Understanding Network Intrusion Detection Systems Using XAI

Faiz Ahmad Khan

Master of Science in Artificial Intelligence

from the

University of Surrey

Department of Electronic Engineering

Faculty of Engineering and Physical Sciences

University of Surrey

Guildford, Surrey, GU2 7XH, UK

September 2022

Supervised by: Dr Chuan Heng Foh and Sulyman Abdulkareem

DECLARATION OF ORIGINALITY

Understanding Network Intrusion Detection Systems Using XAI

Faiz Ahmad Khan

Khan

Author Signature Date: 09/09/2022

Supervisor's name: Dr Chuan Heng Foh

WORD COUNT

Number of Pages:	62
Number of Words:	12196

ABSTRACT

In this report, we review why there is a need for network intrusion detection systems, especially for internet of things (IoT) networks. We observe the limitations of an IoT environment and the need for efficient yet accurate classification algorithms. As it is often observed that complex machine learning algorithms cannot be realistically used in an IoT network. We detail the types of available intrusion detection systems (IDS) and the need for network-based systems for low-power devices. This is followed by a review of two novel IDSs that are implemented using the Bot-IoT dataset. The experimentation requirements and results lead to a search for techniques that can simplify the entire process. Next Explainable Artificial Intelligence (XAI) methods are used to better interpret the predictions made by these intrusion detection systems and the performance benefit of better understanding the algorithm is evaluated. The two tested XAI methods work in diverse ways and thus, the explanations generated by both were compared.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The rapid growth of digital communication technologies and the internet in the 21st century has entailed the development of advanced data and network security solutions. Specifically, the wide deployment of Internet of Things (IoT) enabled devices has increased the number of susceptible devices on the internet as they often have access to sensitive personal information[1]. These devices are now adopted in multiple fields ranging from home automation and city infrastructure to autonomous vehicles and healthcare devices[2]. The use of security tools such as a firewall, antivirus software or intrusion detection system (IDS) is essential to maintain a secure network. An IDS provides security by monitoring and flagging suspicious activity in a host device or network. It achieves this by using machine learning (ML) and deep learning (DL) techniques to learn useful features/patterns from network traffic data which are then used to predict malignant and benign activity on a network. The issue with these approaches is that they are used as black boxes, i.e., lack transparency in the decision-making process. Thus, any model biases or incorrectly learned weights cannot be diagnosed easily[3]. Additionally, the decisions made by the system cannot be explained by humans and so trust in the system is reduced[4].

This is where explainable artificial intelligence (XAI) is introduced to counter the issue of transparency as well as optimise model performance. The goal of this project is to evaluate how XAI techniques can be used to interpret the predictions made by an IDS and to try and improve IDS performance in IoT environments.

1.1 Background and Context

The previous decade introduced cloud computing and IoT devices to the mainstream. This trend will receive a major boost with the advent of widespread adoption of 5G communications in the next few years. As data becomes more decentralised and the number of low-end, internet enabled devices increases, concerns over security and privacy intensify as well. There is special concern regarding critical national infrastructure as essential services and information depend on these systems. These are often the targets of cyberattacks and of concern to both organisations and nations[5].

Thus, IDSs have become a vital research domain as well as essential tool in network security. They are used to detect misuse, abuse, and unauthorised use in a network. Network based intrusion detection systems (NIDSs) are especially popular for IoT networks because of the low-end nature of IoT devices. The other type of IDS is host based but most IoT devices cannot support the resource consumption rate of these systems. In general, developing for IoT devices presents various challenges. Older IDSs were developed for traditional networks and do not adapt well to IoT networks. They cannot counter state of the art intrusions. Another challenge is that IoT networks can be of diverse types, with each ecosystem having its own protocol stack. Additionally, IoT devices have low computational power and storage capacity, thus limiting IDS complexity. The complexity of the IDS depends largely on the complexity of the ML/DL algorithm it utilizes. The greater the complexity of the algorithm, the lower its interpretability. Figure 1 plots the relationship of model accuracy with respect to model interpretability and thus complexity.

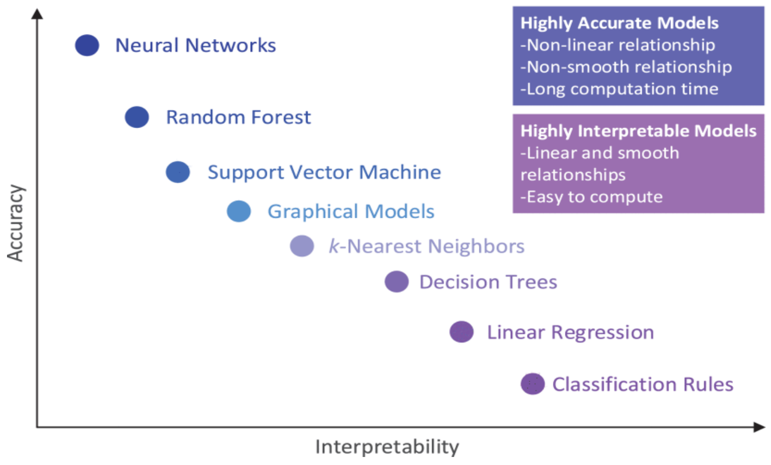


Figure 1: Trade-off between model accuracy and interpretability[38]

As the complexity of machine learning models increases so does the lack of understanding of how the system works. This leads to growing distrust in a system by users. Additionally, if the reasons for a particular decision by the system cannot be assessed, then debugging the system in the event of an incorrect prediction becomes especially difficult. On the other hand, if the system is well understood, then by performing feature engineering, the complexity of the model can be reduced[6]. This leads to improved execution performance of the NIDS which is important as the entire network will be slowed down if the NIDS performance is poor. Recent research has also argued that focusing on explain ability of a model during development will result in a more robust model as it will only learn the strongest of correlations[7]. Thus, multiple methods have been developed to better interpret complex machine learning models and are commonly labelled under Explainable Artificial Intelligence (XAI). These methods, although varied share two common core components, i.e., to produce explainable models while maintaining a high level of prediction accuracy and to enable humans to trust, manage and understand AI systems effectively[8]. These methods can be used be used with images, text, or tabular data. Some methods are specialised for a specific machine learning algorithm while others are model agnostic. The two most popular XAI methods are SHAP first proposed in [9] and LIME developed by [10][11].

1.2 Scope and Objectives

1. From published research, evaluate the best-in-class network intrusion detection systems and associated machine learning algorithms for IoT networks.
2. Evaluate the use of XAI techniques to improve NIDSs interpretability in an IoT network.
3. Evaluate the use of XAI techniques to improve NIDSs performance in an IoT network.

1.3 Achievements

In order to complete the first objective, a multi-class classification NIDS using Naive Bayes algorithm was evaluated and the results achieved were similar to the original paper the system was proposed in[12]. Additionally, a single class classification NIDS using Naive bayes, Light-GBM, XGBoost and Logistic Regression was evaluated as proposed in the paper and the results achieved were comparable with the original[13]. Next, the XAI methods SHAP and LIME were developed, and the NIDS developed before were examined to better understand their behaviour and to identify the performance improvements that could be made. Both these methods were compared along with how different algorithms and type of explanation affected the output of these methods.

1.4 Overview of Dissertation

The subsequent sections of the report are organised as follows:

Chapter 2 introduces the approaches to intrusion detection systems, specifically NIDS. This is followed by is review of the dataset used for the evaluations. After the discussion on the dataset a review of the state-of the-art network intrusion detection systems along with the algorithms used has been conducted. This section is followed by descriptions of the data pre-processing methods used for the project. The chapter is concluded with the theory behind the concept of XAI and a review of the state-of-the-art XAI methods that have been used in the report.

Chapter 3 presents the implementation of the reviewed NIDS. Additionally, the pre-processing methods used have been listed and evaluation metrics have been explained. Finally, the results of the evaluated NIDS have been analysed.

