Lightweight Network Intrusion Detection in IoT Environments Using Machine Learning

[1] C. Satya Kumar, [2] N. Sambasiva Rao, [3] G. Venkata Rami Reddy
[1]Research Scholar, JNTUH, Hyderabad
[2]Professor, IARE Engineering College
[3]Professor, JNTUH, Hyderabad

Abstract: ***With the rapid expansion of IoT devices, network intrusion security has indeed become a major issue. This work introduces a machine learning-based IDS using the IoTID20 dataset representing the imbalanced real-world network traffic. Data preprocessing pipeline converts categorical attributes into numerical attributes through Label Encoding, normalizes data via Min-Max Scaling, and selects features using Pearson Correlation. Three classifiers, the Support Vector Machine (SVM) based on One-Versus-Rest, Decision Tree, and Random Forest, have been evaluated. The SVM achieved an accuracy of 83%, while Decision Tree and Random Forest achieved 89.06% and 96.02% respectively. It is evident from the above results that with the correct preprocessing and feature selection, the detection performance can be greatly affected even on imbalanced datasets. The built system thus functions as an accurate yet lightweight IDS solution that can be deployed in resource-limited IoT environments.***

Keywords-- IoT Security, Intrusion Detection System (IDS), IoTID20 Dataset, Machine Learning, Support Vector Machine (SVM), Random Forest, Decision Tree, Pearson Correlation, Imbalanced Dataset, Min-Max Scaling, Feature Selection, One-Versus-Rest (OVR)

I. INTRODUCTION

The advent of the IoT has truly been transformative with billions of physical entities intricately linked through diverse domains such as healthcare, agriculture, transportation, industry, and smart homes [1]. The devices interact, exchange information, and sometimes act alone to make the world a smarter and smarter place. On the flip side, the very large scale and interconnectivity do pose significant cyber security challenges [2]. Most IoT devices are resource-constrained without embedded security features [3], allowing such devices to be highly vulnerable to cyber threats. Further, many of such devices are even installed with default configurations, weak authentication, or with firmware devoid of all patches, thereby exposing them to malicious activities such as Distributed Denial of Service (DDoS) attacks, spoofing, scanning, and malicious data alteration [4].

In recent times, hard-hitting attacks on the scale of Mirai shown that compromised IoT networks could be terribly destructive—not just for individual devices but also in affecting the wider Internet. Therefore, the varying types of Intrusion Detection System (IDS) must secure an IoT ecosystem. IDS monitors and analyzes network traffic, searching for signs of unauthorized activities or abnormalities. However, the traditional IDS methods are mostly signature-based and depend on an attack being known; thus, they cannot be exploited against zero-day attacks or against a changing threat, which would make it unsuitable for the dynamic and heterogeneous nature of IoT.

Thus, machine learning, because it is based on learning from data and recognizing complicated patterns, necessarily has to conquer the above limitations in order to be used for intrusion detection [5]. ML-IDS judged both known and unknown types of attacks by going beyond predefined rules. They are adaptive, detecting various traffic [6] patterns and working in real-time or near-time. Implementation of an effective ML-based intrusion detection in IoT further faces challenges [7]. The data imbalance problem is the primary one, where clearly benign traffic greatly outweighs the irrelevant malicious activity, introducing biases in the trained systems [8]. Besides that, IoT datasets are generally highly dimensional and mostly will comprise several noisy or irrelevant features that negatively degrade the model's performance and increase the computational cost. Moreover, any proposed solution must be effective yet lightweight to be deployed on devices with resource limitations.

This work aims at resolving these issues through the design of an ML-based IDS by using the IoTID20 dataset, which corresponds to real IoT network traffic recorded in a smart home lab environment. It offers several classes of malicious activity such as Mirai (scan, ACK flooding, and UDP flooding), reconnaissance, and denial of service (DoS) that also include normal traffic. One notable aspect of this dataset is the reflection of real-world scenarios irrespective of class imbalance and the presence of mixed categorical and numerical attributes, making it a fit for real-world development and evaluation of ML-based models.

In essence, the approach is initiated with an extensive preprocessing pipeline encompassing Label Encoding for the categorical features into numerical format, Min-Max

Normalization where numerical values are all mapped to the range [0, 1], and Feature Selection based on Pearson Correlation to reduce features' dimensionality so that only the most informative attributes correlated to the target class were retained.

A total of three popular classifiers from Machine Learning being more interpretable in nature are implemented and evaluated on the processed dataset: Support Vector Machine (SVM) with an OVR strategy, Decision Tree, and Random Forest. The Pearson correlation to observe the effect of feature selection on classification performance [9].

Results show that Random Forest was able to detect SiP patterns to the extent of 96.02% accuracy, followed closely by Decision tree with 89.06% and SVM with 83%. These results lend credence to the research that states that proper preprocessing and feature selection boost the performance of a simple classifier on an IoT dataset. Hence, the research will remobilize a lightweight yet accurate IDS framework for real-time intrusion detection in IoT environments that focus on resource efficiency and scalability.

