|| Volume 9 || Issue 7 || July 2025 || ISSN (Online) 2456-0774
INTERNATIONAL JOURNAL OF ADVANCE SCIENTIFIC RESEARCH AND ENGINEERING TRENDS

Improved Interpretability with the use of Explainable AI for Intrusion Detection and Classification in Internet of Things Networks

[1]A A R Senthil Kumaar, [2]Dr. U. Moulali
*[1]*Research Scholar, Department of Computer Science, Faculty of Computing & Information Technology, Himalayan University, Itanagar, Arunachal Pradesh, India.*
[2]Supervisor, Department of Computer Science, Faculty of Computing & Information Technology, Himalayan University, Itanagar, Arunachal Pradesh, India.

---***---

Abstract: The exponential growth of the Internet of Things (IoT) has been phenomenal, changing the face of several sectors and making formerly mundane tasks much easier. There is a critical need to ensure the safety of IoT networks due to the fact that the expansion of IoT devices has created new entry points for cybercriminals. To protect the confidentiality and availability of IoT networks, intrusion detection and classification systems are vital. Cyberattacks nowadays are very dynamic and complex, making it difficult for traditional intrusion detection systems that rely on rules or signatures to stay up. Consequently, there has been a lot of focus on using machine learning techniques for intrusion detection and classification in IoT networks recently. Intrusion detection systems can adapt and develop with new threats by using machine learning algorithms to automatically discover patterns and correlations from enormous amounts of data. The rapid expansion of IoT devices has increased security vulnerabilities, necessitating robust intrusion detection mechanisms. This research proposes a hybrid approach combining XGBoost for feature selection and deep learning models (CNN, LSTM, and MLP) for effective intrusion detection. We employ CNN for spatial feature extraction, LSTM for temporal pattern recognition, and MLP for classification. By examining data from network traffic, these algorithms may identify unusual behavior and group it into distinct types of attacks, safeguarding the system in real-time. Exploring and evaluating several machine-learning techniques for intrusion detection and classification in IoT networks is the goal of this project. The goal is to find the best models for detecting and correctly classifying intrusions in IoT settings. We will examine the efficacy of common machine learning methods including Decision Trees, Random Forest, Support Vector Machines (SVM), Neural Networks, and Ensemble Techniques by making use of the extensively used UNSW-NB15 dataset (Moustafa & Slay, 2016), which is an exhaustive dataset for network intrusion detection. We hope to find out what works and what doesn't by do an empirical evaluation of these algorithms using the UNSW-NB15 dataset. To evaluate how well the models, identify and categorize network intrusions, performance measures including recall, accuracy, precision, and F1-score will be utilized. In order to better protect IoT networks from cyber-attacks, this study will examine the possibility of using machine learning-based techniques.

Keywords: *Internet of Things, Cyberattacks, XGBoost, CNN, LSTM, and MLP.*

---***---

I.INTRODUCTION:

Advanced machine learning algorithms and models provide significant benefits for enhancing IoT network security. These techniques analyze input features from network traffic, assisting cybersecurity personnel in making critical threat detection decisions. However, the complexity of these models often makes them difficult for human analysts to interpret. As a result, analysts may rely on traditional tools that, while more explainable, may not be as effective for modern cybersecurity challenges.

Understanding the internal workings of an ML-based intrusion detection system (IDS) is often challenging, which can reduce trust in its predictions—even when performance metrics indicate high accuracy. Without clear insights into the decision-making process, cybersecurity experts may hesitate to rely fully on ML models.

Explainable AI (XAI) provides a range of tools for interpreting model behavior and assessing feature importance. By offering transparent explanations of how ML models make decisions, XAI enhances trust in their predictions, enabling cybersecurity professionals to better understand classifier behavior and confidently integrate ML-driven insights into their security strategies.

UNSW-NB15 Dataset

Using classifications like "fuzzer," "analysis," "backdoor," "denial of service," "exploit," "generic," "reconnaissance," "shellcode," and "worm," UNSW-NB15 organizes network traffic records based on the Internet of Things. He used the IXIA Perfect Storm program from the Cyber Range Lab of the Australian Cyber Security Center (ACCS) to construct raw network packets from the UNSW-NB 15 dataset, which were then mixed with real-world routine activity and modern threats. I managed to catch a variety of motions. Internet of Things (IoT) - The foundation makes the network. Figure 3.1 illustrates the process of creating the UNSW-NB15 testbed's configuration records and functions.

The developer pre-splits UNSW-NB15 into two sets, UNSW_NB15_trainingset.csv and UNSW_NB15_testing-set.csv, so that the model may be trained and tested separately. Each record in the test set—which has 82,332 records—includes a target response, an attack, and the expected behavior related to traffic. The training set contains 175,341 data. A total of 39 numerical characteristics make up the dataset. The UNSWNB15 features.csv file lists the features along with their descriptions. The experimental procedure will be enhanced by using a binary categorization of aggressive and normal behavior as the goal attribute. Within the data subset, Figure 2 displays the details and score distribution of each assault class. The range from 0 (normal) to 1 (aggressive) is rather broad. It is clear that the dataset contains balanced activity behavior binary response variables.