Chapter 4 compares the output of the two XAI techniques used along with the effect of different algorithms and prediction types on the output. Additionally, the top features identified by the XAI techniques were used to identify if any performance gain can be made by having a better understanding of the system.

Chapter 5 presents a summary of the work done, followed by an evaluation of the work done through the project. The chapter is concluded by a discussion on the possible avenues for future work in the topic.

2 BACKGROUND THEORY AND LITERATURE REVIEW

Intrusion detection systems are an essential part of network security, especially so when every device is in an interconnected and global, online network. This section will present how IDSs use machine learning algorithms to identify threats, followed by an analysis of the Bot-IoT dataset and the state-of-the-art NIDS proposed in the reviewed literature. Additionally, the data processing and machine learning algorithms used in these papers are examined. The chapter is concluded by a review of the explainable AI methods used to evaluate the NIDS.

2.1 Intrusion Detection System (IDS)

An intrusion can be defined as a sequence of activities that intend to compromise the confidentiality, integrity, or availability of information or to counter the security mechanisms of a device or network[14]. An intrusion detection system is an information security technology that is used to discover, determine, and identify unauthorized use, duplication, alteration, and destruction of information or networks. Figure 2 illustrates a generic intrusion detection model where the data collection module collects network traffic data. The analysis module assesses the network traffic from the data collection module through the use of statistical analysis or machine learning. It is used to reduce human intervention and enable the identifying of intrusions in real time. The data storage module is used to securely store data collected and any learnt threat signatures. The response module enables the system to either react to intrusions or be proactive, i.e., an alarm is set when an intrusion is detected[15]. The two main types of intrusion detection systems are Host-based Intrusion Detection System (HIDS) and Network Intrusion Detection System (NIDS).

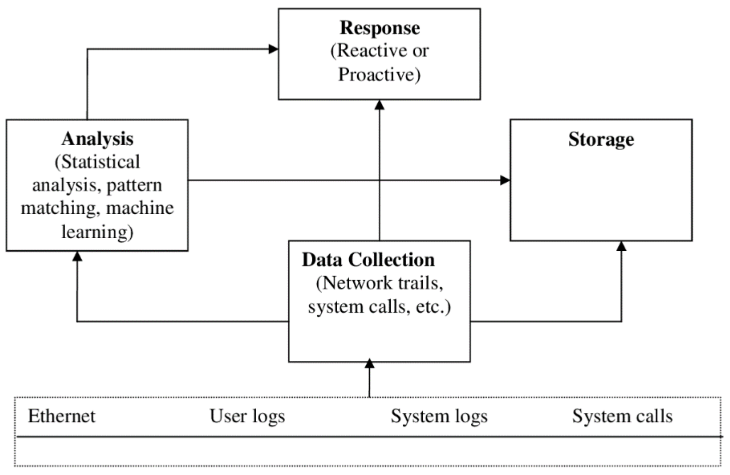


Figure 2: Generic Intrusion Detection model[15]

2.1.1 Host-based Intrusion Detection System (HIDS)

A Host-based IDS monitors activities in a single host and reports any detected malicious activity. It accomplishes this by analysing processes and network traffic related to the software environment associated with a specific host[16]. A HIDS on its own is not an effective protective measure and needs to be complemented with other security tools and best practices. In an IoT environment a HIDS is difficult to implement as the devices in this network have limited resources and often results in degradation of host performance. Figure 3 illustrates a basic HIDS architecture.

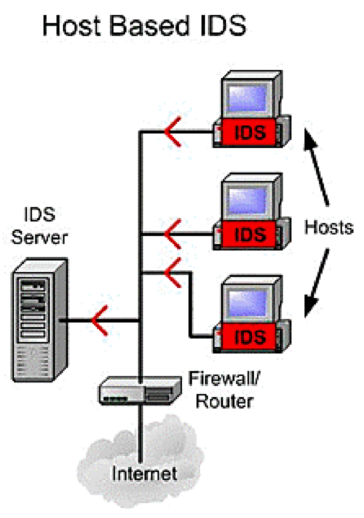


Figure 3: HIDS architecture[39]

2.1.2 Network Intrusion Detection System (NIDS)

A network intrusion detection system monitors traffic throughout the network and is independent of host devices. The system requires access to all network traffic but does not interfere with the network[17]. A NIDS can be further classified into either signature-based, anomaly-based or hybrid systems. Figure 4 illustrates a basic NIDS architecture.

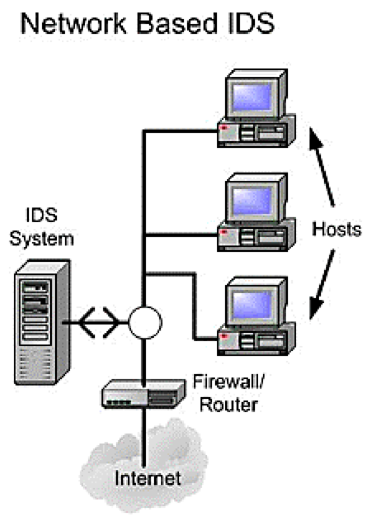


Figure 4: NIDS architecture[39]

Signature-based NIDS utilise the known signature of attacks to identify them. A signature is a pattern or series of signals that corresponds to a known attack or threat. As the system uses the knowledge accumulated by specific attacks and system vulnerabilities, it is also known as Knowledge-based Detection[14]. The advantage of these systems is that they are effective at identifying known types of attacks without generating an excessive number of false positives. The disadvantages of these systems are that they require frequent manual updates to the database of known attacks and that they cannot detect novel (zero-day) attacks[16].

Anomaly-based NIDS observe network behaviour and users over a period of time. During this period, the system is on alert for any anomalous behaviour, i.e., deviation from known network behaviour, profiles, and connections. The advantage of these systems is that they can detect novel attacks and exploitation of unforeseen vulnerabilities. Additionally, these systems facilitate identification of user privilege abuse[14]. The main disadvantage of these systems is that they have a high potential for false positives, i.e., legitimate but previously unseen network behaviour is flagged as an anomaly. Other disadvantages include issues with triggering alerts in the right time and unavailability of the system when behaviour profiles are being updated[14]. Anomaly detection is also usually computationally expensive because of the overhead of keeping track of and possibly updating several system profiles[15].

Hybrid techniques combine signature and anomaly-based detection methods. They are utilised to raise detection rate of known intrusions and decrease the false positive rate when dealing with unseen behaviour. Most of the implemented systems are hybrid in nature[16]. NIDSs in general face issues when dealing with high-speed network traffic overflow, signal generation lag and encryption[18].

2.2 Bot-IoT Dataset

The Bot-IoT[19] dataset consists of several types of attack traffic used by botnets along with normal IoT-related and other network traffic data. A botnet is essentially a network of compromised machines (bots) that are under the control of an attacker (botmaster). A botnet can launch a number of attacks, such as Distributed Denial of Service attacks (DDoS), keylogging, phishing, spamming, click fraud, identity theft and even proliferate other Bot malware[20]. The dataset consists of twenty-nine features and 73,370,443 instances. The dataset is available online in the form of multiple comma separated values (CSV) files. The category, subcategory, and number of instances in the dataset are listed in Table 1 while the features in the dataset have been listed in table 2.

