.4      | Sr         | ShADF         | 2
TCP       | 443       | 49877     | 4.1      | Sr         | ShADF         | 220
TCP       | 80        | 50002     | 0.3      | Sr         | ShADF         | 2
```

Figure 12: LLM Response 2 from Gemini: Extracted Feature Values

5.1. Feature Extraction from Unstructured Logs

Utilizing Gemini Flash 2.0, relevant features were reliably extracted for optimal algorithm performance [11].

5.2. Model Performance Analysis

Model performance was evaluated with and without feature selection and dimensionality reduction. The SMOTE was employed to address class imbalance, expanding the dataset from under-sampled 200,000 instances to balanced 323,454 instances. Fifteen correlated features were discarded. Further, three feature selection techniques were applied, with their features combined to obtain the final set. Fisher score was considered, adding novelty to the study. Domain-informed univariate analysis highlighted features such as protocol type (*proto tcp*), port numbers, and packet history (e.g., *history Sr* and *history Dd*).

5.3. Dimensionality Reduction

PCA and LDA were applied for dimensionality reduction, with LDA slightly exceeding PCA in terms of accuracy, precision, and recall. PCA and LDA exhibited comparable results due to prior feature selection and the dataset dimensionality. Notably, the accuracy of GaussianNB improved markedly post-PCA/LDA, which is a novel finding. This may have resulted from component-based modeling, as opposed to feature-based modeling.

5.4. Top Three Models

First, top five models were selected by modeling using PyCaret. Subsequently, these top five models were implemented using two flows as mentioned earlier: one with feature selection and dimensionality reduction and the other with a straightforward model application. XGBoost classifier exhibited the best performance among the logistic regression model, neural network, KNN, and Naïve Bayes model, showing 99.99% accuracy and very good precision and recall scores. The second-best model was the KNN, with an accuracy of 99.82%.

5.5. Why did XGBoost Perform Better than the Other Models?

XGBoost performed better than other models because boosting algorithms give more importance to minor instances that might go misclassified during training in other algorithms. Hence, notably, boosting algorithms will be beneficial for modeling IDSs in IoT environments.

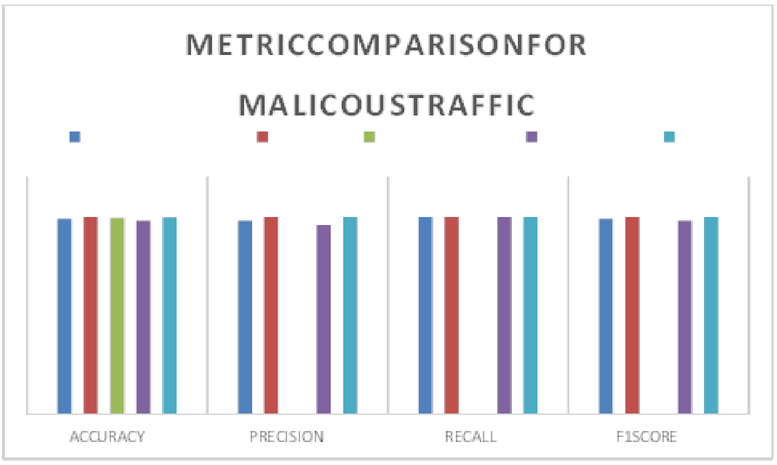


Figure 13: Evaluation metric comparison for malicious data

5.6. Are Clustering Algorithms Worth Experimenting for use in IDSs?

The answer to this question is evidently yes because the KNN achieved a high accuracy and good results in terms of precision, recall, and F1 Scores, showing an accuracy of 99.82%.

5.7. Did Feature Selection with Dimensionality Reduction Lead to a Significant Increase in the Accuracy of Any Model?

Only Gaussian Naïve Bayes models exhibited a significant increase in accuracy (~48%). This was a unique observation not reported in the literature and supports the novelty of this research. This observation can be supported by the fact that PCA and LDA generated different components from the same features, which might be beneficial for creating a component vector instead of a feature vector.