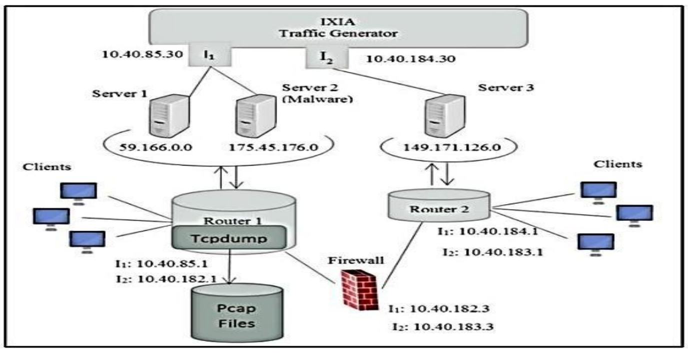


Figure 1: IXIA Traffic Generator Overview

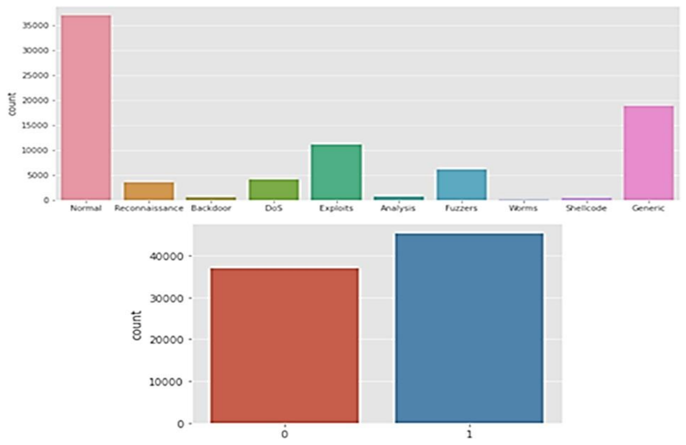


Figure 2: Training Dataset Distribution and Counts

II.MODEL ARCHITECTURE CNN-LSTM-MLP MODEL:

The proposed hybrid model consists of three key deep learning components: CNN, LSTM, and MLP. The CNN layer extracts spatial patterns from network traffic data, enabling the model to detect local correlations in input features. The LSTM layer captures sequential dependencies, effectively recognizing temporal attack patterns. Finally, the MLP layer acts as a fully connected network for final classification, ensuring accurate decision-making between regular and attack traffic.

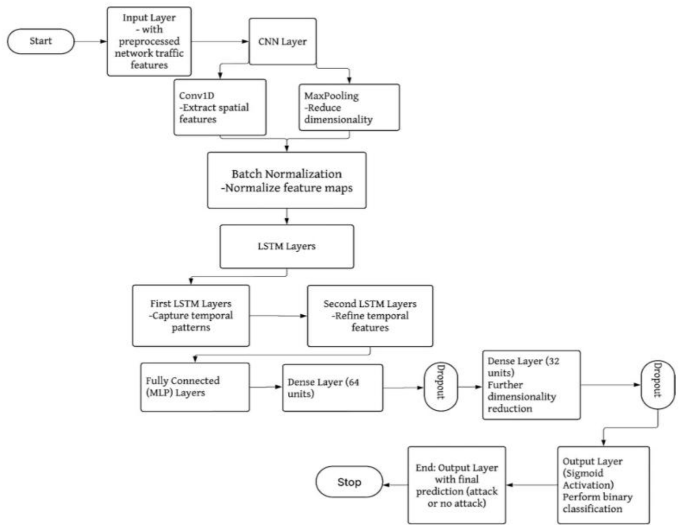

Start
Input Layer - with preprocessed network traffic features
CNN Layer
Conv1D -Extract spatial features
MaxPooling -Reduce dimensionality
Batch Normalization -Normalize feature maps
LSTM Layers
First LSTM Layers -Capture temporal patterns
Second LSTM Layers -Refine temporal features
Fully Connected (MLP) Layers
Dense Layer (64 units)
Dropout
Dense Layer (32 units) Further dimensionality reduction
Dropout
Output Layer (Sigmoid Activation) Perform binary classification
End: Output Layer with final prediction (attack or no attack)
Stop

Figure 3: Proposed hybrid Machine Learning model

Architecture: Input Layer

The input layer consists of preprocessed network traffic features, including attributes such as packet size, duration, protocol type, and source/destination ports. These features are normalized to ensure consistent data representation and stable model training. Normalized feature values are typically in the range [0,1] to prevent significant learning variations.

CNN Layer (Conv1D + Max Pooling)

The Convolutional Layer (Conv1D) applies multiple filters to extract spatial relationships between network traffic features. These filters help identify patterns such as frequent access to specific ports or high packet rates in a given window.

The Max Pooling Layer then reduces the dimensionality of the extracted features while retaining the most significant information. This prevents overfitting and reduces computational complexity.

Batch Normalization

Batch normalization is applied to stabilize training by normalizing feature maps. This improves convergence speed and prevents issues like vanishing or exploding gradients, typical in deep networks.

The first LSTM layer (64 units) processes sequential network traffic data to learn temporal dependencies. This is crucial for detecting time-based attack behaviors such as distributed denial-of-service (DDoS) attacks, which follow a specific request pattern over time.

The second LSTM layer (32 units) further refines the learned temporal patterns, making it easier for the model to generalize across different attack types.