Category	Subcategory	Number of Instances
Normal	Normal	9543
DoS	UDP	20,659,491
	TCP	12,315,997
	HTTP	29,706
DDoS	UDP	18,965,106
	TCP	19,547,603
	HTTP	19,771
Reconnaissance	OS Fingerprinting	358,275
	Service Scanning	1,463,364
Information Theft	Keylogging	1469
	Data Exfiltration	118

Table 1 : Bot-IoT dataset[19]

Features	**Description**
pkSeqID	Row Identifier
Stime	Record start time
flgs	Flow state flags seen in transactions
flgs_number	Numerical representation of feature flags
Proto	Textual representation of transaction protocols present in network flow
proto_number	Numerical representation of feature proto
saddr	Source IP address
sport	Source port number
daddr	Destination IP address
dport	Destination port number
pkts	Total count of packets in transaction
bytes	Total number of bytes in transaction
state	Transaction state
state_number	Numerical representation of feature state
ltime	Record last time
seq	Argus sequence number
dur	Record total duration
mean	Average duration of aggregated records
stddev	Standard deviation of aggregated records
sum	Total duration of aggregated records
min	Minimum duration of aggregated records
max	Maximum duration of aggregated records
spkts	Source-to-destination packet count
dpkts	Destination-to-source packet count
sbytes	Source-to-destination byte count
dbytes	Destination-to-source byte count
rate	Total packets per second in transaction
srate	Source-to-destination packets per second
drate	Destination-to-source packets per second
attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
category	Traffic category
subcategory	Traffic subcategory

Table 2: BoT-IoT features and descriptions[19]

This specific dataset has been selected for evaluation as it satisfies the eleven characteristics of a comprehensive IDS dataset as proposed by [21]. These characteristics are as follows:

- *Anonymity*
- *Attack Diversity*
- *Complete Capture*
- *Complete Interaction*
- *Complete Network Configuration*
- *Available Protocols*
- *Complete Traffic*
- *Feature Set*
- *Metadata*
- *Heterogeneity*
- *Labelling*

Due to the computational expense of using the full dataset, a subset of it was extracted. For this test, we make use of 5% of the BoT-IoT dataset proposed by[19] which includes 3,668,522 rows with forty-six columns divided among 4 CSV files. The dataset includes attack and benign categories, with 477 normal classes and 3,668,045 attack classes. The proposed dataset by [19] also includes additional network flow features extracted during simulated network flows. These generated flow features can be observed in Table 3.

	Feature	**Description**
1	TnBPSrcIP	Total Number of bytes per source IP
2	TnBPDstIP	Total Number of bytes per Destination IP
3	TnP_PSrcIP	Total Number of packets per source IP
4	TnP_PDstIP	Total Number of packets per Destination IP
5	TnP_PerProto	Total Number of packets per protocol
6	TnP_Per_Dport	Total Number of packets per dport
7	AR_P_Proto_P_SrcIP	Average rate per protocol per Source IP
8	AR_P_Proto_P_DstIP	Average rate per protocol per Destination IP
9	N_IN_Conn_P_SrcIP	Number of inbound connections per source IP
10	N_IN_Conn_P_DstIP	Number of inbound connections per destination IP
11	AR_P_Proto_P_Sport	Average rate per protocol per sport
12	AR_P_Proto_P_Dport	Average rate per protocol per dport
13	Pkts_P_State_P_Proto-col_P_DestIP	Number of packets grouped by state of flows and proto-cols per destination IP
14	Pkts_P_State_P_Proto-col_P_SrcIP	Number of packets grouped by state of flows and protocols per source IP

Table 3: Generated flow features[19]

2.2.1 Types of Attacks in dataset

The types of attacks in the Bot-IoT dataset are as follows:

- *Reconnaissance attacks*- These are malicious activities that are targeted to gather information about victims or vulnerabilities in their devices/networks. They are also called fingerprinting attacks. These attacks may be active, if the attacker generates network traffic or passive, the attacker simply captures packets in the network stealthily. Another way to categorise reconnaissance attacks is on the basis of the target of the attack. If the target is to gather information about the operating system (OS) of a remote system, then it is called an OS fingerprinting attack. This attack is usually conducted by comparing the responses of the system to pre-existing ones or on the basis of the TCP/IP stack implementation. If the target is the services that operate behind the system's ports, then this is called a service fingerprinting attack. This is done by sending request packets to the remote system[22].
- *Denial of Service*- These are malicious attacks that aim to disrupt a service and prevent legitimate users from accessing this service. When a single machine is used to disrupt the victim system then it is called a Denial-of-Service attack (DOS) and if multiple machines are used to disrupt the victim, then it is called a Distributed Denial of Service attack (DDOS). These attacks may be carried out by overloading the victim with dummy network traffic to cause large delays for legitimate requests or cause the machine to crash and go offline entirely. The other way to carry this attack out is by abusing the mechanics of internet protocols such as TCP, UDP or HTTP to prevent the victim machine from responding to requests due to depleted computation or memory resources.
- *Information Theft*- These malicious attacks aim to compromise the security of the victim machine/network in order to obtain sensitive information. This information might be stolen by compromising the victim and then either gaining unauthorised access to data or by recording keystrokes and gaining access to sensitive credentials. The first of these is called data exfiltration and the latter is called keylogging.

2.3 State-of-the-art NIDS

2.3.1 NIDS using Naive Bayes Classifier

In this paper [12], the authors propose an intrusion detection system consisting of three-tier fog computing architecture. The system uses a combination of three classifiers, two of which work concurrently, and the output is guided to the final classifier. Classifier one is a REP Tree, classifier two is a Jrip and classifier three is a Forest PA. The model was compared with several ML algorithms of which Naive Bayes was selected to be used for experimental comparison.

2.3.1.1 Naive Bayes Algorithm

Naive Bayes methods are a set of supervised learning algorithms based on the application of Bayes' theorem with the assumption of conditional independence between every pair of features given the value of the class variable. Naive Bayes classifiers can be extremely fast compared to more sophisticated methods due to the decoupling of the class conditional feature distributions. This means that each distribution can be independently estimated as a one-dimensional distribution[23]. Equation (1) illustrates the Bayes theorem. In simple words the theorem states that the posterior probability of an observation can be calculated from the ratio of prior probability of that observation into the likelihood and evidence.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{1}$$

Here, *c* is the class variable which in a NIDS may represent if the instance is an attack or not. The variable *x* represents the parameters. Thus, *x* can be represented as a set of features such as x_1, x_2, x_n as in equation (2).

$$x = (x_1, x_2, x_3, \dots, x_n) \tag{2}$$

If the expansion of *x* is substituted in equation 1 then by expanding the equation using the chain rule, an equation is obtained where each feature value can be substituted. For every entry in the dataset, the denominator does not change, hence, it can be removed, and proportionality is introduced. Thus, the final equation is obtained to calculate probability given all the feature points and is represented in equation (3).

$$P(c|x_1, \dots, x_n) \propto P(c) \prod_{i=1}^{n} P(x_i|c) \tag{3}$$

2.3.2 NIDS using Ensemble Feature Selection

In this paper [13], the authors propose a NIDS for specifically detecting the subcategories of information theft, keylogging, and data exfiltration. Classification of these attacks by ensemble feature selection techniques. Ensemble learning is a machine learning technique that utilises a combination of predictions from different algorithms to improve the accuracy of the classifier. The main ways of building an ensemble model are by using either bagging, boosting, or stacking. In bagging, multiple decision trees are built in parallel with each tree having a subset of training data and features. The aggregate over all predictions of each tree is the final result. In boosting, multiple trees are built in sequence. The subsequent tree corrects the errors of the preceding tree. LightGBM and XGBoost are some implementations of boosting[24]. In stacking, models with a variety of learning algorithms are created and this is followed by the training of a combiner algorithm to make final predictions based on the predictions generated by the learning algorithms[25]. Figure 5 illustrates the bagging and boosting mechanisms.

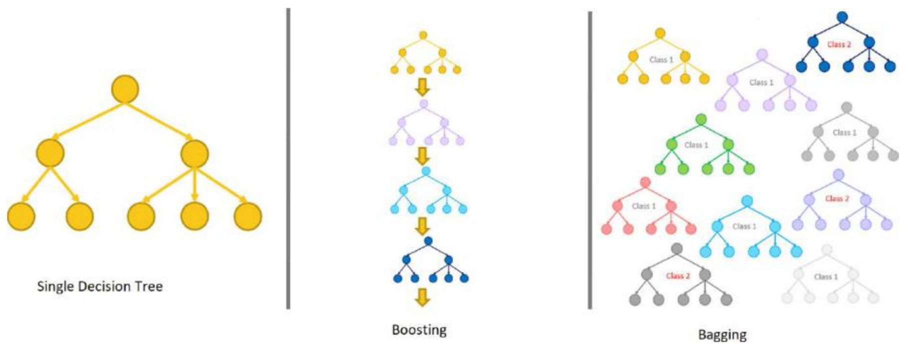


Figure 5: Bagging and boosting mechanisms[24]

In large datasets, a significant amount of time is taken to train the model due to redundant features and limited compute power. Additionally, the model accuracy is reduced. Feature selection techniques are utilised to isolate the important and non-redundant features to solve these problems[26]. Thus, the model focuses on the relevant information to make more accurate predictions with less computing power and time spent.