Figure 13 compare different evaluation metrics. Evidently, XGBoost classifier performed best in terms of all metrics for both classes.

5.8. Model Explainability Analysis

Determining which explainability technique to employ and its corresponding scenario was crucial in this research as this research focused on elucidating the reasons behind a particular prediction and its influencing factors. Accordingly, we adopted a distinct process or flow for exploring model explainability, which began with the analysis of plots and their corresponding algorithms, highlighting the importance of the plots and not just the explainability technique. This approach is unique, indicating the novelty of this work. Plots are comprehensively discussed and analyzed, as illustrated in Figures 15 and 16.

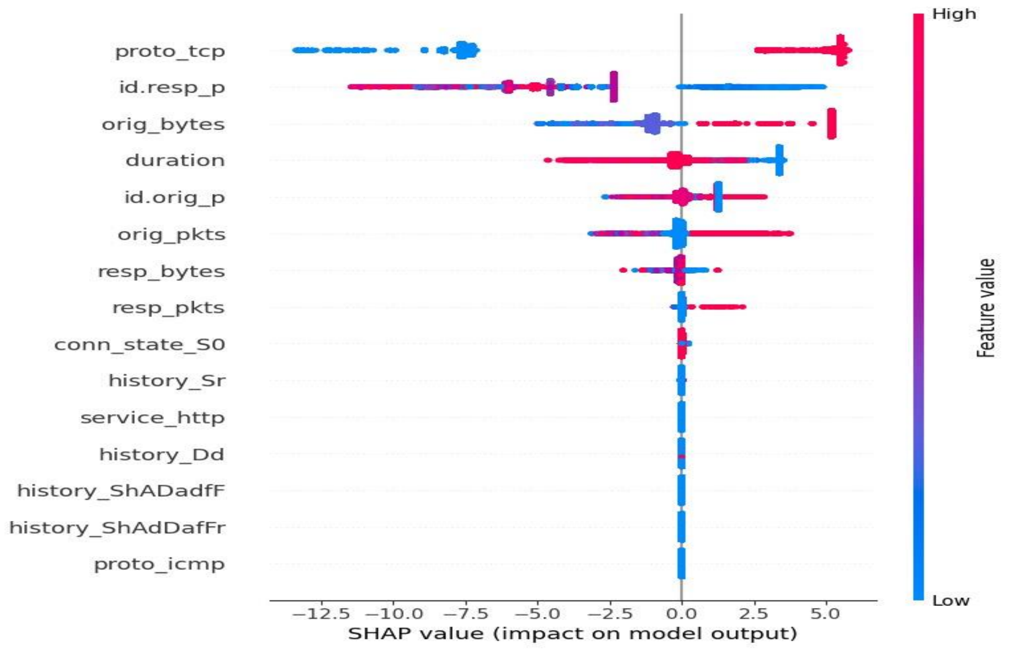


Figure 14: Impact of the SHAP value on model output

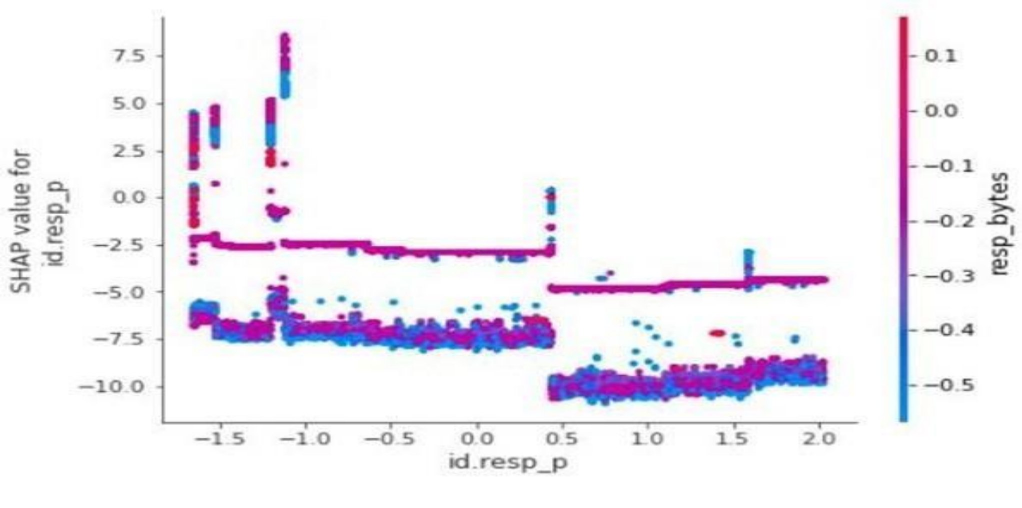


Figure 15: SHAP dependency and inter-relationship plot

5.8.1. Summary Plot

The summary plot shows the contribution of a feature to the predicted result in terms of its SHAP value for a given instance, as illustrated in Figure 14. A high *history S* value increased the SHAP value such that the chances of a packet being malicious increased. The value “1” of *poto _tcp* indicated that the packet was sent via the TCP protocol and had a positive effect on the packet being malicious. The *orig ip bytes* value decreased as the impact on the packet being malicious increased. Lower value of *history D* increased the chances of the packet being malicious. The *orig pkts* value increased with increasing probability of the packet being malicious.