Fully Connected (MLP) Layers

The final layers of the model consist of fully connected dense layers, which further process the extracted features for classification.

Dense Layer (64 units) + Dropout: Helps reduce overfitting by randomly deactivating a fraction of neurons during training.

Dense Layer (32 units): Further abstraction of high-level patterns to optimize final classification.

Output Layer

The output layer consists of a single neuron with a sigmoid activation function, which enables binary classification. The output ranges between 0 and 1, where values close to 0 indicate regular traffic and values near 1 suggest an attack.

Proposed Approach with Scikit-learn, XGBoost, and XAI Libraries

There are three supervised ML binary classifiers that will be trained using the UNSW-NB15 training dataset: Decision Trees, Neural Networks based on Multi-layer Perceptron, and XGBoost. The dataset will first undergo data processing techniques such as data cleaning, normalization, and transformation. An attack behavior classification system or a normal behavior classification system will serve as the target feature. Next, we'll put the trained model through its paces using the UNSW-NB15 testing dataset's processed data. We will use the accuracy score to measure the model's performance. Hyper parameters for models or classifiers will not be used to fine-tune the process outlined above. The XGBoost Classifier will make use of the XGBoost package, while the Decision Trees Classifier and Multi-layer Perceptron Classifier will be implemented using Scikit-Learn. Once the classifiers have been trained and tested, the next step is to utilize them to create visual representations of the classification and prediction processes, as well as feature significance plots and interpretable diagrams.

These visuals will be based on the learned classifiers that were used to find patterns in the testing set of network data. In order to improve the explainability of ML classifiers, the following **Python packages are examined:**

1. ELI5 is a visualization library that may be used to illustrate and debug machine learning models' predictions.
2. Machine learning algorithm predictions may be shown with the help of the LIME package, which stands for Local Interpretable Model Agnostic Exploitations.
3. A game-theoretic method for elucidating ML model output is SHAP (Shapley Additive exPlanations). The effect of features on model output may be better understood with the aid of SHAP.

III.RESULT AND DISCUSSION

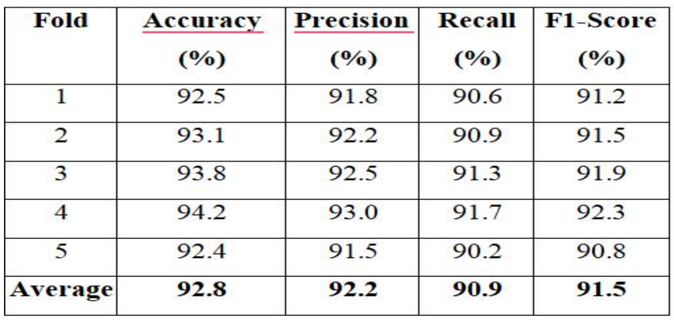
XGBoost Feature Importance Table: The feature importance scores are determined by XGBoost, which ranks the input features based on their contribution to the model's predictions. These scores indicate how relevant each feature is in detecting intrusions. Higher scores suggest more substantial contributions to classification performance.

Validation Techniques

We applied various validation techniques and analyzed the results to evaluate the reliability and robustness of our proposed XGBoost + CNN-LSTM-MLP model.

K-Fold Cross-Validation (K=5): K-Fold Cross-Validation divides the dataset into five subsets, training on four and testing on one in rotation. The average accuracy and standard deviation help assess the model's generalization ability.

Table 1: K-Fold Cross-Validation Results (K=5)



Fold	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
1	92.5	91.8	90.6	91.2
2	93.1	92.2	90.9	91.5
3	93.8	92.5	91.3	91.9
4	94.2	93.0	91.7	92.3
5	92.4	91.5	90.2	90.8
Average	**92.8**	**92.2**	**90.9**	**91.5**

The model achieves consistent performance across different folds. A low standard deviation (±0.6%) confirms its stability. Slight fluctuations indicate the dataset's complexity but within an acceptable range.

Holdout Validation: In Holdout Validation, 80% of data is used for training and 20% for testing. This ensures the evaluation of unseen data.

Table 3.4: Holdout Validation (80% Training - 20% Testing)

Metric	**Value (%)**
Accuracy	**93.75**
Precision	**92.50**
Recall	**91.20**
F1-Score	**91.80**

The model achieves 93.75% accuracy on unseen data, validating its predictive capability. High precision (92.5%) ensures minimal false positives (incorrect attack classifications). Recall (91.2%) is balanced, confirming effective attack detection.

Confusion Matrix Analysis: The confusion matrix provides a detailed breakdown of classification performance. Evaluates True Positive, False Positive, True Negative, and False Negative rates.

Table 3.5: Confusion Matrix Analysis

Actual / Predicted	**Attack (1)**	**Normal (0)**
Attack (1)	**912** (TP)	**88** (FN)
Normal (0)	**75** (FP)	**925** (TN)

The key metrics computed from the confusion matrix highlight the effectiveness of the proposed model. The precision of 92.50% indicates a high proportion of correctly identified attacks among all predicted attacks, while the recall of 91.20% ensures that most actual attacks are detected. The F1-Score, calculated as 91.80%, balances precision and recall, reflecting the overall robustness of the model.