II. RELATED WORK

With the rapid growth of IoT applications, IoT network security has become a major concern. Conventional Intrusion Detection System (IDS) architectures that rely on signature-based detection fail to detect novel or evolving attacks. Further, these IDS architectures are neither optimized nor considered for the resource-constrained IoT environments. Thus, a shift has occurred toward ML and DL methods for building adaptive and scalable IDSs [10].

Doshi et al. [11] proposed a supervised machine learning approach to detect Distributed Denial of Service (DDoS) attacks generated by compromised IoT devices. In their approach, they created a custom dataset and considered Decision Trees, k-NN, and SVM classifiers that achieved high-level accuracy. The limitation of their work was that it considers only binary classification and only a limited set of attacks. In the same vein, Meidan et al. [12] have introduced N-BaIoT, a deep learning-based anomaly detection system using autoencoder techniques in order to detect behavior of compromised IoT devices [13]. While being effective, the model is computationally expensive and unsuitable for lightweight deployment.

Diro and Chilamkurti [14] addressed data privacy and decentralization by implementing a federated learning approach through Deep Neural Networks (DNN) on the NSL-KDD dataset. Albeit performing rather well, the method needs a huge amount of computational resources and cannot be operated in real time at edge IoT nodes. Iqbal et al. [15], on the other hand, used Random Forest, SVM, and Naïve Bayes on CICIDS2017 and show high accuracy of detection, but this dataset does not depict IoT characteristics and rather resembles traffic on conventional IT networks.

Similarly, Haddad Pajouh et al. [16] and Abeshu and Chilamkurti [17] considered hybrid and layered models of IDS mixing shallow and deep learning methods [18]. While such an architecture and framework improve detection, the increased complexity and overhead may be prohibitive in resource-constrained IoT devices.simpler, more efficient Pearson Correlation as an indigenous mode of filtering the relevant features.

Secondly, most of the referenced work uses old or generic datasets [19] (e.g., NSL-KDD, UNSW-NB15), hence failing to capture the nuances of modern IoT traffic. The IoTID20 dataset, on the other hand, provides a test environment with real IoT devices and varied IoT-specific attacks, including Mirai variants, DoS, and reconnaissance. However, only a handful of studies, thus far, have looked into the effects of feature selection and class imbalance on detection performance within the realms of this dataset.

This study proposes to fill this gap by developing an efficient ML-based Intruder Detection System from the IoTID20 dataset. Hence, the method involves Pearson Correlation-based feature selection, Min-Max normalization, and evaluation of three classical ML algorithms, namely SVM, Decision Tree, and Random Forest. We specifically intend to highlight the effect of correlation thresholds in the performance of classifiers and show that with appropriate optimization, classical ML algorithms can deliver very high accuracy (up to 96.02%) under the imbalanced and high-dimensional setting of IoT data.

Whereas, in the previous studies, much of the focus was given to maximizing the classification accuracy, only a few of them had considered the interplay between feature dimensionality vis-à-vis performance under the constraints of real-time IoT implementation [20]. Feature selection is crucial in ensuring fast detection; additional computational overhead is therefore introduced, which may help in generalizing across various types of attacks. While more powerful models such as deep neural networks [21] do exist, these are practically impossible to deploy at the edge-layer IoT systems setting that requires lightweight, explainable, and quick solutions.

An imbalance in datasets poses another challenge for intrusion detection, just like in the case of IoTID20, wherein most of the traffic is benign, with just a tiny fraction conforming to attacks. Such an imbalance biases the classifiers in favor of the majority, which renders high-level overall accuracy yet poor recall for minority attack classes. This issue must be addressed via careful preprocessing, including proper scaling, label transformation, and feature selection techniques that retain the discriminative power of minority classes.

This method solves the gaps by opting for cheap and fast data preprocessing methods, solving classical machine learning methods that are interpretable and deployable in

the real world. A systematic study of how changing the Pearson correlation thresholds affected accuracy provides viewpoints on the feature selection procedure to be optimal in detection while avoiding overfitting. A comparison of SVM, Decision Tree, and Random Forest that guides model selection with respect to accuracy, simplicity, and computational cost.

In brief, this work contributes to the flourished branch of IoT security, emphasizing feature efficiency, interpretation of classifiers, and deployment feasibility. By utilizing the IoTID20 dataset, the study assesses the models' performances on realistic, imbalanced data to encourage scalable and effective IDS design for next-generation IoT infrastructures.

III. METHODOLOGY

1. Data Collection

The IoTID20 dataset features 625,783 flow records composed of 83 parameters, both categorical and numerical. The features include flow-level statistics such as Flow Duration, Total Fwd Packets, Fwd Packet Length Max, Flow IAT Mean, ACK Flag Count, and Protocol. The class label categorizes each flow into one of six classes: Normal, DoS, Mirai Scan, Mirai ACK Flood, Mirai UDP Flood, and Reconnaissance.