5.8.2. Dependency Plot

The SHAP value distribution of a particular feature can be verified using a dependency plot. The dependency plot (Figure 15) of *id.resp p* demonstrated that the response port’s lower values with higher values in response bytes yield a higher SHAP value of *id.resp p*, and thus a higher chance of the packet being malicious. From this plot, it is observed that at approximately -1.5 (*id.resp p*), if resp bytes shows a high value, then the packet is more probable to be malicious. Hence, a dependency plot can be utilized to conclude the summary plot's observations.

5.8.3. Force Plot

Using a force plot, we can determine the variation and contribution of each feature to a particular instance in the data. In addition, we can deduce the scale to which the feature is contributing to the target value and whether the contribution to the target variable is positive or negative.

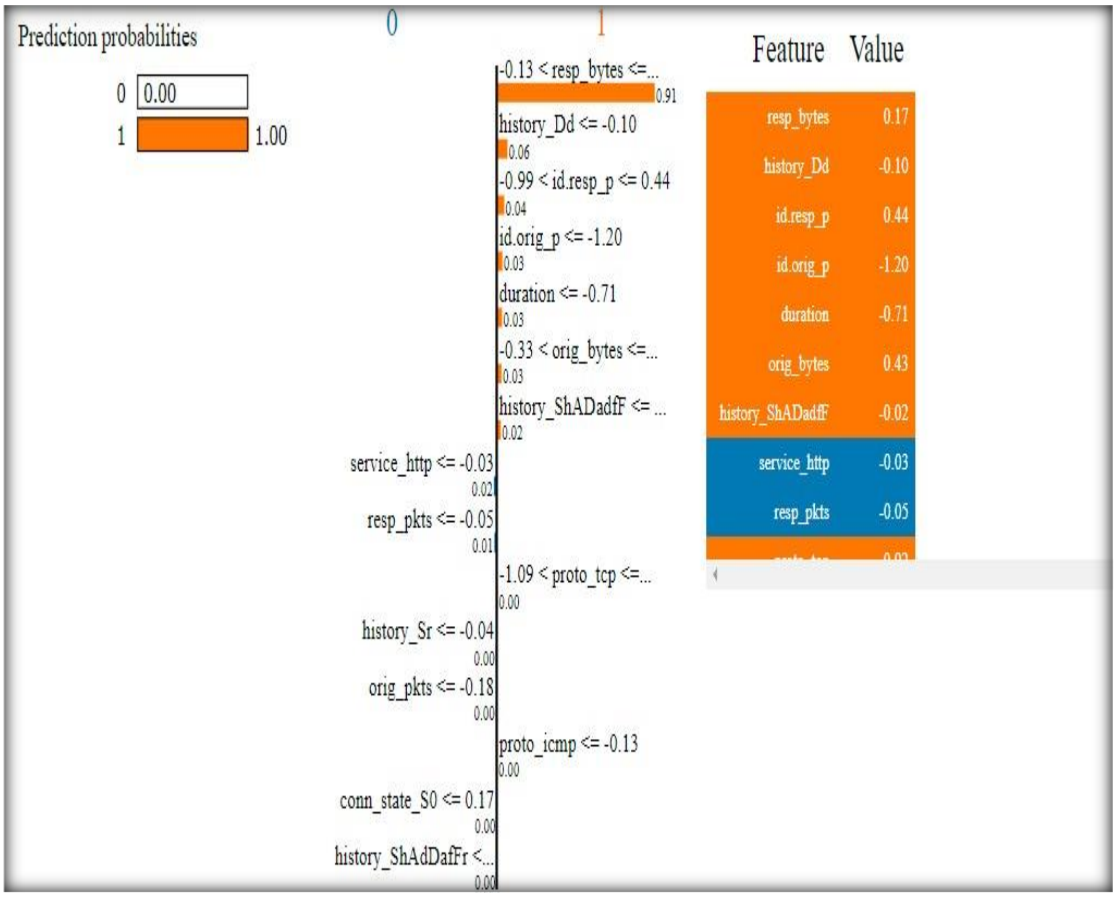


Figure 16: LIME Explainability For Malicious Packet

The force plot demonstrates an instance to be correctly classified as malicious and correctly justifies the summary plot characteristics.

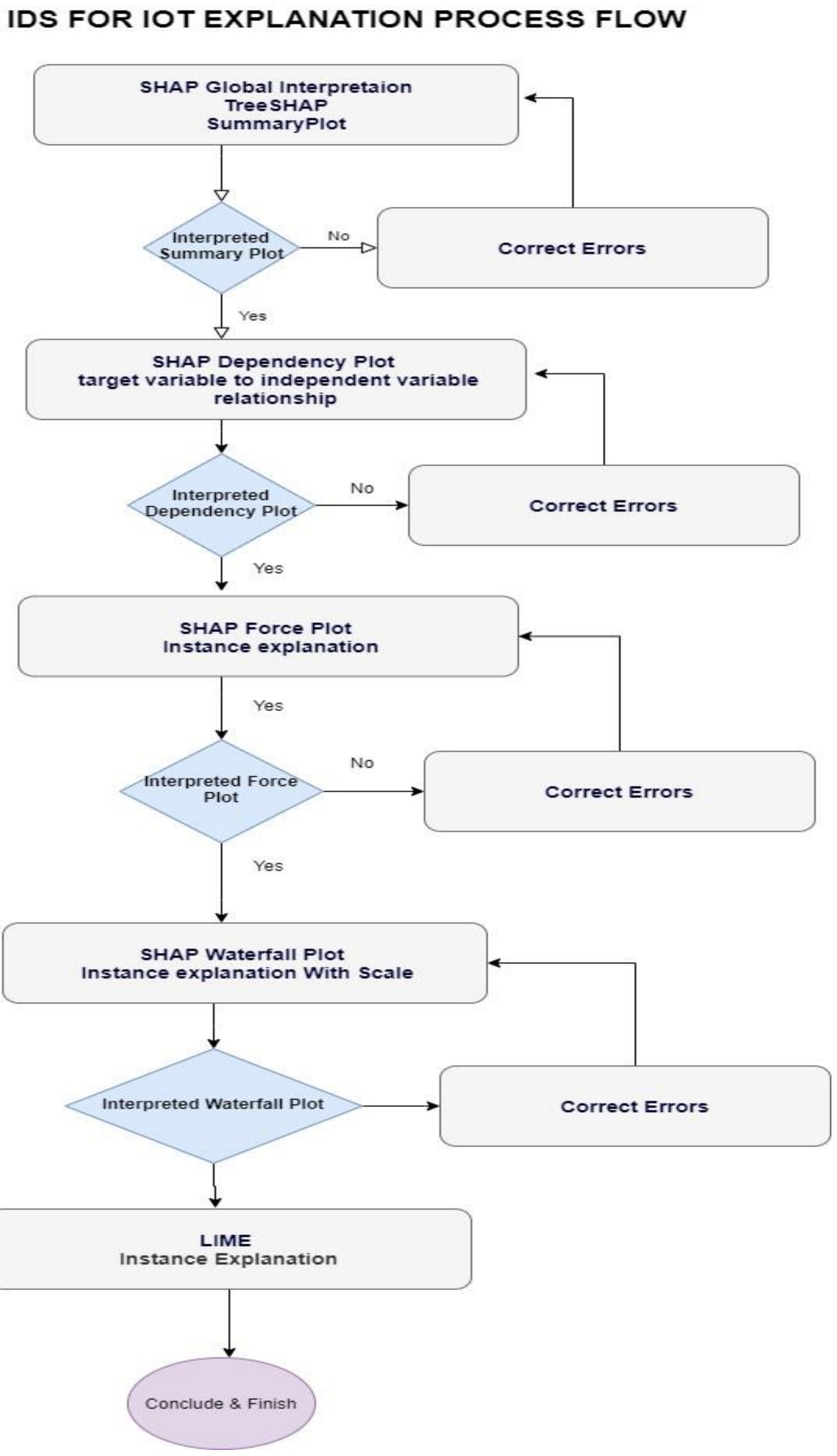

IDS FOR IOT EXPLANATION PROCESS FLOW
SHAP Global Interpretaion TreeSHAP SummaryPlot
Interpreted Summary Plot
No
