Network Intrusion Detection System

Jaswanth Sirigiri
College of Engineering
Texas A&M University-Corpus Christi

Koushik Reddy Kambham
College of Engineering
Texas A&M University-Corpus Christi

Sai Avinash Vagicherla
College of Engineering
Texas A&M University-Corpus Christi

Raghuvamsi Mallampalli
College of Engineering
Texas A&M University-Corpus Christi

***Abstract* – The escalating presence of Internet of Things (IoT) networks has heightened the demand for IDSs (intrusion detection systems) in computer networks. As traditional cybersecurity approaches falter in IoT contexts, IDSs increasingly use machine learning (ML) techniques for efficacy. While machine learning-based intrusion detection systems (IDSs) are a powerful tool for identifying network threats, their perceived complexity and lack of interpretability make them difficult to use. Leveraging the UNSW-NB15 dataset, three ML classifiers Decision Trees, XGBoost, and Multi-Layer Perceptrons were trained to develop a network forensic system capable of monitoring suspicious botnet activities. Subsequent adoption of Explainable AI (XAI) techniques enhanced the interpretability of classification predictions, suggesting the feasibility and value of integrating XAI with traditional ML systems for cybersecurity applications.**

Keywords – UNSW-NB15, NIMS botnet, Naive Bayes, Explainable Artificial Intelligence (XAI), XGBoost, Multi-layer Perceptron Neural Network, ELI5, LIME, SHAP, black boxes, Decision Tree

I. INTRODUCTION

Because IoT networks store so much valuable user data, they are becoming increasingly attractive targets for malicious attacks. This holds particular significance for vital services, as cyberattacks often aim to undermine the fundamental security principles of availability, integrity, and confidentiality. As a result, machine learning (ML) models and algorithms have been used in intrusion detection systems (IDS) to monitor and detect patterns of cyberattacks in networking environments. These systems monitor network traffic for unusual activity and send out alerts when they find possible points of attack. IoT structures and behaviors, however, may not be directly amenable to conventional network security solutions due to their distinct features, such as low computational power and low operating energy. Furthermore, because there isn't a single standard for IoT architecture, policies, and connectivity domains, traditional security measures like authentication and encryption protocols encounter difficulties in IoT environments.

Even though current machine learning (ML) algorithms have the potential to learn IoT network inputs linked to both benign and malicious behavior, ML-based intrusion detection systems (IDSs) are still frequently viewed as "black boxes," with no clear explanation for their classifications or predictions. Professionals in cybersecurity may find it difficult to assess and distribute resources efficiently due to this opacity. Acknowledging the growing need for strong cybersecurity protocols, this work intends to examine machine learning (ML) techniques and investigate Explainable AI (XAI) approaches using Python libraries to clarify feature significance, prediction reasoning, and model classification decisions. These tools have the potential to facilitate human analysts' understanding of ML-based IDS systems by improving transparency and understandability, especially in IoT cybersecurity domains. Also, performance measures like precision will be assessed to support explainability initiatives.

II. RELATED WORK

With the growing integration of Internet of Things (IoT) technology into a variety of applications, including home automation, smart cities, healthcare systems, and manufacturing, cybersecurity professionals are increasingly focusing on the protection of IoT networks. Furthermore, there has been a small but notable amount of focus on the creation and application of intrusion detection systems (IDSs) based on statistical feature learning algorithms and machine learning techniques. To protect IoT network traffic, Nour Mustafa's study, for example, presented an ensemble-based machine learning approach for the detection of intrusion that makes use of suggested statistical flow features. The system derived its statistical flow features by initially scrutinizing the network's attributes and subsequently implementing an AdaBoost ensemble learning technique across three machine learning algorithms: ANN, decision trees, and Naive Bayes (NB). These models efficacy in identifying malicious events was assessed using datasets that mimicked Internet of Things sensor data, such as UNSW-NB15 and NIMS botnet. The results of the experiment showed a high detection rate for both benign and malevolent behavior. When compared to conventional IoT cybersecurity techniques, the suggested ensemble technique demonstrated better detection capabilities overall, as well as a lower false positive rate. Furthermore, a survey carried out by Kelton da Costa examined different machine-learning approaches used in intrusion detection for computer network security and the Internet of Things (IoT). The examination encompassed more than 95 publications addressing diverse subfields within security concerns and machine learning in Internet of Things settings.

About the literature on Explainable Artificial Intelligence (XAI), a lengthy work offers a thorough synopsis of the basic ideas, classifications, prospects, and difficulties associated with responsible AI; it is an invaluable tool for analyzing studies aimed at improving the explainability of machine learning techniques. Due in large part to advanced machine learning techniques like ensembles and Deep Neural Networks' intrinsic lack of explainability, XAI has become increasingly popular recently. As a result, XAI has become essential for ML model deployment, with an emphasis on upholding accountability, transparency, justice, and explainability. Notably, Shraddha Mane and Dattaraj Rao carried out the initial research into creating Explainable AI Frameworks for Network Intrusion Detection Systems. While machine learning (ML) models increase accuracy, their complexity also increases, making them harder to interpret. In their study, they developed a deep neural network aimed at detecting network intrusions. Additionally, they proposed an interpretable AI framework to ensure

transparency across the entirety of the machine learning procedure. They successfully improved model transparency by applying existing XAI algorithms, such as BRCG, LIME, SHAP, CEM, and Protidic to the NSL-KDD dataset to provide explanations for individual predictions.