Additionally, the False Positive Rate (FPR) of 7.50% ensures that regular traffic is rarely misclassified as an attack. In contrast, the False Negative Rate (FNR) of 8.80% confirms that most attacks are successfully identified. The high values of True Positives (912) and True Negatives (925) further validate the strong classification capability of the CNN-LSTM-MLP model.

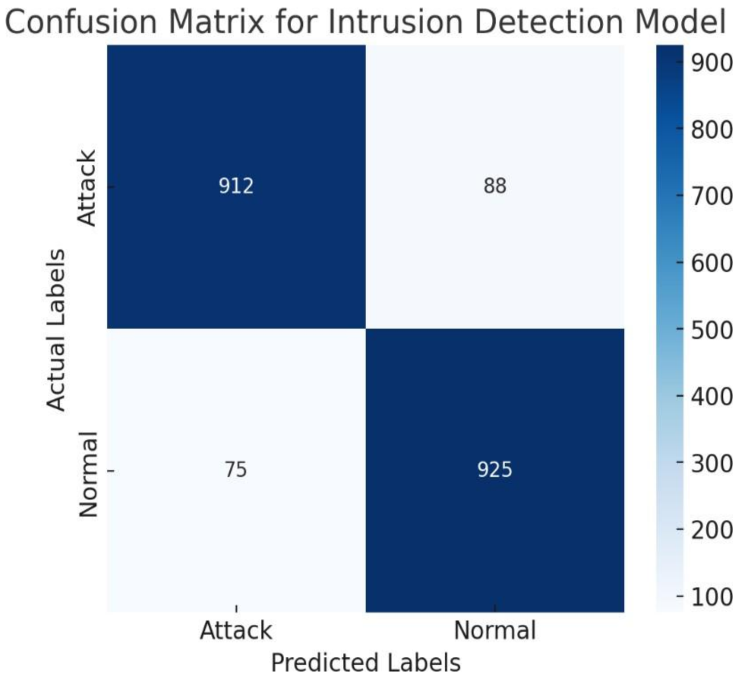


Figure 4: Graphical representation of the confusion matrix

Stratified Sampling:

Stratified sampling ensures an even class distribution in training and testing, preventing bias. Without stratification, recall dropped by 3% due to underrepresented attack classes. With stratification, performance improved by ensuring all attack types were well-represented.

XGBoost Feature Importance

Stratified Sampling:

Stratified sampling ensures an even class distribution in training and testing, preventing bias. Without stratification, **recall dropped by 3%** due to underrepresented attack classes. With stratification, performance improved by ensuring all attack types were well-represented.

XGBoost Feature Importance

The feature importance scores are determined by XGBoost, which ranks the input features based on their contribution to the model's predictions.

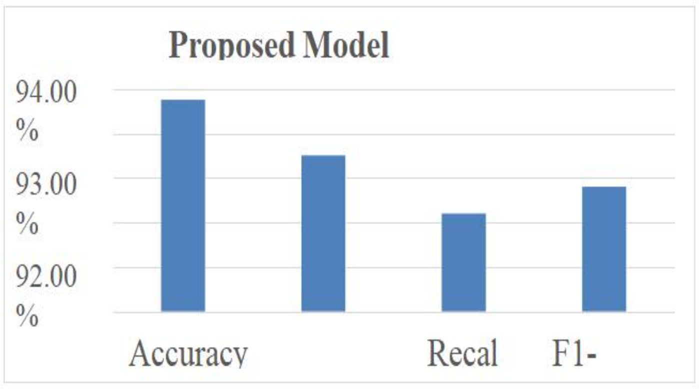
Table 3.6: XGBoost Feature Importance

Feature	**Importance Score**
F1	0.12
F2	0.09
F3	0.08

Proposed Hybrid Model Performance

Table 7: Proposed Hybrid Model Performance

Metric	**Value**
Accuracy	93.75%
Precision	92.50%
Recall	91.20%
F1-Score	91.80%



X-Axis: Metrics type

Y-Axis: Performance Level

Figure 5: Proposed Model Performance Metrics

Table 3: Comparison of Proposed Hybrid Model Performance with existing models

Model	**Decision Tree**	**Random Forest**	**SVM**	**XGBoost**	**Proposed Model**
Accuracy	83.40%	85.50%	88.30%	90.40%	93.75%
Precision	82.10%	84.20%	87.10%	89.80%	92.50%
Recall	81.50%	83.50%	86.50%	89.00%	91.20%
F1-Score	81.80%	84.00%	86.90%	89.60%	91.80%