Correct Errors
Yes
SHAP Dependency Plot target variable to independent variable relationship
Interpreted Dependency Plot
No
Correct Errors
Yes
SHAP Force Plot Instance explanation
Yes
Interpreted Force Plot
No
Correct Errors
Yes
SHAP Waterfall Plot Instance explanation With Scale
Interpreted Waterfall Plot
Correct Errors
LIME Instance Explanation
Conclude & Finish

Figure 17: Prediction Probabilities

5.8.4. LIME

LIME explains the contribution of each feature in an instance to the target variable outcome. Utilizing LIME, we can examine the prediction probability, the feature, and their contributions, together with the values in the input. LIME correctly predicted a malicious packet. LIME also highlighted the range of values for which it classified an instance as malicious. After plotting LIME plots for benign instances, it was facile to segregate the values of the feature for which a particular classification decision is made (Figure 16).

5.9. Is There Any Difference Between Kernel SHAP and TreeSHAP? Which Will Be Recommended for IDS in IoT?

Kernel SHAP is a model-agnostic approach to SHAP value estimation, utilizing sampling-based Shapley value approximation, whereas TreeSHAP is specifically designed for tree-based models, providing exact Shapley values with higher computational efficiency. Given XGBoostClassifier's superior performance and TreeSHAP's faster processing, TreeSHAP is recommended for explainability in IDS for IoT environments.

5.10. What Is the Process Flow in Which Explainability Should Be Targeted in IDS for IoT?

The recommended process for identifying key features begins with a global explanation (e.g., SHAP summary plots) as shown in Figure 17. Subsequently, dependency plots for significant features are analyzed. Instance-level explanations should follow, using SHAP force and waterfall plots to assess local contributions. Obtained results can be cross validated with LIME plots. Both SHAP and LIME methods consistently identified malicious instances and highlighted similar important features in this research.

5.11. SHAP or LIME Instance Explainability in IDS IoT?

When obtaining a faster explanation of an instance is necessary, it is recommended to plot an explanation with LIME. Figure 18 compares feature selection and importance in PCA/Kernel SHAP and TreeSHAP. Evidently, feature Selection, Kernel SHAP, and Tree SHAP yielded the same result in terms of features selected. However, a slight difference emerges in the feature importance, which is because a few features did not significantly affect the model's decision for classification.