After applying data sampling and feature selection, four machine learning classifiers were selected from the paper to be used for classification of the dataset. Two ensemble learning classifiers, LightGBM and XGBoost; two individual learning classifiers, Logistic Regression and Naive Bayes.

2.3.2.1 Decision Trees

This is a supervised learning algorithm that is used for solving both classification and regression tasks, hence they are also known as Classification and Regression Tress or CART for short. A decision tree is structured like a flowchart, where each root node represents a test on a single input variable. The result of the test is represented by branches that lead to either other root nodes or terminal nodes. The terminal nodes represent the output of the tree, i.e., predictions[27].

The selection of input variables and the point of splitting the tree are chosen by minimizing a cost function through a greedy algorithm. This greedy algorithm is called recursive binary splitting. The tree construction ends when a predefined stopping criteria is met.

The Gini score represents how well a split is by calculating how mixed the response classes are in the groups created by the split. In equation (4) pk represents the proportion of same class inputs present in a group. When a group contains all inputs from the same class, then it has perfect class purity. In this case G is zero.

$$G = sum(pk * (1 - pk)) \tag{4}$$

The disadvantage with decision trees is they are vulnerable to variance, i.e., small variations in the data resulting in a completely different tree being generated. This is countered by boosting or bagging algorithms[28]. Gradient boosting is a type of boosting where each predictor is trained using the residual errors of the predecessor as labels, hence, correcting its predecessors' error.

2.3.2.2 XGBoost

An implementation of gradient boosting decision tress is Extreme Gradient Boosting (XGBoost). The authors of this algorithm state that it is a boosting algorithm that uses sparsity-awareness for handling sparse data (missing or zero data values) and weighted-quantile sketch for approximate tree learning, which is a data structure that supports merging and pruning functions[29]. Independent variables are assigned weights which are used by the decision tree which predicts results. The weight of incorrectly predicted variables by the tree is increased and the variables are used by the second decision tree. The ensemble of these individual classifiers returns a more precise model[30].

2.3.2.3 LightGBM

Light Gradient Boosting Model (LightGBM) is an implementation of gradient boosting that has been developed to increase efficiency and reduce memory usage of decision tree models. It was developed by researchers at Microsoft[31]. It introduces Gradient-Based One-Side Sampling (GOSS), and Exclusive Feature Bundling (EFB), two techniques that are characteristic of LightGBM.

Instances of the data with larger gradients contribute more to information gain, thus GOSS keeps these instances with large gradients and randomly drops instances with small gradients to retain accuracy of information gain estimation.

High-dimensional data is usually very sparse which enables the development of a nearly lossless approach to reduce the number of features. In a sparse feature space, many features are mutually exclusive thus, the exclusive features can be safely bundled into a single feature called an Exclusive Feature Bundle. Hence, the speed for training framework is improved without impacting accuracy[32].

2.3.2.4 Logistic Regression

This is a classification model that applies the sigmoid function to weighted input values to generate a prediction. The sigmoid function is illustrated by equation (5).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

A logit transformation is applied on the odds, i.e., the probability of success divided by the probability of failure. This is also commonly known as the log odds, is represented by equation (6).

$$odds = \ln \left(\frac{P}{1 - P}\right) \tag{6}$$

The probability for classification problems with two possible outcomes are modelled using logistic regression, a classification algorithm for assigning observations to a set of discrete classes[33].

2.4 Data Pre-Processing

To correctly implement any machine learning model, the data to be used must first be processed in order to achieve ideal results. Issues include missing data, noisy data, inconsistent data, or just major imbalances in the dataset. All of these issues negatively affect the model accuracy and must be dealt with before training. Some of the pre-processing methods used in this project are described in the following sections.

2.4.1 Label Encoder

The Label Encoder is a transformer that is used to normalize labels or to transform non-numeric labels to numeric labels. It is used to encode the target values and not the inputs to the machine learning model. This is used when the machine learning model cannot deal with non-numeric labels. It is available through the Scikit learn library for python.

2.4.2 Ordinal Encoder

The Ordinal Encoder is a transformer that is used to convert a set of either integers or strings to ordinal integers. It is used to encode input feature values and not output labels. It is used when machine learning algorithms cannot deal with non-numeric input features. It is a useful tool especially when, instances when missing values do not want to be discarded as the encoder can be used to adjust for and encode missing values as well. It is available through the Scikit learn library for python.

2.4.3 MinMax Scaler

The MinMax scaler is used to scale features to a given range. The default range for the output is between 0 and 1. Scaling is an important aspect of data pre-processing as many machine learning models are sensitive to magnitude and rely on scale to converge faster. An example of such an algorithm being support vector machines. In many datasets the units between two different features might be different as well and may result in a feature being more influential than it should be. The MinMax scaler does not change the shape of the original distribution and thus there is no meaningful change in the information gained from the original data. It is available through the Scikit learn library for python.

2.4.4 Variance Threshold

This is used to remove features that have low variance. It is used to remove static features in a dataset or to remove features that are not very predictive. It is often used with unsupervised learning models as it only looks at the features and not desired outputs. By removing features that are not predictive, the computation cost for the algorithm can be reduced. It is available through the Scikit learn library for python.

2.4.5 Random Under Sampler

The random under sampler is used when there is a large disparity between the majority and minor classes in a dataset. This disparity often results in selection bias if the data skews the model or sampling errors due to the random nature of drawing observations. The function samples the majority class in proportion to the minority class. It is available through the Imbalanced learn library for python.

2.5 Explainable Artificial Intelligence (XAI)

According to [34], explainable artificial intelligence (XAI), “is a set of processes and methods that allows human users to comprehend and trust the results and output created by machine learning algorithms.”

In addition to increasing accountability, and compliance with regulations as well as helping retain control and safety, XAI allows users to optimise model performance and decision making. Model performance is optimised as detecting flaws in the model and biases in the data allows for better optimisation. It can help verifying predictions, for improving models, and for gaining new insights. Decision making is improved as the reasons for each decision are possible to analyse. The effect and degree of effect of each factor can be assessed[35].

The next part of the project involves reviewing XAI techniques but in general the techniques fall under one of the following categories: visualisation, local explanation, feature relevance, model simplification, text explanation and explanations by example[36].

XAI methods can be used to interpret individual predictions, i.e., local explanations or the entire predictive characteristics of the model, i.e., global explanations.

At every stage of the ML pipeline explainable AI algorithms can be introduced to get insights. Explanations generated using training data help to identify if there any biases present in the data, which can be modified before training the model. Explanations generated by using trained model help in understanding if the model has learned any inappropriate rules or patterns, by retuning the model or reselecting the features a better model can be obtained. Explanations which are instance specific can help in debugging the model and are also useful after model deployment. Thus, at every stage in ML pipeline these explanations make it easier to design and deploy NIDSs[37]. Encouraging results in the application of XAI to improve NIDS performance have been documented in the work of [7]and[24].

2.5.1 LIME

Local Interpretable Model-Agnostic Explanations (LIME) is an algorithm that approximates complex models locally with an interpretable model in order to explain predictions. It was developed by researchers from the University of Washington, Seattle. Currently, only linear models are used to approximate local behaviour. The explanation produced by LIME is obtained by equation (7).

$$\xi(x) = arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{7}$$

A data point x is explained by a model g that minimizes the locality-aware loss. $L(f,g,\pi_x)$ *is a* measure of how unfaithful g approximates the model to be explained f in the locality defined by π_x.