III. DESIGN

UNSW-NB15 Dataset

The UNSW-NB15 dataset consists of Internet of Things (IoT) network traffic data with discrete categories that distinguish between legitimate activity and malicious attack behaviors carried out by botnets. There are many different kinds of these attacks, such as reconnaissance, backdoors, worms, fuzzers, DoS, and shellcode. The raw network packets of the UNSW-NB15 dataset, sourced from the Cyber Range Lab of the ACCS (Australian Centre for Cyber Security), were created using the IXIA PerfectStorm tool. With the aid of this tool, a hybrid dataset containing both realistic, up-to-date, everyday activities and simulated attack behaviors typical of contemporary IoT networks could be created. Figure 1 illustrates the configuration of the testbed dataset and the feature creation process utilized in the UNSW-NB15 dataset.

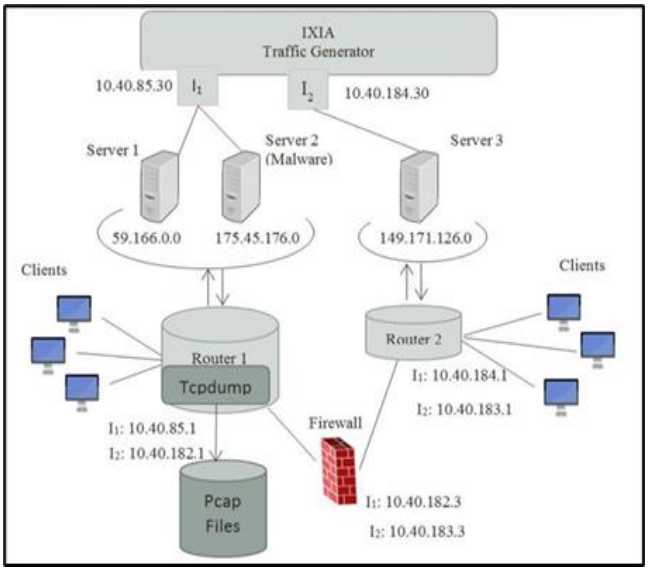

Figure 1: IXIA Traffic Generator Overview

The creators of UNSW-NB15 have split the data into two separate sets: the training set (UNSW_NB15_training-set.csv) and the testing set (UNSW_NB15_testing-set.csv). The training set comprises 175,341 records, and the testing set comprises 82,332 records. Each record in both sets includes a target response indicating whether the traffic behavior is categorized as an attack or normal. The UNSW-NB15_features.csv file includes descriptions for each of the dataset's 39 numerical features.

The objective of the experimental procedures is to develop a binary classification as the target variable that can differentiate between Attack and Normal behaviors. The details and distribution of values for each attack class within the data subsets are shown in Figure 2, where 1 denotes attack behavior and 0 represents normal behavior. The binary response variable linked to activity behavior in the dataset exhibits a well-maintained balance.

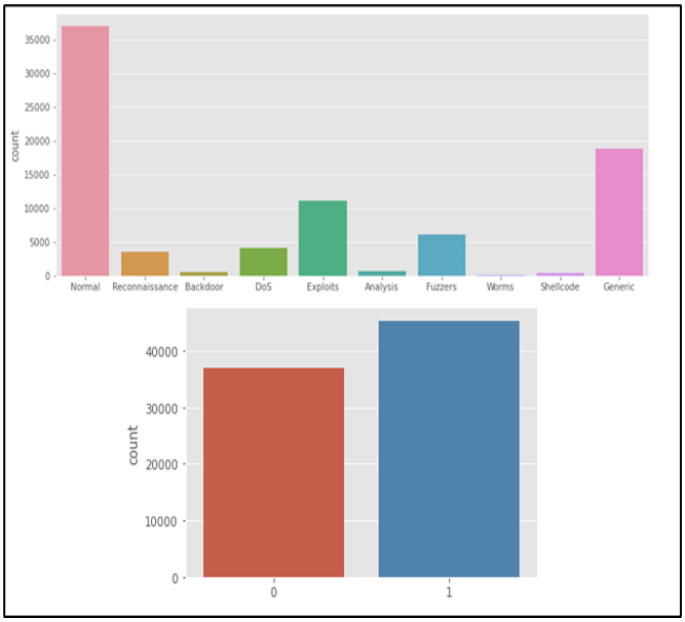

Figure 2: Distribution and Counts of Training Dataset

Overview of ML Methods

To build binary classification classifiers, three supervised machine learning techniques were selected: Multi-layer Perceptron Neural Network, XGBoost, and Decision Trees. The explainability (XAI) of these machine learning algorithms varies, and they are arranged in descending order.