Duplicates and redundant features removed while correlated features dropped to increase the accuracy of the Model. Based on the Violin plot in Table 1, features with very high correlations (44 features) excluded from the dataset. 42 features retained for further analysis shown in Table 2. This step helps ensure that the models will be trained on independent and informative features.

The class distribution is imbalanced in the data, just like the real world where in an IoT network, Malicious attacks are few and far between, and hence difficult to detect.

Table 1. Number of Dropped 44 Correlated features

TotLen_Bwd_Pkts	Fwd_Pkt_Len_Min	Fwd_Pkt_Len_Mean	Bwd_Pkt_Len_Min
Bwd_Pkt_Len_Mean	Flow_IAT_Mean	Flow_IAT_Std	Flow_IAT_Max
Flow_IAT_Min	Fwd_IAT_Tot	Fwd_IAT_Mean	Fwd_IAT_Std
Fwd_IAT_Max	Fwd_IAT_Min	Bwd_IAT_Tot	Bwd_IAT_Mean
Bwd_IAT_Max	Bwd_IAT_Min	Fwd_Header_Len	Bwd_Header_Len
Fwd_Pkts/s	Bwd_Pkts/s	Pkt_Len_Min	Pkt_Len_Max
Pkt_Len_Mean	Pkt_Len_Std	Pkt_Len_Var	PSH_Flag_Cnt
ACK_Flag_Cnt	URG_Flag_Cnt	Pkt_Size_Avg	Fwd_Seg_Size_Avg
Bwd_Seg_Size_Avg	Subflow_Fwd_Pkts	Subflow_Fwd_Byts	Subflow_Bwd_Pkts
Subflow_Bwd_Byts	Fwd_Act_Data_Pkts	Active_Max	Active_Min
Idle_Mean	Idle_Std	Idle_Max	Idle_Min

Table 2. Remaining 42 features

Flow_ID	Src_IP	Src_Port	Dst_IP
Dst_Port	Protocol	Timestamp	Flow_Duration
Tot_Fwd_Pkts	Tot_Bwd-Pkts	TotLen_Fwd_Pkts	Fwd_Pkt_Len_Max
Fwd_Pkt_Len_Std	Bwd_Pkt_Len_Max	Bwd_Pkt_Len_Std	Flow_Byts/s
Flow_Pkts/s	Bwd_IAT_Std	Fwd_PSH_Flags	Bwd_PSH_Flags
Fwd_URG_Flags	Bwd_URG_Flags	FIN_Flag_Cnt	SYN_Flag_Cnt
RST_Flag_Cnt	CWE_Flag_Count	ECE_Flag_Cnt	Down/Up_Ratio
Fwd_Byts/b_Avg	Fwd_Pkts/b_Avg	Fwd_Blk_Rate_Avg	Bwd_Byts/b_Avg
Bwd_Pkts/b_Avg	Bwd_Blk_Rate_Avg	Init_Fwd_Win_Byts	Init_Bwd_Win_Byts
Fwd_Seg_Size_Min	Active_Mean	Active_Std	Label
Cat	Sub Cat		

2. Architecture

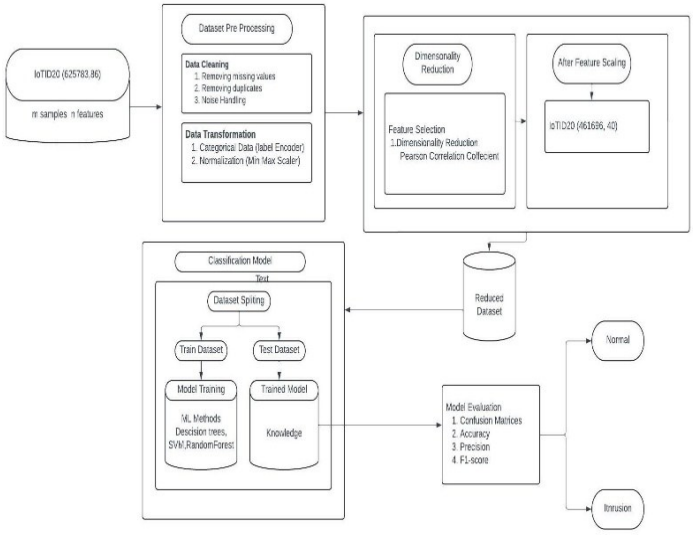

Figure 1. Architecture Diagram

Figure 2 describe the overall architecture. The architecture proposed consists of an adaptable pipeline that accommodates the limitations in the resources in IoT environments. At first, raw traffic data is ingested from IoTID20 and sent further onward to the preprocessing module. The preprocessing includes encoding, normalization, and feature selection. The filtered data serves as training data for the machine learning classifiers, such as Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF). The model evaluation module evaluates the trained models based on a variety of performance metrics. It thus makes the system scalable, efficient, and flexible for different deployment scenarios, either on edge devices or centralized gateways.