$\Omega(g)$is the measure of complexity and in order to ensure both interpretability and local fidelity, minimize $\mathcal{L}(f, g, \pi_x)$ must be minimum while $\Omega(g)$ is low enough to be interpretable by humans[10].

In Figure 6, the blue region represents the class blue, and the pink region represents class pink. A non-linear model fits a complex decision boundary, like kernel SVM or Random Forest or any ensemble. The feature importance for a point marked by the bold '+' point in red is derived from the surrogate interpretable model indicated by the dotted line.

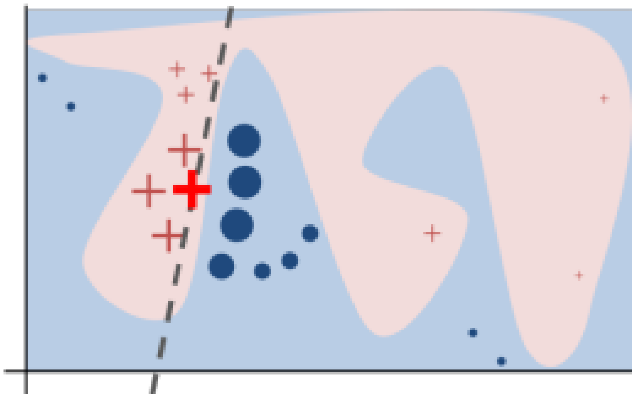


Figure 6: Intuition for LIME[10]

2.5.2 SHAP

The SHapley Additive exPlanations (SHAP) is a method based on cooperative game theory that is used to explain the predictions of a machine learning model. The basis for this method comes from the theory of Lloyd Shapley regarding fair distribution of pay-outs for a group of cooperating agents in a game. The pay-out is divided amount the agents on the basis of their contribution to achieving said pay-out and the bargaining power they hold within the group of agents. In SHAP, the authors replace the agents with individual features and the game is the black box machine learning model[9]. The contribution of each feature to achieving a prediction is calculated to gain insight into the machine learning model.

The technique is generally a local explanation technique, i.e., it is used to explain individual predictions of black-box models, however valid global explanations are also achievable by aggregating over individual explanations. Equation (8) represents the formula for Shapley values.

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!} [f_x(z') - f_x(\frac{z'}{i})] \tag{8}$$

Here, ϕ_i represents the Shapley value for feature i. f is the black-box model and x is an input data point such as a row in a tabular dataset. The method iterates over all possible subsets of feature groups

z' to account for the interactions between the individual feature values. The method evaluates the black-box model output for a subset with and without the feature of interest. The difference in the two outputs signifies the contribution of the feature in the subset. The process is repeated for each permutation of feature subsets to calculate the contribution of an individual feature. Additionally, each permutation is weighted according to how many features out of the total number of features *M!* are in the subset. The intuition behind weighting is that the contribution of a feature should be greater if many features are included in the subset as this would indicate that the selected feature has a greater contribution to the prediction even though many other features are included in the subset. On the other hand, a higher weight is also placed on small coalitions as in this case the features are isolated and their effect on predictions can be directly observed.

The complexity of calculating Shapley values can be fairly high as calculating all the permutations of feature subsets is computationally expensive. The number of all the possible subsets is given by the term 2^n where n is the number of features.

2.6 Summary

There has been considerable research in the development of effective IDSs. There are various architectures such as Host-Based and Network-based IDSs as well as hybrid architectures. Specifically for NIDS, after reviewing several novel approaches some common drawbacks have become clear. These systems still remain a black box, especially the models that show higher accuracy. In addition to this, few NIDSs are specifically designed for IoT networks and those too require considerable time-consuming feature selection and complex architectures. XAI is gaining momentum in the academic as well as corporate communities for multiple reasons. Thus, two XAI techniques, namely, LIME and SHAP have been introduced that can be used to show local and global explanations.

3 NETWORK INTRUSION DETECTION SYSTEMS

This chapter will present the initial work that has been conducted based on the approaches in [13] and [12]. Details of required pre-processing and performance evaluation metrics has been included as well.

3.1 Naive Bayes NIDS

The aim of this test is to classify all eleven subcategories of attacks in the dataset, evaluating the Naive Bayes Classifier (NB) used in [12]. This test was chosen as the literature indicates that ensemble ML classifiers have a superior classification performance compared with those of single learners.

3.1.1 Data Pre-processing

To create the training and testing dataset, The 4 CSV files were concatenated into one file. Next, the 'flgs', 'state', and 'proto' features were removed as they are redundant due to the inclusion of numerical features, 'flgs_number', 'state_number' and 'proto_number'.

The ordinal encoder is used to encode the features, 'saddr', 'daddr', 'sport' and 'dport' as they contain a mix of IP as well as hexadecimal values and positive and negative integer values, respectively. These changes, streamline the training process and as the true value of these features is not relevant for predictions, encoding the values has no negative effect. To reduce computation complexity, a variance threshold was applied to remove features that have no variation in the dataset. The dataset was split 80:20 into training and testing datasets.

3.1.2 Classification and Results

This section shows the how the ability of the NIDS to classify network traffic into the appropriate subcategory has been measured. A confusion matrix is used to display the classification results. Table 4 represents a standard classification matrix.

		Actual Class	
Predicted Class		Positive	Negative
	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Table 4: Confusion matrix

The correct positive prediction is known as True Positive (TP). The correct negative prediction is True Negative (TN). False Positive (FP) refers to a false positive prediction, whereas False Negative (FN) refers to a false negative prediction.

From the confusion matrix we can calculate the recall, precision, accuracy, and F-measure. Equations (9), (10), (11) and (12) illustrate the calculation of these metrics.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{11}$$

$$F\ Measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{12}$$

Recall is how often the positive class was predicted correctly (Actual TP). Precision is the correct positive prediction (Predicted TP). Accuracy is the correct positive and negative prediction and F-Measure is the weighted average of the recall and precision. Table 5 illustrates the confusion matrix of our Naive Bayes (NB) classifier.

		Actual Class										
		Data Exfil-tration	HTTP DDoS	HTTP DoS	Keylog ging	Normal	OS Fin-gerpr int	Service Scan	TCP DDoS	TCP DoS	UDP DDoS	UDP DoS
Predicted Class	Data Exfil-tration	1	0	0	0	0	0	0	0	0	0	0
	HTTP DDoS	0	248	0	0	0	0	0	0	0	7	0
	HTTP DoS	0	0	347	0	0	0	1	0	0	0	0
	Keylog-ging	0	0	0	18	0	0	0	0	0	0	0
	Normal	0	0	0	0	85	0	21	0	0	0	0
	OS Finger-print	0	0	0	0	0	4485	0	0	0	0	0
	Service Scan	0	0	0	0	3	0	18328	0	0	0	0
	TCP DDoS	0	0	0	0	0	0	0	243317	718	1	1
	TCP DoS	0	0	0	0	0	0	0	0	154341	0	0
	UDP DDoS	0	0	0	0	0	0	0	0	0	236721	4
	UDP DoS	0	0	0	0	0	0	0	1	7	0	258476

Table 5: Naive Bayes classifier confusion matrix

In the paper, NB classifier achieves ~75% accuracy on the Bot-IoT dataset, while in our experiments the accuracy is ~91.7%. The paper does not mention any parameter tuning but that might be affecting the results.

3.2 Feature Selection NIDS

In this NIDS approach [13], the aim is to detect the theft attack category. The information theft category, as well as its data exfiltration and keylogging subcategories, are the attack types identified. As a result, three datasets are created, each containing normal instances and a unique attack type such as one of which is made up of normal instances and information theft classes.

3.2.1 Data Pre-processing

In the BoT-IoT dataset, there are six features that are not relevant to our experiments. The pkSeqID and seq features were dropped as they only provide information on the order of the instances of the data. The features stime and ltime were also dropped because they only provide information of the start and last packet time of each instance and the same information is derived from the features dur, rate, srate and drate. Thus, stime and ltime will not provide additional information and may cause the model to overfit according to the authors. The features saddr and daddr were also dropped, they provide the source and destination IP address of each instance in the dataset and are just identifiers.