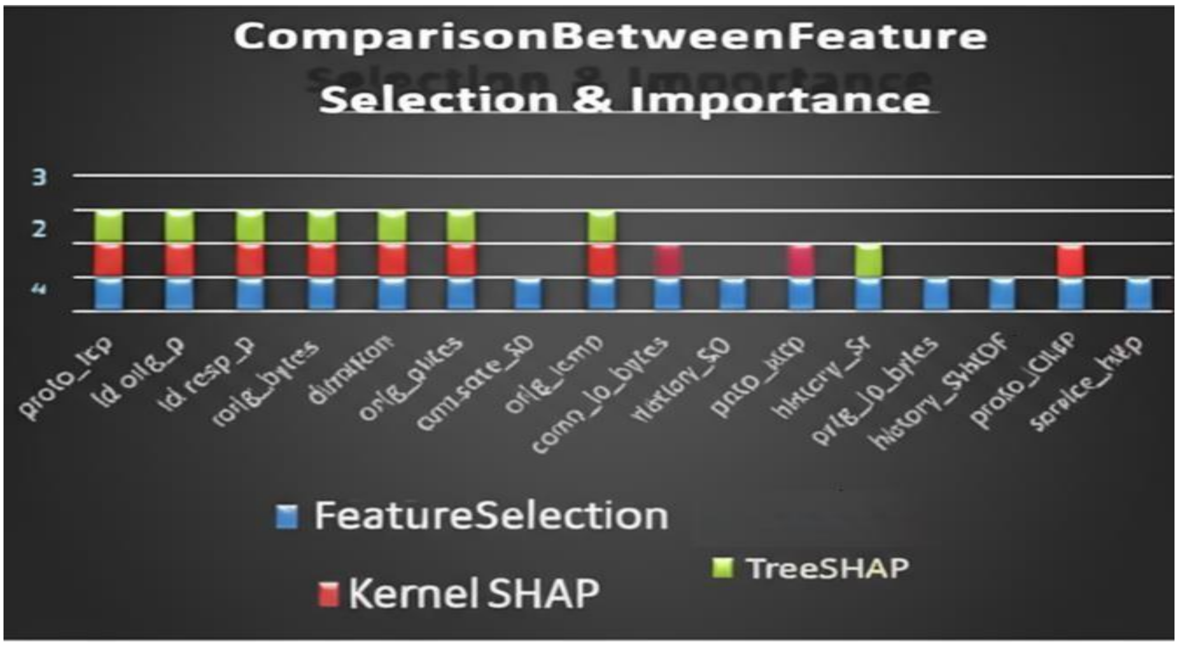


Figure 18: Feature Selection Comparison between PCA/Kernel SHAP and TreeSHAP

5.12. Is t-SNE a Helpful Visualizer for Interpreting Instances in IDS for IoT?

t-SNE is *an optimal* visualizer *in* IDS for IoT *because* we can *observe* the distribution into *several* clusters *regarding* raw data depicting different kinds of attack datasets. t-SNE also *exhibits* proper segregated clustering of SHAP values into two prominent classifications, one for *benign* and the other for malicious. Figures 19 and 20 illustrate how feature clustering improved after creating the model.

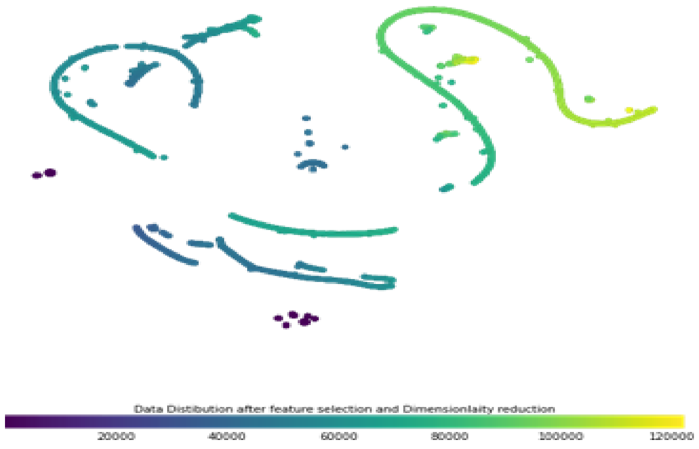


Figure 19: t-SNE Visualization of Data Distribution of instances before modeling using SHAP values

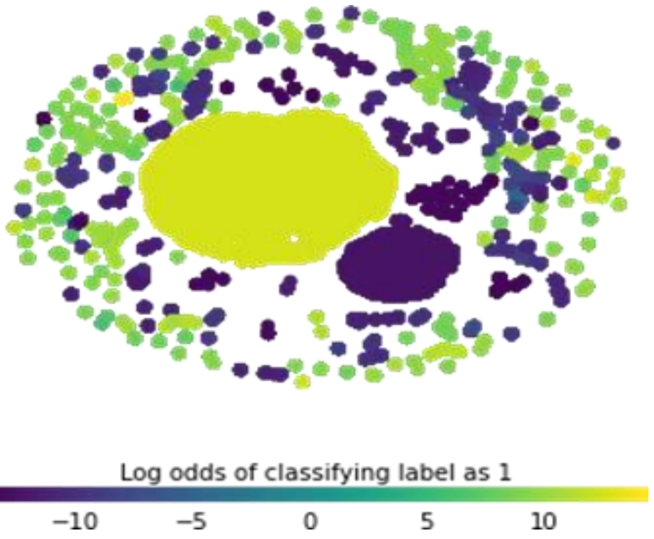


Figure 20: t-SNE Visualization of Data Distribution of the cases after modeling using SHAP values

5.13. Summary

The results indicated that XGBoost classifier outperformed other boosting algorithms. Feature selection via SHAP, LIME, and a dedicated algorithm yielded consistent key features. Notably, LIME provided faster instancelevel classifications than SHAP. All model explainability visualizations were comprehensively analyzed, delineating both positive and negative feature contributions. A systematic process for interpretability was established, and tSNE effectively visualized clustering for feature-selected and SHAP-valued model data.

6. CONCLUSIONS AND RECOMMENDATIONS

This study advanced IDSs used in the IoT using ML via robust workflow design and comprehensive classifier evaluations. Further, a novel application of Fisher scores was developed for feature selection. Key packet features, especially protocol types and flag bits, emerged as the most influential features. Dimensionality reduction significantly enhanced the performance of certain models (e.g., accuracy of the Gaussian Naïve Bayes model increased from 50% to 98%). Model interpretability was addressed using SHAP and LIME. Both