Decision Tree Classifier

To classify Attack or Normal behavior using the 39-input feature, the supervised machine learning algorithm known as the decision tree (DT) classifier will be utilized. With this algorithm, decision points are represented by nodes in a framework for decision-making that resembles a tree. One can preset the decision tree's maximum depth. Explainability is a feature of decision trees by design, as seen in the visuals of the final trees.

Multi-layer Perceptron Classifier

Artificial neural networks called Multi-Layer Perceptron (MLP) classifiers use single-neuron models or perceptrons to solve challenging predictive modeling problems. As a neural network, the MLP classifier possesses an inherent capability to discern patterns within the training data and establish the most effective connections with the desired output variable. Essentially, neural networks learn a mapping function. Artificial neurons serve as the fundamental components of neural networks, processing weighted input signals and employing an activation function to generate an output signal. This activation function incorporates both the activation threshold and the output signal strength, mapping the sum of the weighted inputs to the neuron's output. Within a network, neurons are arranged into layers, and the initial weights are essentially educated guesses. The most popular neural network training algorithms are gradient descent or back-propagation, which are frequently combined with a sigmoid function. To produce an output value, the network processes inputs upward, turning on neurons along the way. Any differences between this output and the expected output are compared, and an error calculation is made. The weights are subsequently adjusted based on their respective impact on the error, as this error is propagated backward through the

network, layer by layer. Every example in the training set goes through this iterative process again, with the weights being gradually adjusted until they converge to a local optimum.

In general, neural networks like MLP Classifiers do not provide sufficient interpretability and explainability for models. Neural networks, which consist of visible and hidden layers of neural units, are frequently referred to as "black-box" algorithms because of the interactions and opacity of the hidden layers after training.

XGBoost Classifier

A well-known gradient-boosted decision tree implementation that is quick and effective is called XGBoost. "eXtreme Gradient Boosting," or XGBoost, is the name of an open-source software library that has been diligently created to optimize memory consumption and processing speed. It is a popular option because of its design architecture, which guarantees the best possible distribution of resources for model training. To put it simply, the gradient-boosting decision tree algorithm is implemented in the XGBoost library. Boosting, an ensemble technique, addresses errors in current models by incorporating new models. In gradient boosting, subsequent models are constructed to forecast residuals or discrepancies from preceding models, and their forecasts are aggregated to produce the ultimate prediction. Moreover, the gradient descent algorithm aims to minimize loss when integrating new models. This methodology forms the basis for classifying Attack or Normal behavior in predictive modeling.

Methodology Proposed Utilizing Scikit-learn, XGBoost, and Explainable AI (XAI) Libraries

The three Machine Learning classifiers Multi-layer Perceptron Neural Network, Decision Trees, and XGBoost will be trained on the UNSW-NB15 training dataset after data processing techniques like normalization, cleaning, and transformation have been applied. A binary classification that differentiates between Normal (0) and Attack (1) behaviors will be the target feature. The processed UNSW-NB15 testing dataset will then be used to test the trained model. The accuracy score of the model will be used to evaluate its performance. Hyperparameters for the model or classifier will not be adjusted during the previously described process. Scikit-Learn will be used to implement the Multi-layer Perceptron Classifier, and Decision Trees Classifier, and the XGBoost library will be used to implement the XGBoost Classifier.

After training and testing the classifiers, the next step involves employing the trained classifiers to identify network traffic patterns within the testing dataset. This process includes creating clear diagrams, plots showcasing feature importance, and visual aids to clarify the classification and prediction process. We explore various Python packages to adjust machine learning classifiers for enhanced interpretability.1. ELI5: A visualization library useful for explaining and debugging machine learning models.

2. LIME: An application package made to provide context for machine learning algorithms' predictions.

3. SHAP: An approach based on game theory that explains any machine learning model's output. SHAP facilitates a better comprehension of how features affect the model's output.

IV. IMPLEMENTATION

Decision Tree Classifier

utilizing the tree. Using the training set, a Decision Tree classification model was created to differentiate between Normal and Attack behavior using the DecisionTreeClassifier() function. When compared to the testing set, the model showed an accuracy of 85%, as shown in Figure 3.

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

Figure 3: Decision Tree Classifier Report

ELI5's Permutation Importance toolkit in conjunction with the scikit-learn library allowed for the visualization of the significance of the top 10 features. The decrease in node impurity, adjusted based on the probability of reaching that particular node, dictates the importance of a feature. As such, in the resulting tree-like visualization, the most important features will be positioned higher.

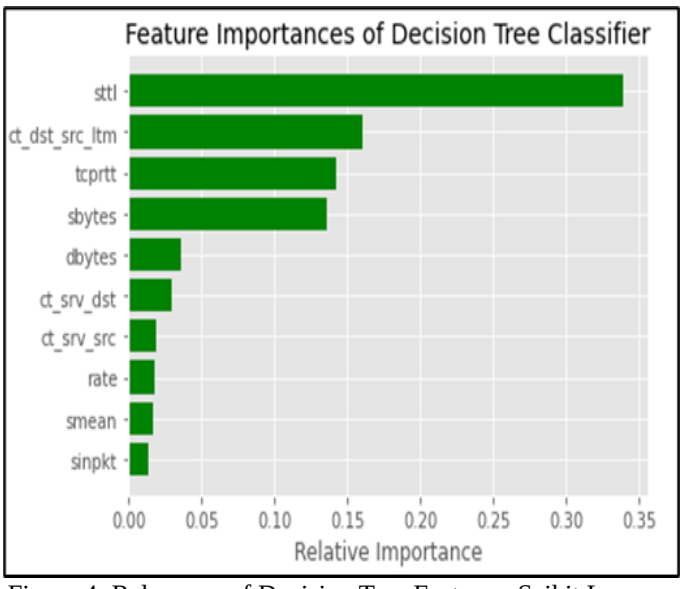

Figure 4: Relevance of Decision Tree Features: Scikit Learn

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 5: ELI5 Permutation Importance is a crucial decision tree feature.