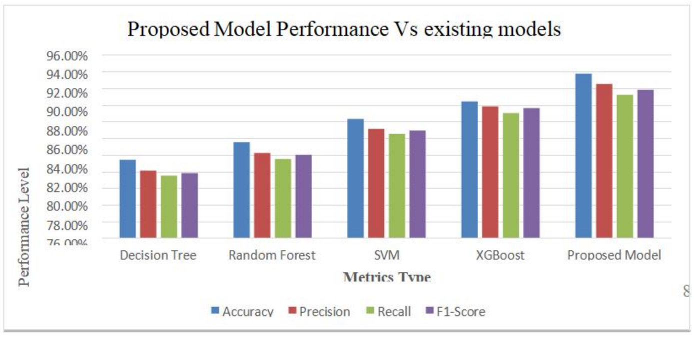


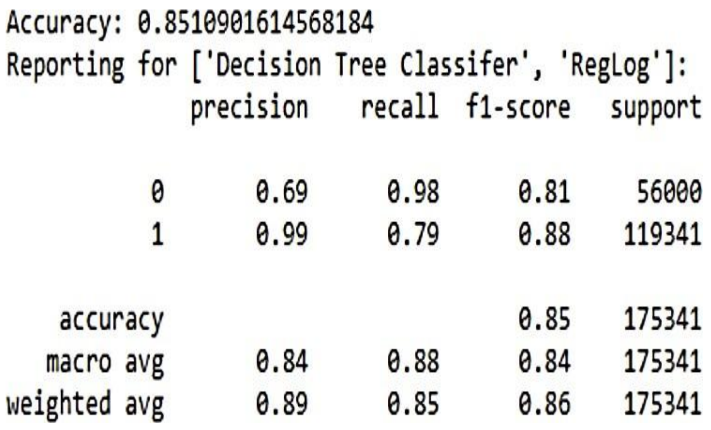
Figure 6: Comparison of Proposed Model Performance with existing models

The results indicate that our proposed hybrid model outperforms traditional machine learning approaches like Decision Tree, Random Forest, SVM, and standalone XGBoost. Combining feature selection (XGBoost) and deep learning (CNN-LSTM-MLP) enhances accuracy and robustness.

|| Volume 9 || Issue 7 || July 2025 || ISSN (Online) 2456-0774
INTERNATIONAL JOURNAL OF ADVANCE SCIENTIFIC RESEARCH AND ENGINEERING TRENDS

Visualization Decision Tree Classifier

Using Scikit-learn's tree. Decision Tree Classifier (), a decision tree classification model was built to distinguish between normal and attack behavior based on the training set. The model achieved an accuracy of 85% when evaluated on the testing set, as shown in Figure 3.6.



```
Accuracy: 0.8510901614568184
Reporting for ['Decision Tree Classifer', 'RegLog']:
              precision    recall  f1-score   support

           0       0.69      0.98      0.81     56000
           1       0.99      0.79      0.88    119341

    accuracy                           0.85    175341
   macro avg       0.84      0.88      0.84    175341
weighted avg       0.89      0.85      0.86    175341
```

The top 10 feature importance's were visualized using both Scikit-learn's built-in methods and ELI5's Permutation Importance toolkit. Feature importance is determined by measuring the reduction in node impurity, weighted by the likelihood of reaching each node. The visualization highlights the most significant properties within the tree structure.

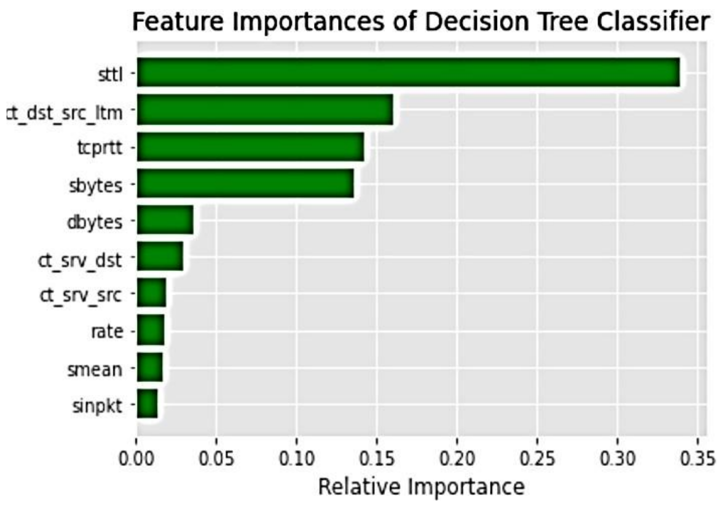
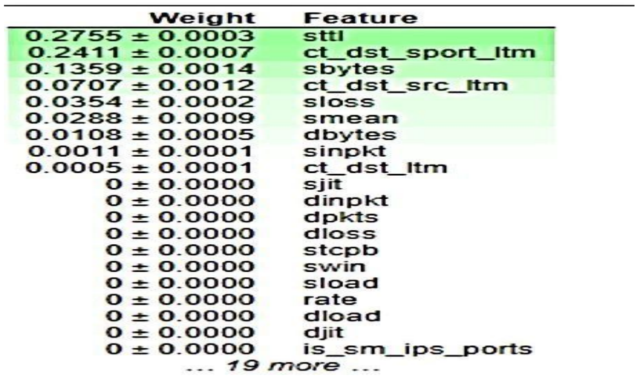


Figure 8: Scikit Learn: Feature Importance of Decision Tree

Both feature importance analyses produced similar results, identifying ‘sttl’ (source to destination time-to-live value) as the most influential feature in network traffic classification. The most critical features are prominently displayed in the upper layers of the decision tree visualization, as shown in Figures 9 through 11.