techniques reliably identified contributory features, with TreeSHAP demonstrating more computational efficiency than Kernel SHAP and LIME excelling at rapid instance-level clarifications. A structured framework for explainability, which combined global, dependence, and local analyses, was developed, facilitating model interpretability at multiple layers.

The research recommends refining real-time response mechanisms for detected threats in IoT environments and conducting further investigations into advanced feature selection methods, granular explanations of attack subtypes, comparative deep learning analyses, and sampling technique evaluations to enhance IDS effectiveness.

DECLARATION OF COMPETING INTEREST

There is no conflict of interest.

AUTHOR CONTRIBUTIONS

Chandrani Mukherjee: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Roles/writing - original draft, and writing - review & editing.

FUNDING

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

ACKNOWLEDGMENT

The author acknowledges Suvajit Mukhopadhay for his assistance with proofreading the manuscript.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work, the author(s) used ChatGPT [22] to correct grammar and improve the manuscript's clarity. After using this tool/service, the author(s) reviewed and edited the content as required and assume(s) full responsibility for the publication's content

REFERENCES

[1] A. Adadi, M. Berrada, Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI), IEEE. Access. 6 (2018) 52138– 52160. https://doi.org/10.1109/access.2018.2870052.

[2] F. Alizadeh, T. Jakobi, J. Boldt, G. Stevens, GDPR reality check on the right to access data, in: Proceedings of Mensch Und Computer 2019, ACM, NY, 2019, pp. 811–814.

[3] F. Almaguer-Angeles, J. Murphy, L. Murphy, A.O. Portillo-Dominguez, Choosing machine learning algorithms for anomaly detection in smart building IOT scenarios, in: 2019 IEEE 5th World Forum on Internet of Things (WF-IOT), Piscataway, NJ, USA,IEEE, 2019, pp. 491–495.