3. Algorithms

An intrusion detection model for IoT networks has to be both competent and agile. Thus, three classical machine learning algorithms were selected: Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF). These algorithms were selected since they can produce high classification performance on high-dimensional tabular data while remaining computationally feasible to deploy in resource-constrained environments, such as edge IoT devices. The Support Vector Machine (SVM) algorithm constructs hyperplanes in high-dimensional space to separate different classes. It is particularly useful when data is not linearly separable, as kernel functions like Radial Basis Function (RBF) are used to model complex boundaries. For multi-class classification in this study, an One-Versus-Rest (OVR) method was implemented. The Random Forest model showed good results on the IoTID20 dataset, especially when used in conjunction with Pearson correlation-based feature selection. It recorded its strongest accuracy of 96.02% at a correlation threshold of 0.7, proving to be fair even in imbalanced data situations.

The Decision Tree (DT) algorithm was chosen for its simplicity, interpretability, and speed. It recursively splits the dataset over the branches given feature thresholds, usually chosen depending upon criteria such as Gini impurity or information gain. Being a lightweight and extremely explainable model, a Decision Tree suits the real-time analysis. Slightly below ensemble schemes in accuracy, the Decision Tree provided nice results on the IoTID20 dataset with an accuracy of 89.06% at a correlation threshold of 0.7.

The Random Forest (RF) was implemented in this study as an ensemble of many decision trees to maximize accuracy, reduce stability issues, and in presence of noisy imbalanced data: on-model average voting reduces over-fitting of any one tree and improves generalization. In terms of performance on the IoTID20 dataset, it was the overall winner with an accuracy of 96.02% at a Pearson correlation of 0.7 the statement clearly shows Random Forest's ability to detect complex patterns in multi-class, high-dimensional intrusion-related data.

4. Model Training

The training stage is of paramount significance in building an effective intrusion detection system, more so when faced with a complex and imbalanced dataset such as IoTID20. So, after cleaning and preprocessing the dataset, stratified sampling was employed to split the dataset into 80% training data and 20% testing data, where stratification ensured the preservation of the original class distribution. This step was crucial so that minority classes such as "Reconnaissance" and "Mirai ACK Flood" were able to stand in good numbers for the model while learning.

Each of the three machine learning algorithms, SVM, DT, and RF, was trained individually on the filtered feature sets that were obtained by Pearson correlation-based feature selection, with the aim to study the effect of dimensionality reduction on both the training time and the performance. For SVM, multi-class training was achieved using the OVR category, while the RBF kernel was used for its potential to model non-linear relationships among features.

The decision tree was constructed with Gini impurity as the splitting criterion. For Random Forest, it was instated with multiple estimators or trees, while a randomized approach for feature selection served in better generalization. Grid search was then exchanged for hyperparameter optimization for each model, where SVM hyperparameters such as penalty factor C and kernel coefficient gamma were tuned. For decision trees, the max_depth and min_samples_split parameters were modified, whereas n_estimators (number of trees) and max_features were tuned for the random forest parameters. This made sure that the models produced good accuracy and efficiency, avoiding overfitting and excessive computation time. Training carried out in Python, leveraging the Scikit-learn library, each model was trained separately on feature subsets as per Pearson correlation 0.7. This helped us analyze the best balance between feature reduction and classification accuracy.

Random Forest gave the best accuracy of 96.02% when trained with features selected at a threshold of 0.7, while support vector machines could achieve 83% accuracy at a threshold of 0.7 and decision tree achieved down to 89.06%. It is due to this training scheme that each model is well tested and tuned to perform at its highest peak on the IoTID20 dataset, especially prioritizing the generalization of the model and real-time applications to IoT security systems.

5*. Handling Class Imbalance*

Class imbalance poses a serious problem for intrusion detection, particularly in real-world datasets such as IoTID20, where malicious traffic is greatly outnumbered by legitimate network traffic. The majority of records in the IoTID20 dataset show benign activity, but attack classes like Mirai ACK Flood, DoS, and Reconnaissance are much less common. Due to this imbalance, biased models may have high overall accuracy but struggle to detect minority class attacks, failing to recognize threats when they arise.

In order to overcome this, our methodology included a number of techniques that lessened the effects of imbalance without changing the dataset's realistic distribution. In order to guarantee that every class, particularly the uncommon attack classes, was fairly represented in both the training and testing sets, we first used stratified train-test splitting.