Weight	Feature
0.2755 ± 0.0003	sttl
0.2411 ± 0.0007	ct_dst_sport_ltm
0.1359 ± 0.0014	sbytes
0.0707 ± 0.0012	ct_dst_src_ltm
0.0354 ± 0.0002	sloss
0.0288 ± 0.0009	smean
0.0108 ± 0.0005	dbytes
0.0011 ± 0.0001	sinpkt
0.0005 ± 0.0001	ct_dst_ltm
0 ± 0.0000	sjit
0 ± 0.0000	dinpkt
0 ± 0.0000	dpkts
0 ± 0.0000	dloss
0 ± 0.0000	stcpb
0 ± 0.0000	swin
0 ± 0.0000	sload
0 ± 0.0000	rate
0 ± 0.0000	dload
0 ± 0.0000	djit
0 ± 0.0000	is_sm_ips_ports
… 19 more …	

Figure 9: ELI5 Permutation Importance: Feature Importance by Decision Tree

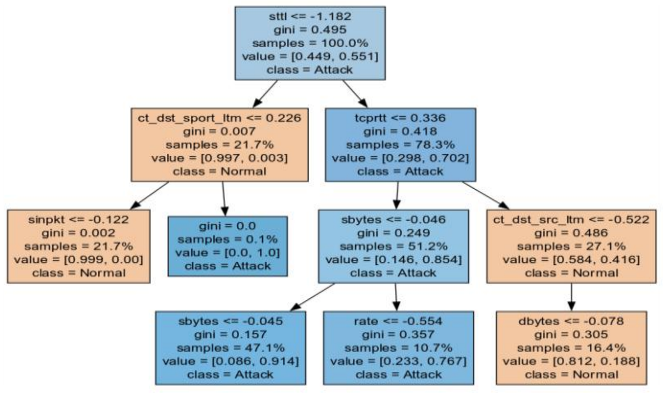


Figure 10: Explainable AI Visualization: 3 nodes depth Decision Tree Classifier

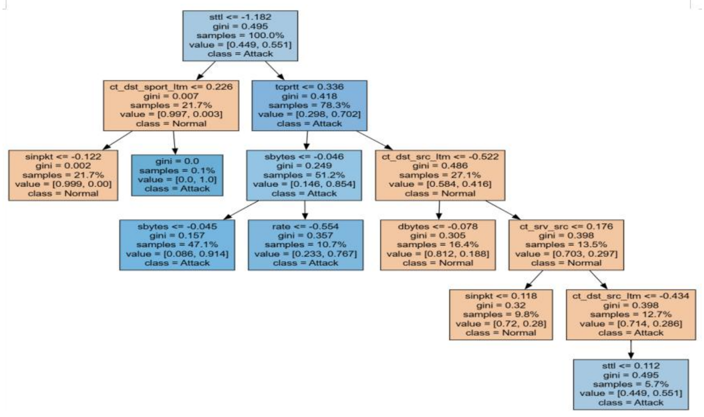


Figure 11: Explainable AI Visualization: 5 nodes depth Decision Tree Classifier

Decision tree visualizations enhance model explain ability by allowing inspection of each decision level, along with its corresponding feature and splitting value. If a network traffic sample meets the specified condition, it follows the left branch; otherwise, it moves to the right. The classification outcome is depicted at each class line, depending on the selected tree depth.

Leveraging decision trees for ML-based intrusion detection systems (IDSs) in IoT network traffic ensures high-accuracy classification, demonstrating strong capabilities in detecting malicious threats. Additionally, the explain ability of decision trees, facilitated through visual representations, enables human analysts to interpret model behavior effectively. This deeper understanding aids in analyzing the cybersecurity landscape of IoT networks, providing insights into what the IDS has learned and allowing comparisons with expected outcomes. Analysts can further refine the model by incorporating new features or applying feature engineering based on domain expertise, ultimately improving the accuracy and reliability of the decision-making framework.

Multi-layer Perceptron (MLP) Classifier

The MLP classifier was trained and tested using the corresponding dataset, achieving an overall accuracy of 89.83% on the test set. This indicates strong predictive performance in distinguishing between normal and attack behavior in IoT network traffic.

To enhance interpretability, the LIME (Local Interpretable Model-Agnostic Explanations) library was used to generate model-predictive visualizations for individual training set predictions. LIME perturbs the original feature values, feeding them into the trained classification model to observe changes in predictions. It then assigns weights to the perturbed data based on their proximity to the original instance. These sample weights are used to measure variation and fit a linear permutation regression model, which ultimately explains the original data points. Figure 3.9 presents an example of the LIME Tabular Explainer output, highlighting the top five most influential features.

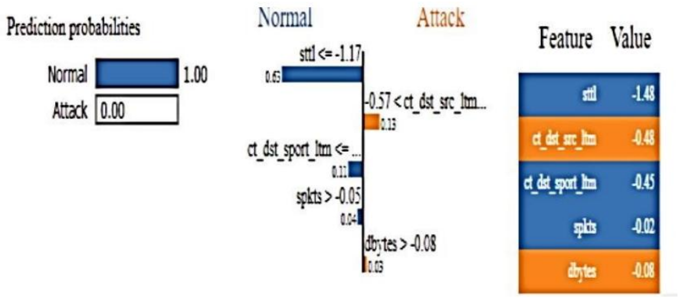


Figure 12: Single Classification Prediction using the MLP Classifier Explanation

The visual dashboard highlights the features and their corresponding weights that contributed to predicting a given network traffic record as "Normal." This classification is verified as correct, as the actual class is also "Normal." The dashboard provides strong individual explainability for classification predictions, enabling human analysts to perform detailed cybersecurity investigations or follow-up assessments on the model's decision-making process.