[4] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, H. Ming, AD-IOT: Anomaly detection of IOT cyberattacks in smart city using machine learning, in: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Piscataway, NJ, USA, IEEE, 2019, pp. 0305–0310. https://doi.org/10.1109/ccwc.2019.8666450.

[5] P. Amangele, M.J. Reed, M. Al-Naday, N. Thomos, M. Nowak, Hierarchical machine learning for IOT anomaly detection in SDN, in: 2019 International Conference on Information Technologies (infotech), Piscataway, NJ, USA, IEEE, 2019, pp. 1–4.

[6] K. Amarasinghe, M. Manic, Improving user trust on deep neural networks based intrusion detection systems, in: IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, Piscataway, NJ, USA, IEEE, 2018, pp. 3262–3268.

[7] Y.Y. Aung, M.M. Min, An analysis of k-means algorithm based network intrusion detection system, Adv. Sci. Technol. Eng. Syst. J. 3 (2018) 496–501. https://doi.org/10.25046/aj030160.

[8] M. Bagaa, T. Taleb, J.B. Bernabe, A. Skarmeta, A machine learning security framework for IOT systems, IEEE Access. 8 (2020) 114066– 114077. https://doi.org/10.1109/access.2020.2996214.

[9] M. B. Antonio, Evaluating the Security of Machine Learning Algorithms, PhD thesis, University of California at Berkeley, USA, 2008.

[10] D. Brauckhoff, K. Salamatian, M. May, Applying PCA for traffic anomaly detection: Problems and solutions, in: IEEE INFOCOM 2009, IEEE, 2009, pp. 2866–2870.s

[11] CM, IoT network logs. https://github.com/ chandrani121189/IOTNetworkLogs/tree/ f33ee715adf38bde70077836d1addd58648b4f2b, 2025 (accessed: 28 May 2025).

[12] S.S. Dhaliwal, A.-A. Nahid, R. Abbas, Effective intrusion detection system using XGBoost, Information. 9 (2018) 149. https://doi.org/10. 3390/info9070149.

[13] F. Doshi-Velez, B. Kim, Towards a rigorous science of interpretable machine learning. http://arxiv.org/abs/1702.08608, 2017 (accessed 27 May 2025).

[14] M. Eskandari, Z.H. Janjua, M. Vecchio, F. Antonelli, Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices, IEEE Internet Things J. 7 (2020) 6882–6897. https://doi.org/10.1109/jiot.2020.2970501.

[15] J. Garc´ıa, M. Grill, J. Stiborek, A. Zunino, IoT-23: A labeled dataset with malicious and benign IoT network traffic (version 1.0.0) [dataset], Zenodo, 2021. https://doi.org/10.5281/zenodo.4743746.

[16] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, A. Robles-Kelly, Deep learning-based intrusion detection for IOT networks, in: 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), Piscataway, NJ, UUSA, IEEE 2019, pp. 256–25609.

[17] D. Gonzalez-Cuautle, A. Hernandez-Suarez, G. Sanchez-Perez, L.K. Toscano-Medina, J. Portillo-Portillo, J. Olivares-Mercado, H.M. Perez-Meana, A.L. Sandoval-Orozco, Synthetic minority oversampling technique for optimizing classification tasks in botnet and intrusiondetection-system datasets, Appl. Sci. 10 (2020) 794. https://doi.org/10.3390/app10030794.

[18] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM Comput. Surv. 51 (2018) 1–42. https://doi.org/10.1145/3236009.

[19] K. Hausken, M. Mohr, The value of a player in n-person games, Soc. Choice. Welfare. 18 (2001) 465–483. https://doi.org/10.1007/s003550000070.

[20] Interpretable Machine Learning. 5.7 Local Surrogate (LIME) — Interpretable Machine Learning. https://christophm.

[21] github.io/interpretable-ml-book/lime.html, 2020 (accessed 14 December 2020).

[22] C. Mukherjee, IDSIOTDatasetAndCode [dataset], Zenodo, v2, 2025. https://doi.org/10.5281/zenodo.15580843.

[23] OpenAI, ChatGPT [large language model]. https://chat.openai.com, 2023 (accessed 3 June 2024).