Next, we concentrated on evaluation metric selection to account for imbalance rather than using artificial oversampling methods like SMOTE, which could introduce noise or irrational patterns. We focused on class-specific metrics like Precision, Recall, and F1-Score rather than just accuracy, especially for minority classes. These metrics offered a more insightful evaluation of the model's attack detection capabilities. In order to examine false negatives in attack detection—which are crucial in intrusion detection scenarios—confusion matrices were also thoroughly examined. Furthermore, class imbalance was naturally addressed by using Random Forest and SVM with class weighting. The ensemble approach of Random Forest is more resilient to unbalanced distributions because it combines decisions from several decision trees that are trained on bootstrapped subsets and randomly chosen features **[**22]. In a similar vein, SVM was set up with a balanced class weight parameter, which enabled it to impose a heavier training penalty on misclassifications from underrepresented classes. Combining these methods allowed the suggested system to greatly enhance detection performance for minority attack classes while maintaining high accuracy. This well-rounded strategy guarantees the system's practicality in the real world, where protecting IoT

networks requires early detection of uncommon attacks [23].

6. Feature Selection

Effective feature selection is essential for lowering computational overhead and enhancing model performance because of the high dimensionality of the 83-feature IoTID20 dataset. The dataset contains a number of features that might be redundant or correlated, including flow durations, packet counts, TCP flag counts, and inter-arrival times. Particularly in IoT environments with limited resources, training models on all available features may result in overfitting, longer training times, and poor generalization. We used a filter method [24] that measures the linear relationship between each feature and the target class, Pearson Correlation Coefficient using equation (1), based feature selection, to address this. Features with low absolute correlation values were gradually eliminated because they make minimal contributions to the model's decision boundary. To determine the ideal subset of features that maximized accuracy while minimizing redundancy, we assessed the effects of correlation threshold 0.7.

By using this technique, we were able to keep the most informative attributes while reducing the feature space. Features like Flow Duration, Total Fwd Packets, Fwd Packet Length Max, Flow IAT Mean, and ACK Flag Count, for instance, were maintained across several thresholds and continuously demonstrated a strong correlation with the attack labels. These characteristics are essential for detection because they are known to display unique patterns of behavior in attack scenarios such as Mirai floods or reconnaissance scans.

Additionally, training speed and model interpretability improved as a result of the feature count reduction. The Random Forest classifier reached its maximum accuracy of 96.02% when the number of features was decreased using a Pearson threshold of 0.7 proving that the best feature selection can improve performance without compromising accuracy. Additionally, the models became lighter and more deployable in real-time IoT security solutions by lowering the number of input variables. Overall, the efficiency and accuracy of our intrusion detection models on the IoTID20 dataset were greatly increased by the computationally cheap and efficient Pearson correlation-based feature selection method.

7. Model Evaluation

Measuring overall accuracy alone is insufficient to assess intrusion detection model performance in an IoT context, particularly when class imbalance and multi-class classification are involved, as demonstrated by the IoTID20 dataset. In order to evaluate each model's accuracy in identifying both the minority attack classes and the majority class (normal traffic), a thorough evaluation approach was used. The evaluation metrics that are employed are Accuracy, Precision, Recall, F1-Score, and Confusion Matrix. Across all classes, accuracy served as a broad indicator of accurate classifications using equation (2). However, because the dataset was unbalanced, Precision—which calculates the percentage of true positive detections among all positive predictions using equation (3) and Recall—which quantifies the model's capacity to identify all real positive samples using equation (4) added to it. For intrusion detection systems, these metrics are especially important because high precision lowers false alarms and high recall guarantees fewer undetected attacks (false negatives). A single balanced metric for assessing the trade-off between precision and recall was the F1-Score, which is the harmonic mean of the two using equation (5).

To further illustrate the distribution of accurate and inaccurate classifications across all classes, confusion matrices created for each model. We were able to examine misclassifications for crucial attack types like Mirai ACK Flood and Reconnaissance, which were easily overlooked because of their infrequent occurrence, thanks to these matrices. To determine how feature selection affected detection capability, performance was monitored for each classifier—SVM, Decision Tree, and Random Forest—across various Pearson correlation thresholds.

With an accuracy of 96.02% and strong class-wise recall values, especially at a Pearson threshold of 0.7, the Random Forest model produced the best overall results. The Decision Tree model also reached a peak accuracy of 89.06%. Despite achieving a lower overall accuracy of 83%, SVM.

In conclusion, a thorough evaluation of the model's quality was made possible by the application of several evaluation metrics and correlation-based performance tracking. With the help of this assessment method, the suggested IDS is guaranteed to function well in highly unequal class environments, allowing for prompt and precise detection of malicious behavior in IoT networks.

Pearson Correlation Coefficient (r) =

$$\frac{\sum_{i=1}^{n}(x_i-x\prime)(y_i-y\prime)}{\sqrt{\sum_{i=1}^{n}(x_i-x')^2*\sum_{i=1}^{n}(y_i-y')^2}} \quad (1)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

$$F1-score = \frac{2*Precision*Recall}{Precision+Recall} \quad (5)$$

8. Performance Metrics and Visualization

A range of performance metrics and visualization techniques were used to evaluate the suggested intrusion detection system's efficacy. These measurements were

selected in order to give a thorough grasp of the model's advantages and disadvantages, especially with regard to managing multi-class classification and class imbalance, which are features found in the IoTID20 dataset. Beyond simple accuracy, each metric provided a distinct perspective for interpreting model behavior and assessing detection performance in the real world.