The results of both feature importance analyses are remarkably consistent, indicating that the most influential feature in classification prediction for network traffic analysis is 'state', or source to destination time to live value. The decision tree visualizations shown in Figures 6 through 8 prominently feature these important features in their upper layers.

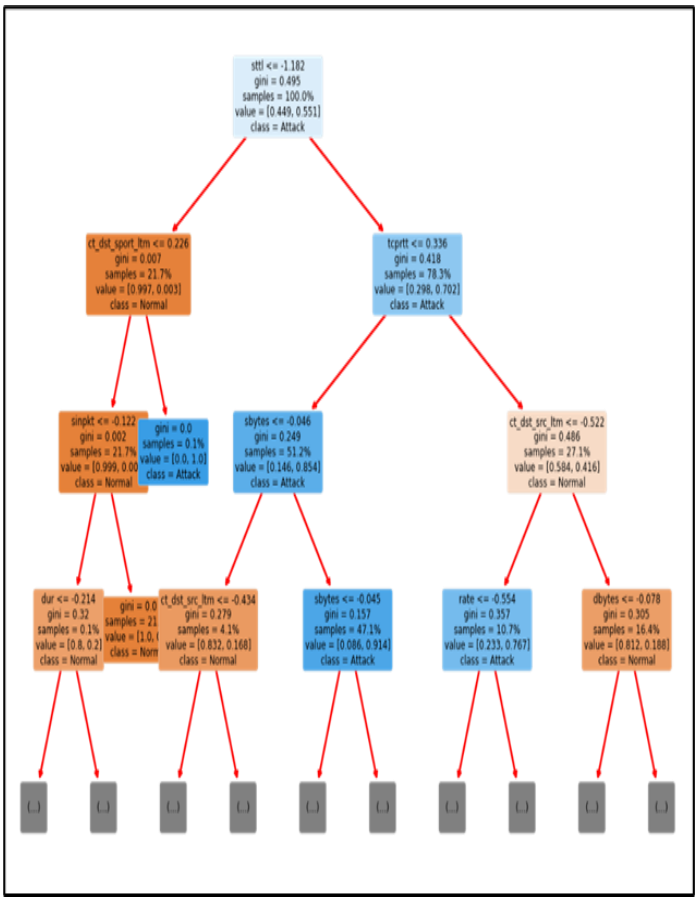

Figure 6: Classifier using Decision Trees (Depth = 3 Nodes) Visualization of Explainable AI

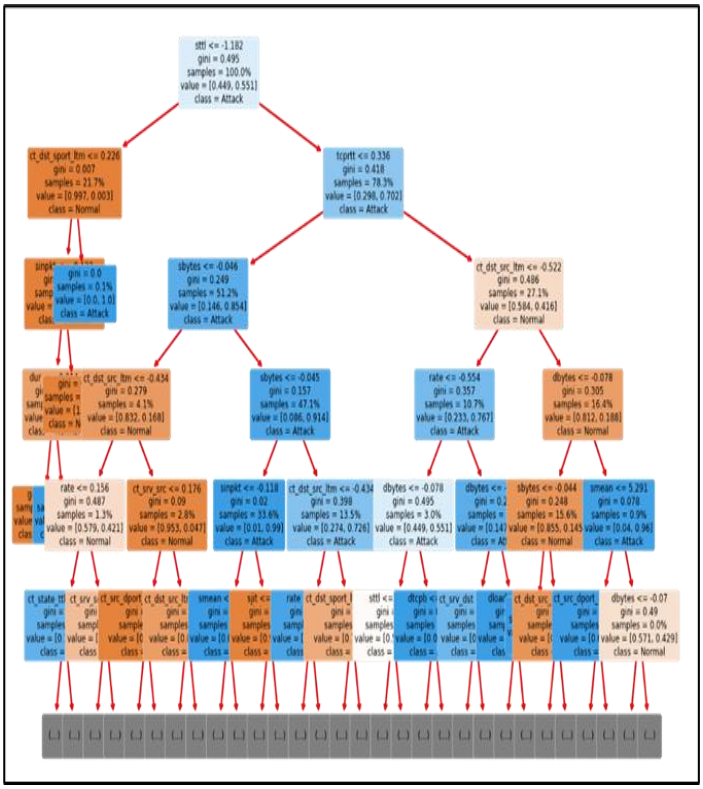

Figure 7: Classifier using Decision Trees (Depth = 5 Nodes) Visualization of Explainable AI