Lastly, all missing and hexadecimal values for sport and dport were changed to –1 indicating an invalid port value. The categorical features (flgs number, fproto number, sport, dport, and state number) were encoded using the One-Hot encoding process, which was implemented using category encoders. The encoder converts categorical features to numerical. Lastly, using the MinMax Scaler, we scaled the features to provide a normalised range from 0 to 1 for all numerical values.

3.2.2 Data Sampling

Due to significant class imbalance between the normal and Information Theft class, random under sampling (RUS) was utilised to balance the dataset. The ratio of the Information Theft class category to the normal class category is 79:477. Random Under sampling technique is used where instances of the majority class are dropped until the desired ratio to the minority class is achieved, in this case it was 1:3 and 1:1.

3.2.3 Classification and Results

Four classification algorithms were used namely, Naive Bayes(NB), LightGBM, XGBoost and Logistic Regression (LR). The classifiers were implemented using the sci-kit learn, XGBoost and LightGBM python libraries. Here, the AUPRC performance metric was used to evaluate the performance of the models. The results for the information theft subset are illustrated in Table 6.

	AUPRC
NB 1:1	0.85
NB 1:3	0.82
Light GBM 1:1	0.97
Light GBM 1:3	0.99
XGBoost 1:1	0.99
XGBoost 1:3	0.99
LR 1:1	0.99
LR 1:3	0.94

Table 6: Information Theft classification with RUS

The results for the keylogging and data extraction subset are illustrated in Tables 7 and 8, respectively.

	AUPRC
NB 1:1	0.83
NB 1:3	0.83
Light GBM 1:1	0.99
Light GBM 1:3	0.99
XGBoost 1:1	0.99
XGBoost 1:3	0.99
LR 1:1	0.99
LR 1:3	0.99

Table 7: Keylogging classification with RUS

	AUPRC
NB 1:1	0.99
NB 1:3	0.99
Light GBM 1:1	0.5
Light GBM 1:3	0.5
XGBoost 1:1	0.99
XGBoost 1:3	0.99
LR 1:1	0.99
LR 1:3	0.99

Table 8: Data Extraction classification with RUS

AUC is the area under the Receiving Operating Characteristic (ROC) curve. The Receiver Operator Characteristic (ROC) curve is a binary classification evaluation metric. It is a probability curve that compares the True Positive Rate (TPR) to the False Positive Rate (FPR) at various threshold levels, allowing the signal to be distinguished from the noise. The AUC measures how successfully a model distinguishes between positive and negative classifications. The higher the AUC, the better the predictive performance of the model or classifier. AUPRC is the Area Under the Precision Recall Curve, it plots the precision vs the recall.

The results achieved were similar to those achieved by the authors in their experiments. It can be observed that the most complex algorithm used, i.e., XGBoost performs the best in all cases while the simplest algorithm, i.e., Naive Bayes classifier has the lowest accuracy. Additionally, it was observed that the Logistic regression is significantly affected by the scaling of features. In regression variables are centred so that the predictors have mean zero, making it easier to interpret the intercept term. In the cases of keylogging and especially data extraction, the effect of limited datapoints is observable. The dataset includes only six cases of data exfiltration and thus it has no predictive power. This is clearly evident in the case of LightGBM which has no predictive power and hence accuracy is at 0.5, i.e., the prediction is as good as a random guess.

4 EXPLAINABLE ARTIFICIAL INTELLIGENCE

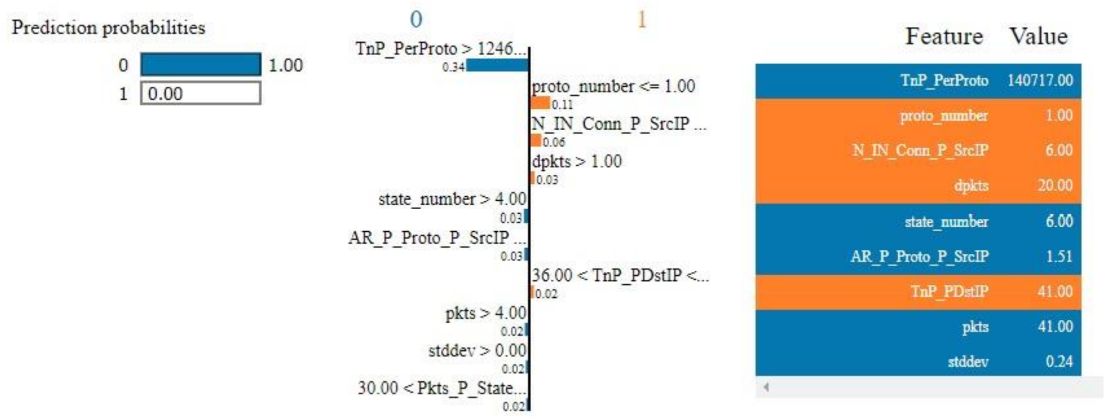
This chapter illustrates how XAI methods can be used to improve the interpretability of network intrusion detection systems. If complex algorithms such as XGBoost and LightGBM can be interpreted, then any issues in the system can be better understood and trust in the system is increased. Along with better understanding of the system, features that have low contribution can be eliminated to decrease computational complexity. In the sections ahead, LIME and SHAP will be used to evaluate the machine learning algorithms that have been examined in the previous chapter.

4.1 Attack Category Based Explanations

The experiments conducted in this section used the dataset partition proposed in [13]. In this section, the effect of different algorithms on the explanations generated by LIME is evaluated. Additionally, the explanations generated by LIME and SHAP were compared if used upon the same machine learning algorithm. Finally, the output generated from local and global explanations is compared.

4.1.1 Effect of algorithm on feature ranking

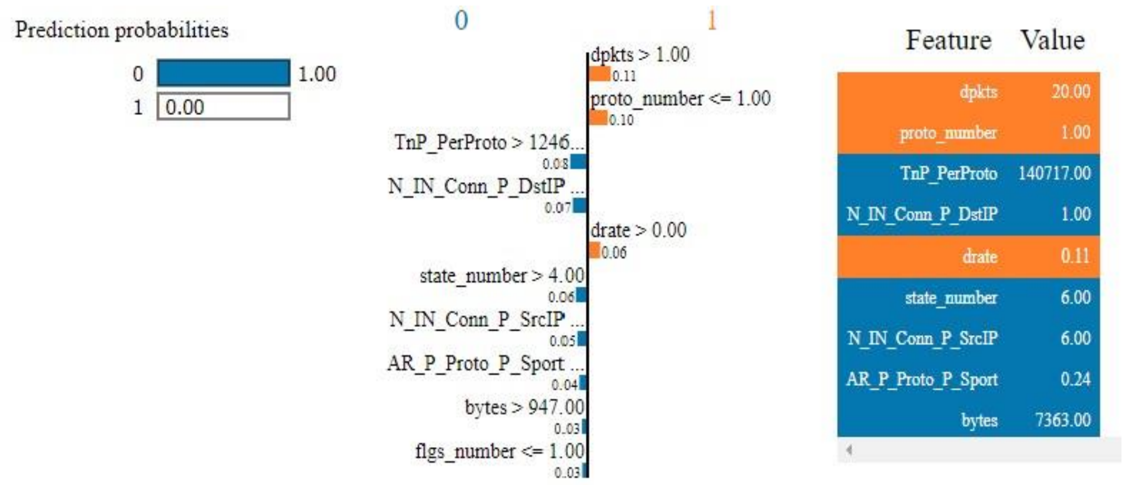
In Figures 7, 8, 9 and 10, the LIME output for the machine learning algorithms Light GBM, XGBoost, Logistic Regression and Naive Bayes have been illustrated. All of these algorithms were run on the Keylogging subset of the Bot-IoT dataset. LIME presents the contribution of each feature to the prediction of an instance of data. The tests below were conducted on the second instance of data in the test dataset. The central figure illustrates the contribution of the top features to the prediction corresponding to the weights of linear model. The right-hand table illustrates the top features along with their value in the dataset.