The percentage of correctly classified instances over all predictions was calculated using accuracy as a baseline metric. Accuracy by itself can be deceptive in imbalanced datasets, but it is useful for general benchmarking. As a result, other metrics were given priority, particularly F1-Score, Precision, and Recall. Recall measures the proportion of actual attacks that were successfully identified, whereas precision quantifies the proportion of predicted attack instances that were actually attacks. These metrics are essential for IoT security systems because false positives can cause far more harm than false negatives, which fail to detect an attack. To assess overall classifier performance, especially in situations of class imbalance, the F1-Score—a balanced metric that balances precision and recall—was employed.

Figure 2 show the Plotting confusion matrices for each algorithm, which display the number of true positives, false positives, false negatives, and true negatives for each class, allowed for the visualization of model performance. Furthermore, precision, recall, and F1-score values were compared between models at Pearson correlation threshold 0.7 represented in the Table 3 and using bar graphs shown in figure 3. These plots demonstrated the effects of correlation-based selection on classification performance when fewer features were used.

With the highest accuracy (96.02%) and balanced precision-recall values across classes, the Random Forest classifier continuously outperformed the other models in all metrics, according to the results. With an accuracy of 89.06%, the Decision Tree classifier fared reasonably well, particularly when the Pearson correlation threshold was set at 0.7. The SVM despite having a slightly lower accuracy rate of 83%.

Together, these metrics and visualizations showed that well-tuned classical machine learning models, bolstered by thoughtful feature selection and appropriate assessment, can deliver high-performance intrusion detection appropriate for real-time implementation in Internet of Things systems.

IV. TOOLS AND LIBRARIES USED

Because of their stability, scalability, and community support, a collection of open-source tools and Python-based libraries were used to implement the suggested intrusion detection system. Python 3.10's user-friendliness and robust ecosystem for data science and machine learning applications led to the implementation of the full model development pipeline, including feature selection, training, evaluation, and visualization, as well as data preprocessing. The Scikit-learn library (sklearn) was used to create the three main machine learning models: Random Forest (RF), Decision Tree (DT), and Support Vector Machine (SVM). Model evaluation functions, preprocessing tools, and classification algorithms were all effectively implemented by this library. Large tabular data structures were commonly managed using Pandas for data handling and manipulation, while high-performance numerical computations were supported by NumPy.

Libraries like Matplotlib and Seaborn were used to visualize feature distributions, correlation matrices, model performance metrics, and confusion matrices. These libraries made it possible to create bar plots of expert quality—all of which were essential for presenting and interpreting the results. The built-in tools of Scikit-learn were also utilized to create stratified train-test splits and calculate important performance metrics like precision, recall, and F1-score.

The experiments were carried out in a personal workstation running 64-bit Windows 11 with an Intel Core i7 processor and 16GB of RAM. Jupyter Notebook, an interactive and iterative development workflow perfect for debugging, visualizations, and step-by-step experimentation, was used to create and run all of the scripts.

The system was made reproducible and easily extensible by this combination of open-source libraries and tools, which made it appropriate for both future deployments in actual IoT environments and scholarly research.

V. RESULTS

The IoTID20 dataset, which consists of 625,783 records and 86 columns (83 features + 3 identifiers), was used for the experimental evaluation. Table 3 represents that the three machine learning algorithms—SVM, Decision Tree, and Random Forest—assessed on smaller feature sets following feature selection using Pearson Correlation and preprocessing with Label Encoding and MinMaxScaler. 42 features were chosen using the SVM classifier, which greatly reduced the dimensionality, with a Pearson correlation threshold of 0.7. The Random Forest model demonstrated its resilience in learning efficient class boundaries even in high-dimensional and unbalanced datasets, as evidenced by its 96.02% accuracy rate despite the smaller feature set. The IoTID20 dataset, which consists of 625,783 records and 86 columns (83 features + 3 identifiers), was used for the experimental evaluation. 42 features were chosen after the Decision Tree (DT) classifier was assessed with a correlation threshold of 0.7. With 89.06% accuracy rate, this model demonstrated a strong capacity for accurate network traffic classification. Decision Tree models are renowned for their interpretability and low computational cost, which makes

them perfect for lightweight IoT deployments, even though their performance was marginally worse than that of the other models.

With an accuracy of 96.02%, the Random Forest (RF) model performed better than any other model. It was also trained on 42 features that were chosen at a 0.7 threshold. Its exceptional performance was influenced by its ensemble nature and capacity to manage noisy or unbalanced data. The effectiveness of RF for real-world intrusion detection, where multiple attack types must be detected under skewed conditions, is confirmed by its high accuracy on the IoTID20 dataset.