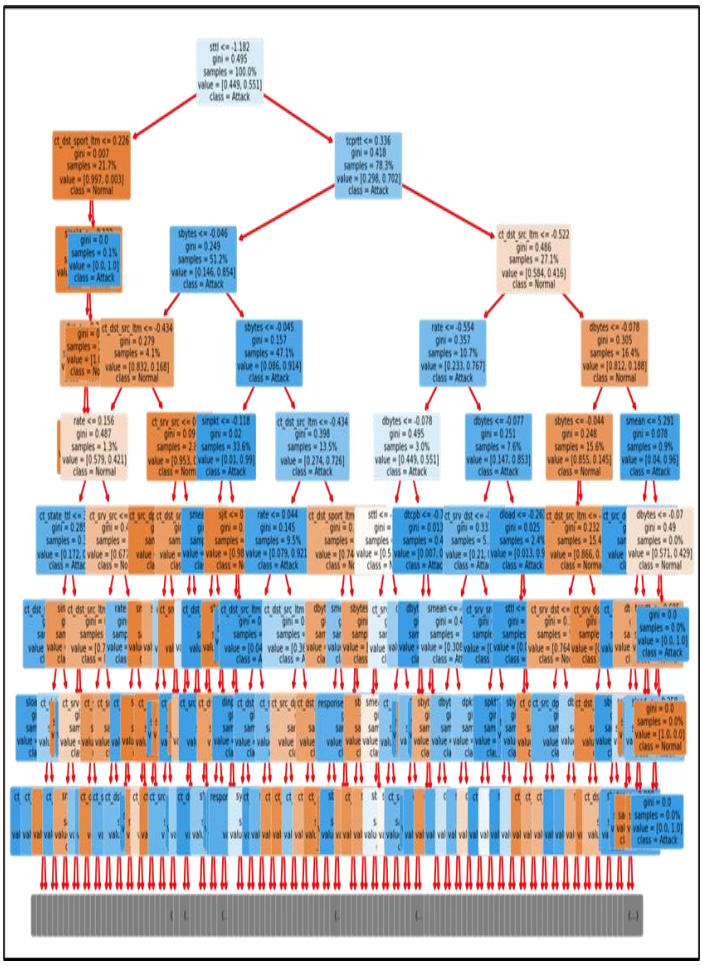

Figure 8: Classifier for Decision Trees (Depth = 8 Nodes) Visualization of Explainable AI

By examining each decision level, its corresponding feature, and the splitting value for each condition, the decision tree visualizations allow for the explainability of the model. A

sample of network traffic is routed to the left branch or node if it meets the condition; if not, it is routed to the right branch. Moreover, depending on the maximum depth of the selected tree, the classification prediction result is displayed within each class line. Machine learning-based Intrusion Detection Systems (IDSs) utilizing decision trees for IoT network traffic achieve a high level of accuracy in classification, showcasing reliable detection of malicious threats. Additionally, the interpretative functionalities embedded in the Decision Tree (DT) algorithm aid human analysts in comprehending the model, particularly when decision trees are presented visually. This promotes a better comprehension of the dynamics of cybersecurity in Internet of Things networks. This kind of understanding includes the capacity to compare expected results or conjecture about what the IDS machine has learned from the features. By adding new features or performing feature engineering based on domain knowledge, human analysts can also aid in the machine's learning process. This cooperative method greatly helps analysts assess the decision framework of the model for accuracy and improve it as needed.

Multi-layer Perceptron (MLP) Classifier

The corresponding datasets were used to train and assess the MLP classifier. With an overall accuracy of 89.83% against the testing set, the model demonstrated exceptional classification accuracy in the difference between Attack behavior and Normal behavior in IoT traffic. Individual predictions made by the MLP Classifier in the training set can be visually represented by utilizing the LIME library. LIME adjusts the original data by inputting the data features and predictions into an internal classification model and examining the resulting outputs. Next, based on how closely the new data outputs resemble the original point, weights are assigned to them.

Afterward, the library applies the sample weights to train a surrogate linear regression model on the adjusted dataset. Ultimately, the newly trained explanatory model can be used to clarify the original data points. An example of the Lime Tabular Explainer output, showcasing the top 5 features, is depicted in Figure 9.

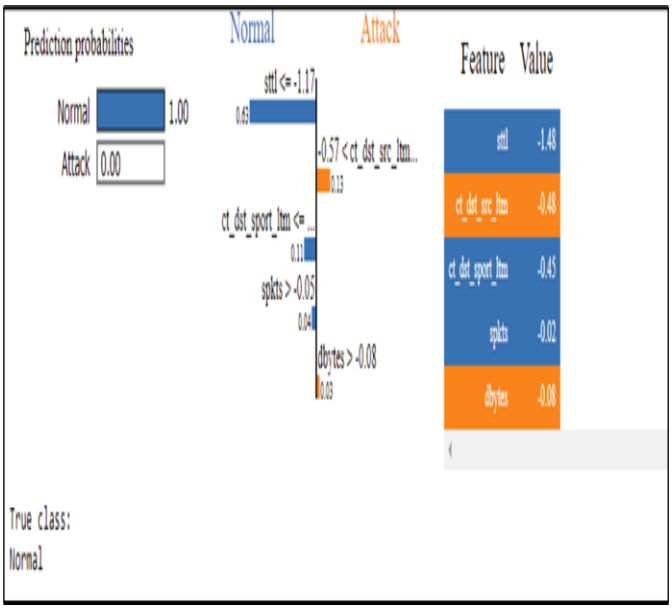

Figure 9: Prediction of Single Classification with the MLP Classifier Explanation