Feature	Value
TnP_PerProto	140717.00
proto_number	1.00
N_IN_Conn_P_SrcIP	6.00
dpkts	20.00
state_number	6.00
AR_P_Proto_P_SrcIP	1.51
TnP_PDstIP	41.00
pkts	41.00
stddev	0.24

Figure 7: LIME output for LightGBM (Keylogging)

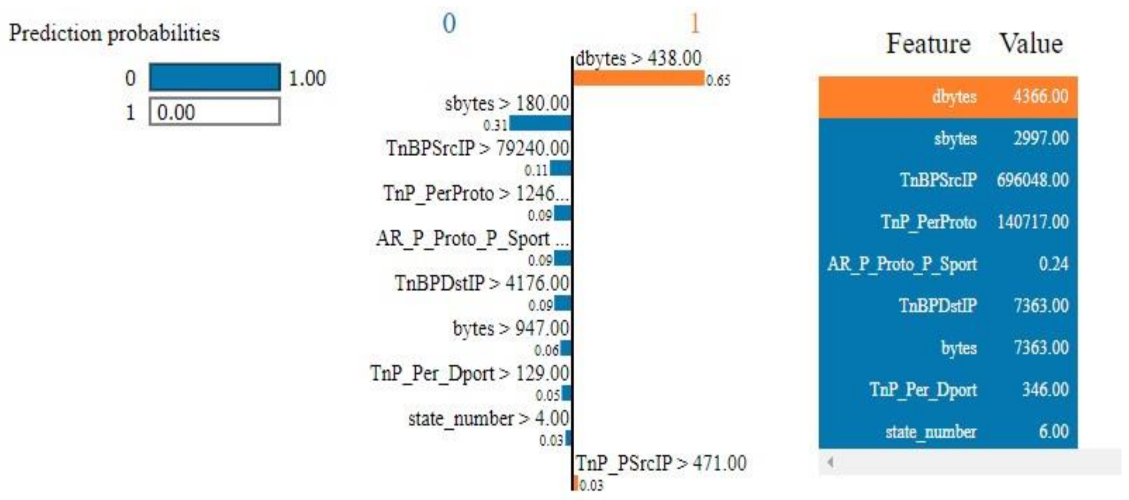
In Figure 7, the Light GBM model has predicted that the instance belongs to the normal category, i.e., benign packets with 100% confidence. In this instance the total number of packets per protocol is greater than 124600 and the state number is six, corresponding to the state- FIN i.e., the packet indicating no more data will be transmitted from sender. Thus, the algorithm has predicted that the communication is benign. Also, to note, the protocol number, number of dropped packets and the number of inbound connections per source IP negatively influence the algorithm. These rules set by LIME make sense to humans as well as it is perfectly reasonable to be suspicious if the number of dropped packets is high.



Feature	Value
dpkts	20.00
proto_number	1.00
TnP_PerProto	140717.00
N_IN_Conn_P_DstIP	1.00
drate	0.11
state_number	6.00
N_IN_Conn_P_SrcIP	6.00
AR_P_Proto_P_Sport	0.24
bytes	7363.00

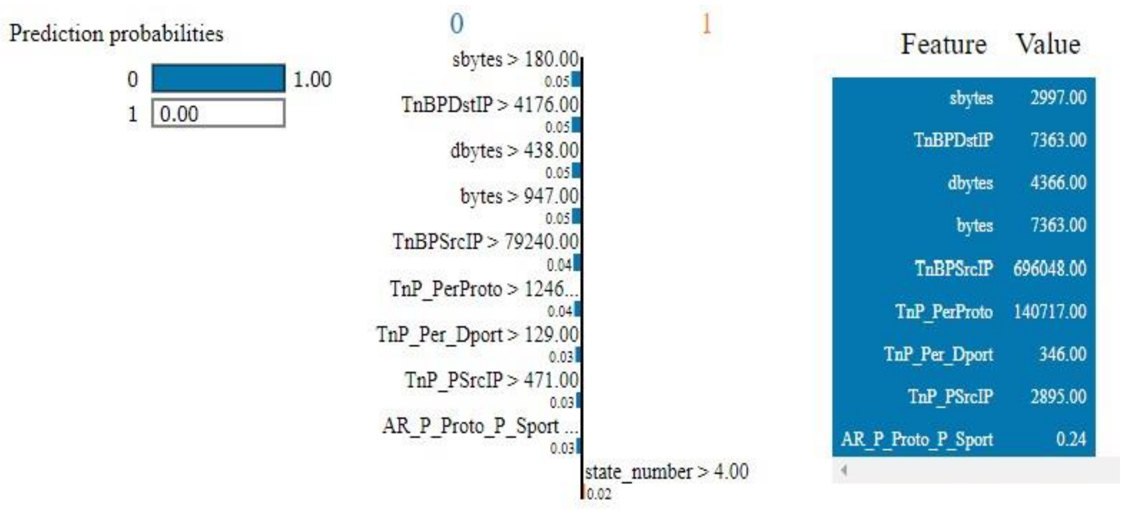
Figure 8: LIME output for XGBoost (Keylogging)

It can be observed that even though the four algorithms are predicting the same output for the same instance of input data, there are small variations in the importance of each feature in each of the algorithms. For example, in XGBoost and Logistic Regression, the features with the most positive influence are total number of packets per protocol and source to destination byte count, respectively. This is due to the difference in the complexity of the two algorithms. As discussed before, LIME makes local approximations of complex algorithms using linear models. Complex algorithms like XGBoost are more difficult to approximate using a linear model, while a logistic function can still be modelled comparatively easily. It must also be noted that each algorithm makes predictions differently and different features might have more predictive power in certain algorithms. In Light GBM and XGBoost, models that both work on decision tress, the top positive and negative contributing features are ranked similarly, thus the role of the algorithm in utilising the predictive power of a feature is evident.



Feature	Value
dbytes	4366.00
sbytes	2997.00
TnBPSrcIP	696048.00
TnP_PerProto	140717.00
AR_P_Proto_P_Sport	0.24
TnBPDstIP	7363.00
bytes	7363.00
TnP_Per_Dport	346.00
state_number	6.00

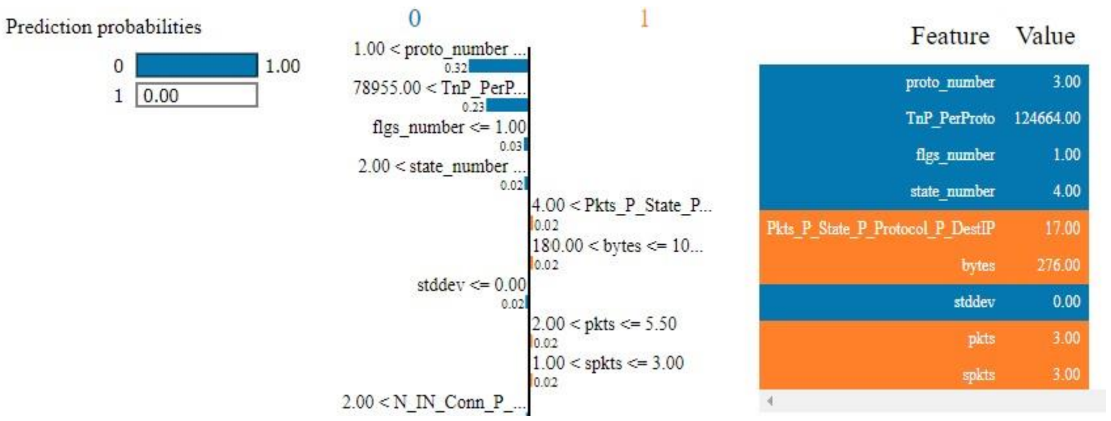
Figure 9: LIME output for Logistic Regression (Keylogging)