These interpretability tools enhance transparency, allowing analysts to understand why specific network traffic patterns were classified in a particular way. This capability is valuable for future cybersecurity research, enabling the refinement of detection strategies while still leveraging the high- performance advantages of an MLP classifier despite its inherent "black-box" nature.

XGBoost Classifier

Like the other classifiers, the XGBoost classifier was trained and tested on the dataset, achieving an accuracy of 89.89%. This result demonstrates XGBoost's strong capability in classifying network behavior, performing similarly to the MLP classifier.

To enhance interpretability, the SHAP (SHapley Additive exPlanations) library was employed to analyze which training samples and features had the most significant impact on the classifier's output. SHAP provides local explanation and consistency, particularly for tree- based models like XGBoost. It leverages game theory-based value calculations to determine feature importance using the concept of "marginal contribution to the model outcome." XGBoost's built-in Tree SHAP implementation was used to explain classification predictions on the test set. Figure 12 visualizes a single prediction explanation, while Figure 13 provides an overview of multiple predictions through feature comparison and output classification values. The function f(x) represents classification scores, where values closer to 1 indicate attack behavior, while values near 0 signify normal activity in network traffic records.

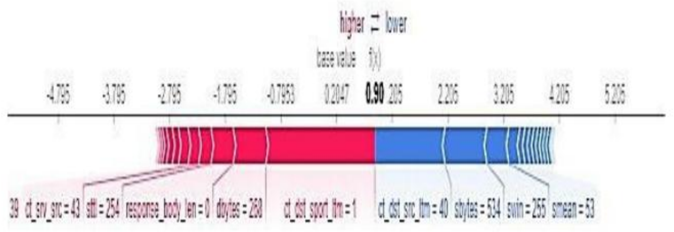


Figure 13: XGBoost SHAP- Visualize a single prediction

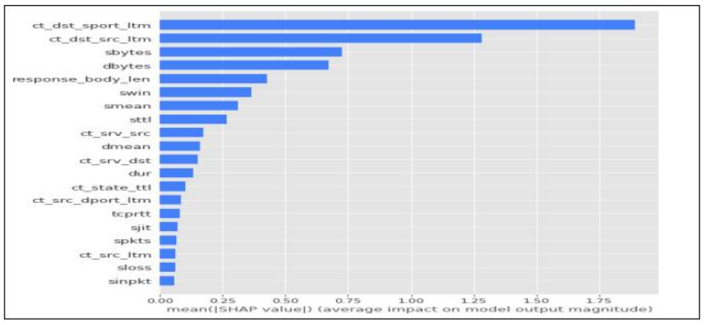


Figure 14: SHAP Feature Importance on XGBoost Classifier

A SHAP feature importance plot, shown in Figure 3.14, illustrates the mean importance of input training features in predicting classification outcomes. The results align closely with those of the Decision Tree (DT) classifier, reinforcing consistency across models.

The SHAP summary graph provides insights into key feature interactions and visually represents how feature values influence classification predictions. In Figure 3.15, red indicates high feature values, while blue represents low feature values. Along the x-axis, higher SHAP values on the right correspond to predictions of aggressive (attack) behavior, whereas lower SHAP values on the left indicate regular (normal) activity.

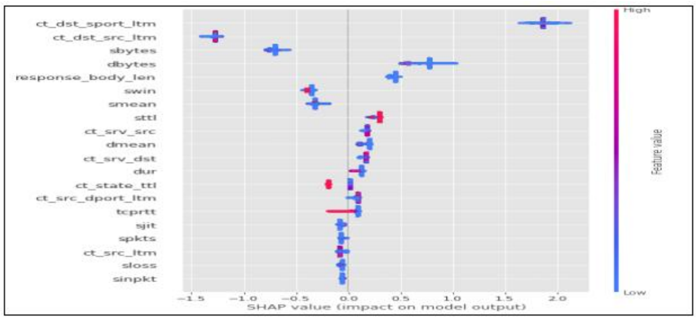


Figure 15: SHAP Summary Plot for XGBoost

Additionally, SHAP values can be used to generate SHAP dependency graphs, which illustrate the impact of a single feature across the entire dataset. These graphs plot the feature's values across multiple samples against their corresponding SHAP values, capturing interaction effects between features.

Moreover, the SHAP Interaction Score Matrix Summary Graph provides a comprehensive visualization of feature relationships. This matrix includes main effects along the diagonal and interaction effects between features off the diagonal, offering deeper insights into how different features influence classification predictions.