The features and their corresponding weights that went into predicting that specific network traffic record's overall behavior classification as "Normal" are displayed in the visual dashboard. Since 'Normal' is the true class, this classification has been confirmed to be accurate. The detailed individual explanations for the predicted classifications are provided by this visual dashboard.

Human analysts have the option to carry out in-depth cybersecurity research or additional analysis to determine the model's classification of a particular piece of network traffic. Even though neural net MLP classifiers are usually thought of as "black boxes" in terms of functionality, this tool improves prediction transparency, which can be utilized for future cybersecurity research.

XGBoost Classifier

For the classification task, the XGBoost Classifier underwent training and then assessed on the testing set, just like the other two classifiers. The model's overall accuracy of 89.89% demonstrated the XGBoost classifier's potent capacity to categorize network behavior. This performance is almost the same as the MLP Classifier's.

The SHAP library was utilized to take advantage of this classifier's explainability features. The training samples and characteristics that have the biggest impact on the model or classifier output can be analyzed thanks to this library. The primary benefits of SHAP include its ability to offer localized explanations and maintain consistency within tree-based model frameworks like XGBoost. SHAP relies on calculations from game theory and offers extensive feature importance by generating values that interpret outcomes from tree-based models through the concept of 'marginal contribution to the model outcome.'

The XGBoost implementation of Tree SHAP can elucidate the classification predictions of the testing set. Figure 10 visually explains a single prediction, while Figure 11 illustrates multiple predictions by comparing features or output classification values. The f(x) values assigned to a network traffic record signify its classification; proximity to 1 suggests attack behavior, while proximity to 0 suggests normal activity.

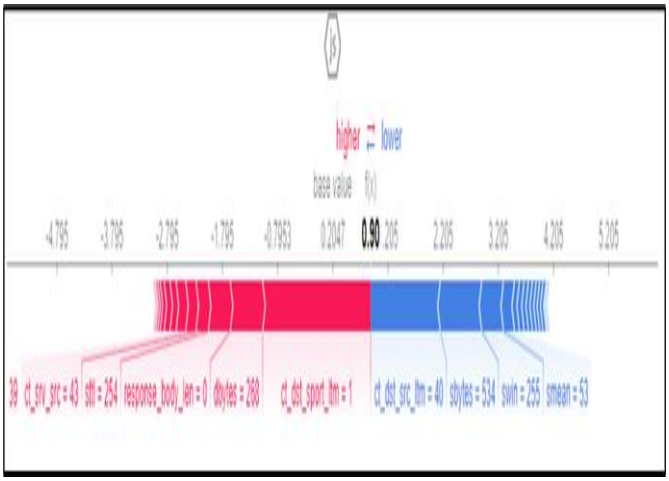

Figure 10: XGBoost SHAP: Display a solitary forecast

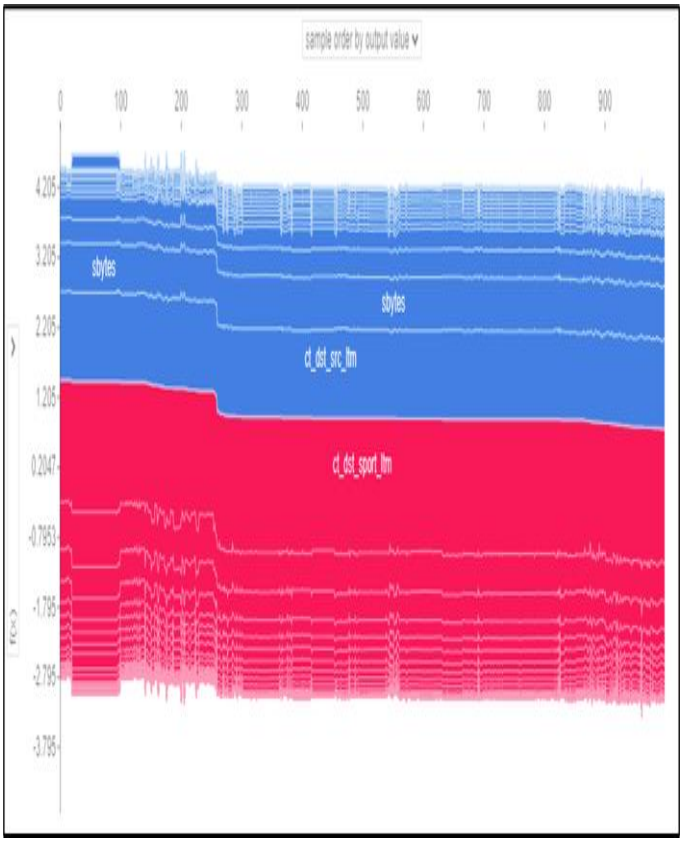

Figure 11: XGBoost SHAP: Visualize Numerous Forecasts

A feature importance plot produced by SHAP to evaluate the average significance of input training features in classification prediction is shown in Figure 12. The results bear a close resemblance to the ones derived from the DT Classifier.