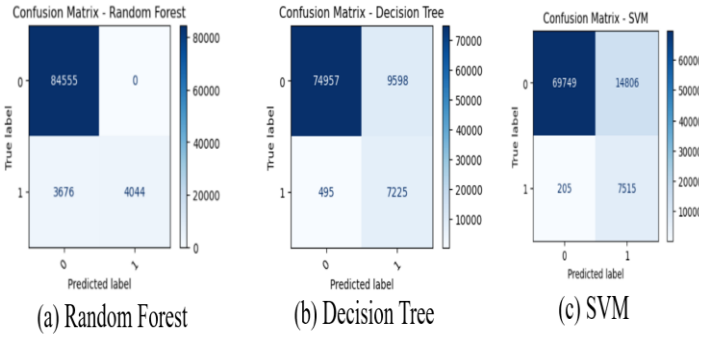

Figure 2. Confusion Matrix of Three Models

IoTID20.csv (Imbalanced dataset)

Algorithm	Dataset size (Original)	Data Pre-Processing	Feature Scaling	Threshold	Dataset Size (Pearson)	Accuracy	Precision	Recall	F1 Score
Random Forest	(625783, 86)	LabelEncoder, MinMaxScaler	Pearson Correlation	0.7	(461373,42)	96.02%	97.92%	76.19%	83.31%
Decision Tree	(625783, 86)	LabelEncoder, MinMaxScaler	Pearson Correlation	0.7	(461373,42)	89.06%	71.15%	91.12%	76.28%%
SVM	(625783, 86)	LabelEncoder, MinMaxScaler	Pearson Correlation	0.7	(461373,42)	83%	67%	82.40%	70%

Table 3. Model Comparison on IoTID20 Datasets

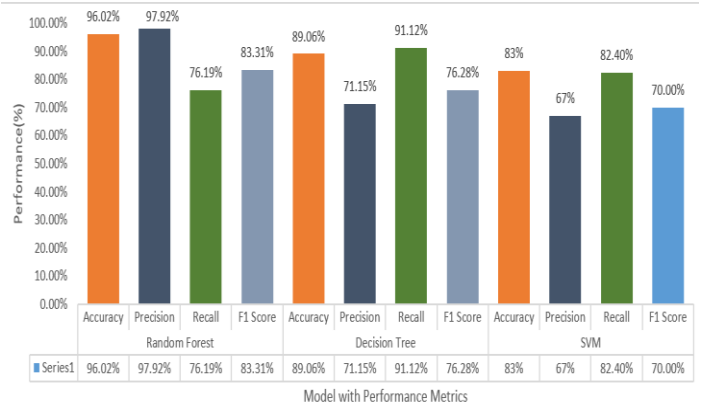

Figure 3. Classifier performance on IoTID20 Dataset.

VI. CONCLUSION AND FUTURE SCOPE

In this study, we used the IoTID20 dataset, which comprises more than 625,000 actual network traffic records, to propose a lightweight and efficient Intrusion Detection System (IDS) for IoT networks using classical machine learning algorithms. In order to overcome issues like high dimensionality and class imbalance, the methodology combined Pearson Correlation-based feature selection, Min-Max Normalization, and Label Encoding. Using 42 features, Random Forest had the highest accuracy of any of the models evaluated (96.02%), closely followed by Decision Tree 89.06% accuracy and Support Vector Machine (SVM) 83% accuracy. These findings show that traditional machine learning models can achieve high detection performance and still be computationally feasible for deployment in resource-constrained IoT environments if they are properly optimized.

The study also emphasizes how crucial feature selection and class imbalance management are to enhancing IDS performance without the need for extensive preprocessing or deep learning. The findings confirm that, even in unbalanced environments, a carefully designed feature set in conjunction with ensemble or kernel-based learning can effectively identify several attack classes, such as DoS, Mirai variants, and reconnaissance. Robustness and generalizability will be further improved by extending the work to incorporate deep learning architectures or hybrid models and testing them on multi-source or real-time IoT traffic. In order to adjust to changing attack patterns in dynamic IoT environments, the system might also be expanded into online learning modes.

Moreover, integrating ensemble deep learning models like CNN-LSTM or Transformer-based IDS can be investigated to capture temporal and spatial patterns in network flows, even though this study concentrated on classical ML models because of their interpretability and computational efficiency.

Using explainable AI (XAI) techniques to improve the transparency of decisions made by black-box models such as Random Forest and SVM is another exciting avenue. By revealing which features have the greatest influence on intrusion decisions, tools like SHAP (SHapley Additive Explanations) and LIME (Local Interpretable Model-agnostic Explanations) can increase the system's credibility for use in mission-critical Internet of Things applications.