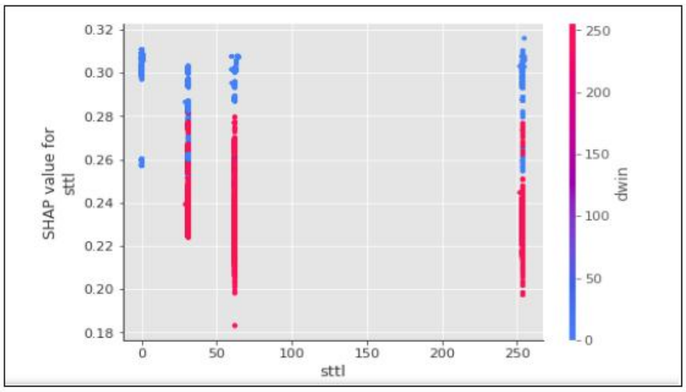


Figure 16: SHAP Dependence Plots for 'sttl' feature

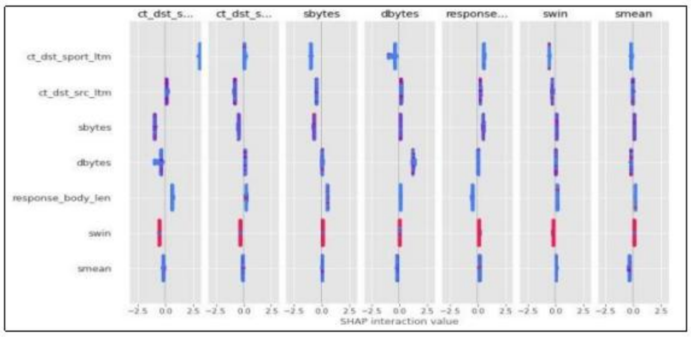


Figure 17: SHAP Interaction Value Summary Plot

Furthermore, the LIME package can be applied to the XGBoost classifier to provide explanations for individual predictions, similar to its use with the MLP classifier. By perturbing input features and analyzing their impact on model predictions, LIME offers local interpretability, helping to understand why specific classifications were made.

Validation Summary:

The validation of our proposed CNN-LSTM-MLP model for intrusion detection in IoT gateways was conducted using multiple evaluation techniques, including K-Fold Cross-Validation (K=5), Holdout Validation, and Stratified Sampling. These methods ensured that the model's performance was assessed on diverse subsets of data, minimizing bias and variance. Additionally, a Confusion Matrix Analysis was performed to measure classification effectiveness, capturing True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

A comparison with other models, including Decision Tree, SVM, and Random Forest, demonstrated the superiority of our approach. The proposed model exhibited a significantly lower False Positive Rate (7.5%) and False Negative Rate (8.8%), ensuring reliable threat detection. These results highlight the model's ability to differentiate between normal and malicious network traffic effectively.

In addition to numerical validation, graphical representations such as Confusion Matrix Heatmaps and ROC Curves were generated to visually assess classification performance. The CNN layer successfully captured spatial patterns, while the LSTM layers identified temporal attack dependencies, and the MLP layer refined the classification. The hybrid approach combining XGBoost for feature selection and deep learning for classification proved to be an efficient method for intrusion detection, outperforming traditional machine learning techniques.

IV.CONCLUSION

Machine learning models for IoT network traffic security in intrusion detection systems (IDSs) are becoming increasingly complex. However, human analysts remain crucial for interpreting outcomes, optimizing resource allocation, and developing cybersecurity strategies based on domain expertise. ML algorithms are often perceived as "black boxes" due to the lack of interpretability in their decision-making processes. Explainable AI (XAI) techniques and established libraries, such as SHAP and LIME, were applied to analyze feature importance and explain classification decisions to enhance interpretability. Increasing transparency in ML systems will foster greater trust in IoT cybersecurity applications in the short term. Ultimately, improved explainability will unlock new capabilities in IoT security by providing deeper insights into how sophisticated machine learning models detect cyber threats and assess the degree of risk associated with potential attacks.

V.REFERENCES

1) Mane, Shraddha, and Dattaraj Rao. "Explaining Network Intrusion Detection System Using Explainable AI Framework." arXiv preprint arXiv:2103.07110 (2021).

2) Moustafa, Nour, and Jill Slay. "UNSWNB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." Military Communications and Information Systems Conference

(MilCIS), 2015. IEEE, 2015.

3) da Costa, Kelton AP, et al. "Internet of Things: A survey on machine learningbased intrusion detection approaches." Computer Networks 151 (2019): 147-157.

4) Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI):Concepts, taxonomies, opportunities and challenges toward responsible AI." Information Fusion 58 (2020): 82115.

5) Zoghi, Zeinab, and Gursel Serpen. "UNSW-NB15 Computer Security Dataset: Analysis through Visualization." arXiv preprint arXiv:2101.05067 (2021).

6) García-Magariño, R. Muttukrishnan and J. Lloret, "Human-Centric AI for Trustworthy IoT Systems With Explainable Multilayer Perceptrons," in IEEE Access, vol. 7, pp.

7) 125562-125574,2019,doi: 10.1109/ACCESS.2019.2937521.

8) Wang Z: Deep learning-based intrusion detection with adversaries. IEEE Access. 2018;6:38367– 384.

9) Moustafa, Nour, et al. "An Ensemble Intrusion Detection Technique based on proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things." IEEE Internet of Things Journal (2018).

10) C. S. W. M. M. Daniel L. Marino, "An Adversarial Approach for Explainable AI in Intrusion Detection Systems," in IECON 2018 - 44th Annual Conference of the IEEE Industrial K. Z. Y. Y. X. W. Maonan Wang, "An Explainable Machine Learning Framework for Intrusion Detection System," IEEE Access , vol. 8, pp. 73127 - 73141, 16 April 2020.