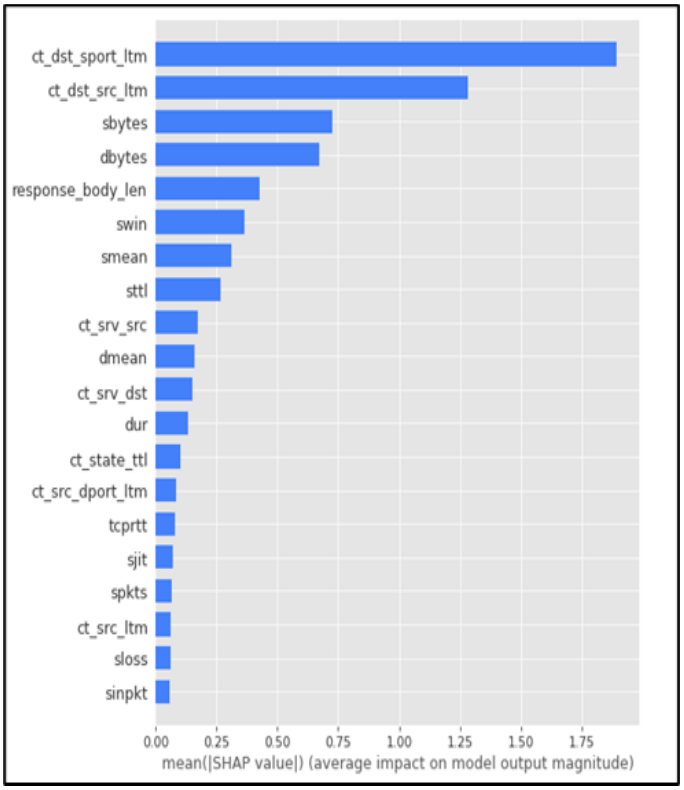

Figure 12: The Significance of SHAP Feature in XGBoost Classifier

Certainly, the SHAP summary plot visually displays the primary feature combinations and how they impact classification predictions. In Figure 13, Higher feature values are indicated by red, and lower feature values are indicated by blue. Lower SHAP values on the left indicate predictions of Normal Activity, while higher SHAP values on the right suggest predictions of Attack Behavior on the x-axis.

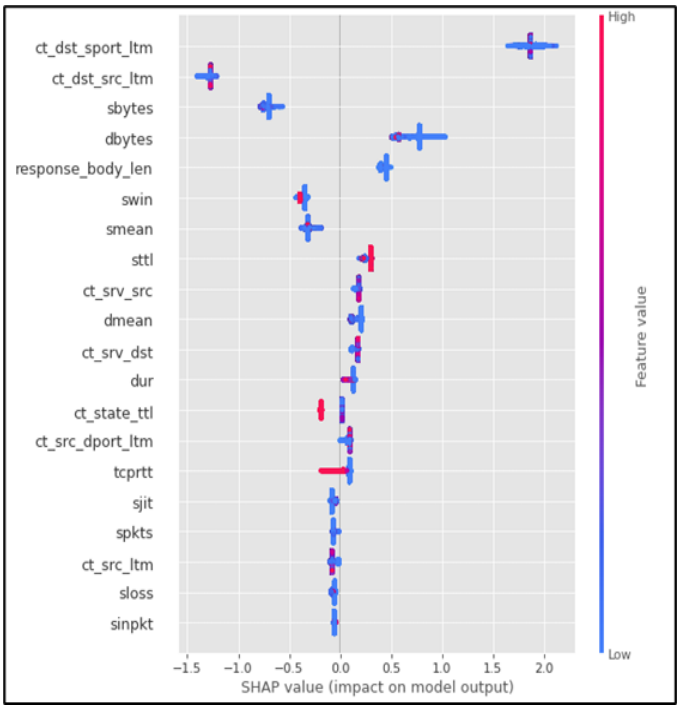

Figure 13: SHAP Summary Plot for XGBoost

Furthermore, SHAP values make it possible to create SHAP dependence plots, which show how one feature affects the entire dataset. These plots take into account any interaction effects that may be present in the features by showing the value of a feature against its SHAP value across multiple samples. A matrix of summary plots with primary effects on the diagonal and interaction effects off the diagonal is also displayed in a summary plot of a SHAP interaction value matrix.