Future versions of this system can package and deploy the models on real IoT gateways or edge devices by utilizing containerization with Docker or lightweight runtime environments. By doing this, the current offline detection framework would be converted into a real-time intrusion detection system (RT-IDS) with the ability to generate alerts and monitor packets in real time. Long-term detection performance in dynamic IoT ecosystems may also be maintained and concept drift may be lessened with the use of adaptive learning techniques, in which the IDS continuously retrains on freshly observed data. In conclusion, the encouraging outcomes from the IoTID20 dataset provide a solid basis for developing real-time,

scalable, and interpretable IDS solutions that are suited for contemporary IoT infrastructures.

REFERENCES

[1] R. Mitchell and I.-R. Chen, "A Survey of Intrusion Detection Techniques for Cyber-Physical Systems," ACM Computing Surveys, vol. 46, no. 4, pp. 55:1–29, 2014.

[2] D. Sfar et al., "A Roadmap for Security Challenges in the Internet of Things," Digital Communications and Networks, vol. 4, no. 2, pp. 118–137, 2018.

[3] M. A. Esfahani et al., "IoT Intrusion Detection Using Deep Learning and Semantic Feature Embeddings," Journal of Network and Computer Applications, vol. 177, 2021.

[4] N. Hubballi and V. Suryanarayanan, "Layer-Wise Detection of DDoS Attacks Using Machine Learning," Computer Communications, vol. 122, pp. 36–45, 2018.

[5] M. T. Hossain et al., "A Review of Machine Learning Algorithms for Detection of Intrusion in Network Traffic," ICT Express, vol. 6, no. 3, pp. 175–179, 2020

[6] F. J. Iglesias and T. Zseby, "Analysis of Network Traffic Features for Anomaly Detection," Machine Learning, vol. 101, no. 1–3, pp. 59–84, 2015.

[7] Y. Wu et al., "A Survey on Intrusion Detection for Internet of Things: State-of-the-Art and Challenges," Computers & Security, vol. 102, 2021.

[8] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303–336, 2013

[9] M. E. S. Rahman and A. Islam, "Feature Reduction for Intrusion Detection Using Principal Component Analysis and Autoencoder," Procedia Computer Science, vol. 199, pp. 253–260, 2022.

[10] M. A. Ferrag and L. Maglaras, "Deep Learning for Cybersecurity: A Review of Recent Advances," IEEE Access, vol. 8, pp. 71564–71580, 2020.

[11] A. Doshi, J. Apthorpe, and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," in *IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 29–35.

[12] Y. Meidan et al., "N-BaIoT: Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," IEEE Pervasive Computing, vol. 17, no. 3, pp. 12–22, 2018.

[13] T. N. Dinh, H. T. Nguyen, and N. T. Nguyen, "Intrusion Detection for IoT Devices Using Deep Learning and AutoEncoder," in Proceedings of the 11th International Conference on Knowledge and Systems Engineering, 2019.

[14] A. A. Diro and K. Chilamkurti, "Distributed Attack Detection Scheme Using Deep Learning Approach for Internet of Things," Future Generation Computer Systems, vol. 82, pp. 761–768, 2018.

[15] W. Iqbal et al., "Anomaly Detection Using Machine Learning in Internet of Things: Current Trends, Issues and Challenges," IJACSA, vol. 9, no. 8, pp. 65–74, 2018.

[16] H. A. HaddadPajouh et al., "A Survey on Internet of Things Security: Requirements, Challenges, and Solutions," Computer Networks, vol. 148, pp. 241–259, 2019.

[17] F. M. Abeshu and N. Chilamkurti, "Deep Learning: The Frontier for Distributed Attack Detection in Fog-to-Things Computing," IEEE Communications Magazine, vol. 56, no. 2, pp. 169–175, 2018.

[18] G. M. T. Abdulkadir et al., "Intelligent Intrusion Detection System for IoT Network Based on Hybrid Feature Selection and Machine Learning," Computers & Security, vol. 109, 2021

[19] A. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems," in MilCIS, 2015, pp. 1–6.

[20] M. A. Ferrag et al., "Deep Learning Approaches for Cyber Security Intrusion Detection: A Review," IEEE Access, vol. 7, pp. 86129–86149, 2019.

[21] M. A. Jan et al., "A Deep Learning-Based Network Intrusion Detection System for Healthcare IoT," IEEE Access, vol. 9, pp. 121434–121449, 2021.

[22] A. B. Kiani and M. R. Meybodi, "An Efficient Intrusion Detection System Based on Feature Selection and Ensemble Learning," Expert Systems with Applications, vol. 138, pp. 112–119, 2019.

[23] T. R. Gadekallu et al., "Early Detection of DDoS Attacks in 5G-Enabled IoT Networks Using Deep Learning," IEEE Internet of Things Journal, vol. 9, no. 15, pp. 12085–12094, 2022.

[24] M. A. Ambusaidi et al., "Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm," IEEE Transactions on Computers, vol. 65, no. 10, pp. 2986–2998, 2016.