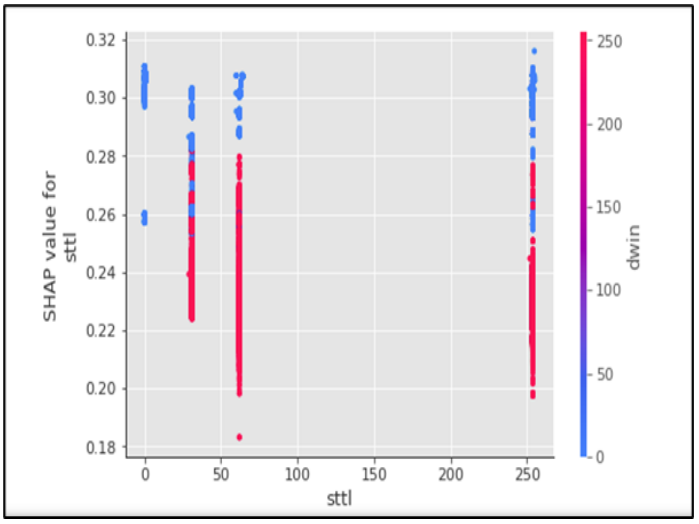

Figure 14: Plots of SHAP Dependency for the "state" feature

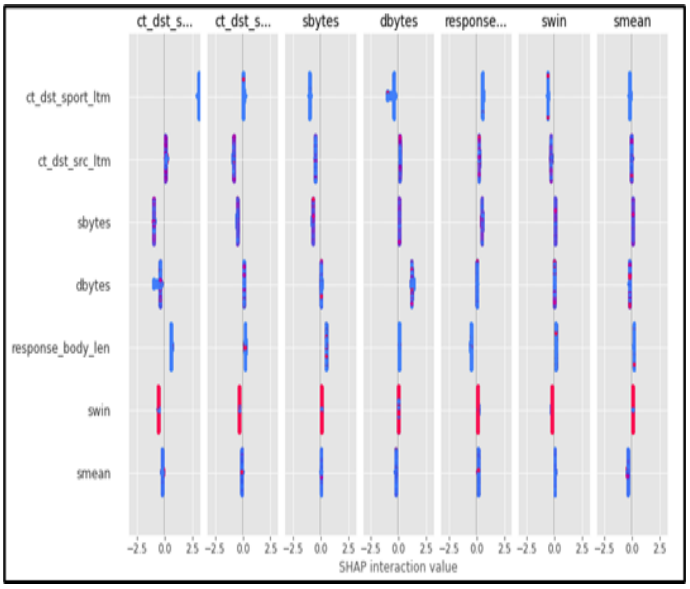

Figure 15: SHAP Interaction Value Summary Plot

Furthermore, the XGBoost Classifier's predictions can also be explained, much like the MLP Classifier's application of the LIME package does.

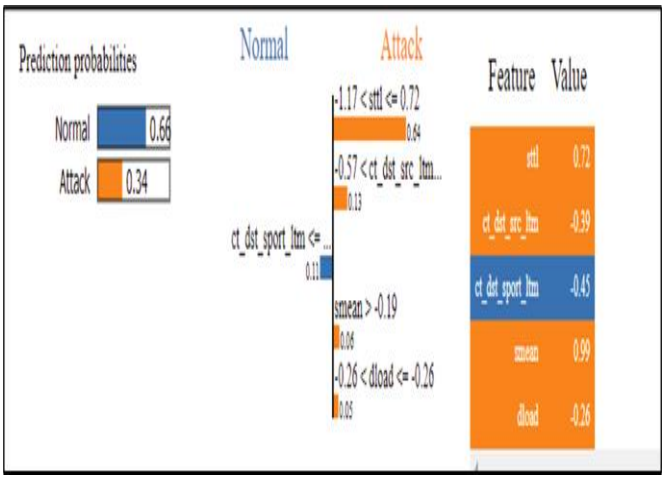

Figure 16: XGBoost Classifier Explanation for a Single Classification Prediction

The model showcases itself as a highly effective and versatile classifier, merging excellent performance with the strong explainability attributes offered by the LIME and SHAP libraries. This combination makes sophisticated black-box algorithms more reliable, which makes it possible to evaluate ML-based IDSs for IoT network security efficiently.

V. CONCLUSION

The ML learning models used to protect IoT network traffic through intrusion detection systems (IDSs) are getting more complex, but human analysts continue to play a critical role in deciphering results based on domain knowledge for resource allocation and cybersecurity strategy development. ML algorithms are sometimes thought of as "black boxes," with no understandable explanations provided for the predictions they produce. Using the UNSW-NB15 dataset to train Decision Tree, MLP, and XGBoost classifiers produced accuracy results that performed better when examining network behavior and differentiating between Attack and Normal Activity among connected clients in an Internet of Things network.

The performance of ML classifiers was then analyzed, and established libraries and techniques were applied to enable explainability, or Explainable AI (XAI), reveal decision-making processes, and evaluate feature importance. This increased transparency has the potential to promote ML system trust in the short term within the context of IoT cybersecurity. In the end, it will open up new possibilities for IoT cybersecurity by deriving knowledge from complex machine learning models, since improved explainability clarifies the variables affecting the likelihood and severity of cyberattacks.

REFERENCES

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

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

[3] 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.

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

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

[6] S. Slundberg, “SHAP (SHapley Additive exPlanations),” slundberg/shap. [Online]. Available: https://github.com/slundberg/shap. [Accessed: 30-Apr-2021].