Feature	Value
sbytes	2997.00
TnBPDstIP	7363.00
dbytes	4366.00
bytes	7363.00
TnBPSrcIP	696048.00
TnP_PerProto	140717.00
TnP_Per_Dport	346.00
TnP_PSrcIP	2895.00
AR_P_Proto_P_Sport	0.24

Figure 10: LIME output for Naive Bayes (Keylogging)

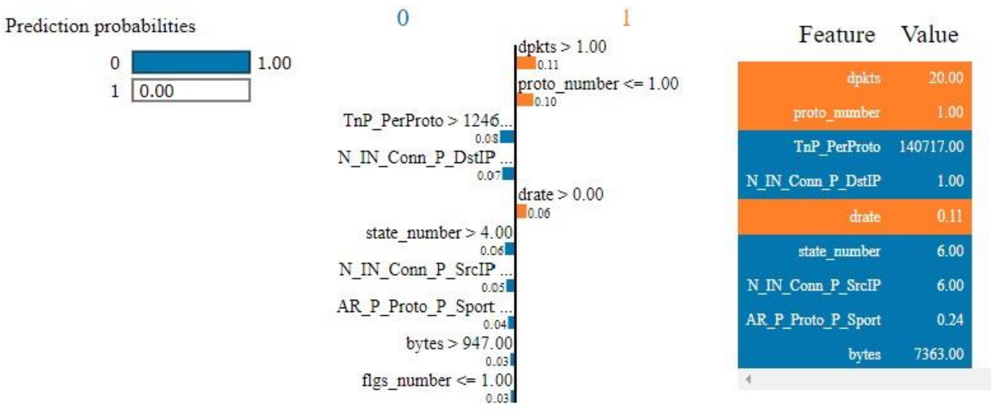
If the same results are observed for the entire information theft subset, then the features that are relevant to both keylogging and data theft can be inferred. Figures 11, 12, 13 and 14 illustrate the output for LIME when the entire information theft subset is evaluated.



Feature	Value
proto_number	3.00
TnP_PerProto	124664.00
flgs_number	1.00
state_number	4.00
Pkts_P_State_P_Protocol_P_DestIP	17.00
bytes	276.00
stddev	0.00
pkts	3.00
spkts	3.00

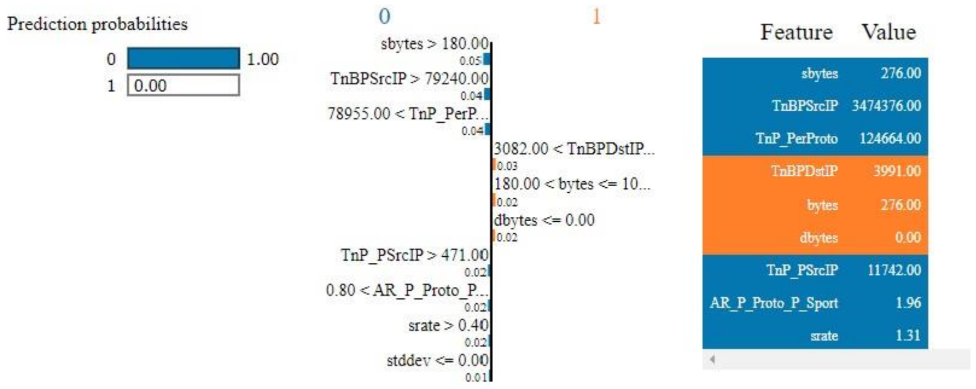
Figure 11: LIME output for LightGBM (Information Theft)

From comparing the results in Figure 7 and 11, it can be observed that even though the two predictions are for different instances of the same attack category in the dataset, certain features are now more influential such as the protocol number, which in this instance corresponds to the UDP protocol. This is due to the fact that the algorithm now predicts based on the attack category instead of the subcategory. The attack category of information theft encompasses both keylogging and data theft.



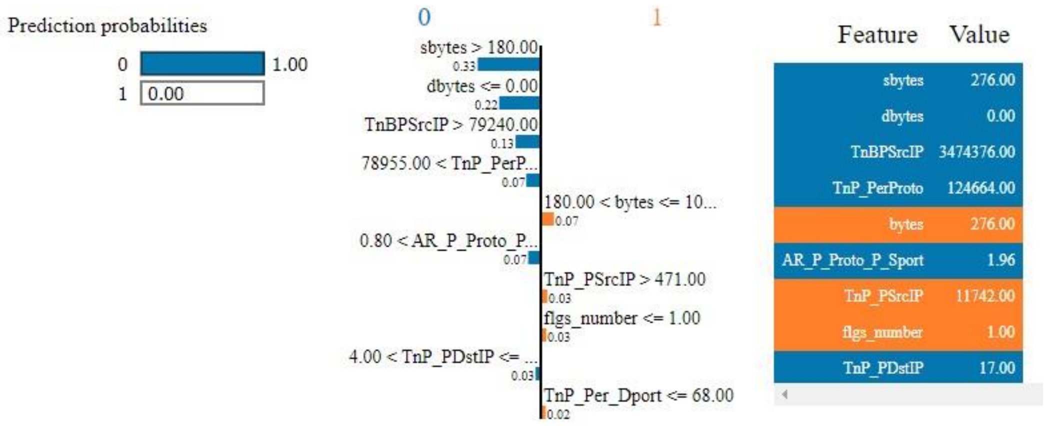
Feature	Value
dpkts	20.00
proto_number	1.00
TnP_PerProto	140717.00
N_IN_Conn_P_DstIP	1.00
drate	0.11
state_number	6.00
N_IN_Conn_P_SrcIP	6.00
AR_P_Proto_P_Sport	0.24
bytes	7363.00

Figure 12: LIME output for XGBoost (Information Theft)



Feature	Value
sbytes	276.00
TnBPSrcIP	3474376.00
TnP_PerProto	124664.00
TnBPDstIP	3991.00
bytes	276.00
dbytes	0.00
TnP_PSrcIP	11742.00
AR_P_Proto_P_Sport	1.96
srate	1.31

Figure 13: LIME output for Naive Bayes (Information Theft)

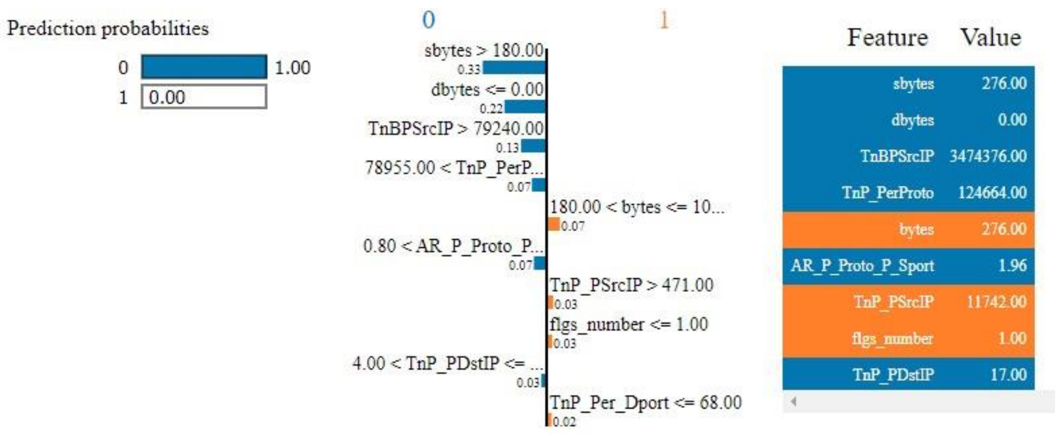


Feature	Value
sbytes	276.00
dbytes	0.00
TnBPSrcIP	3474376.00
TnP_PerProto	124664.00
bytes	276.00
AR_P_Proto_P_Sport	1.96
TnP_PSrcIP	11742.00
flgs_number	1.00
TnP_PDstIP	17.00

Figure 14: LIME output for Logistic Regression (Information Theft)

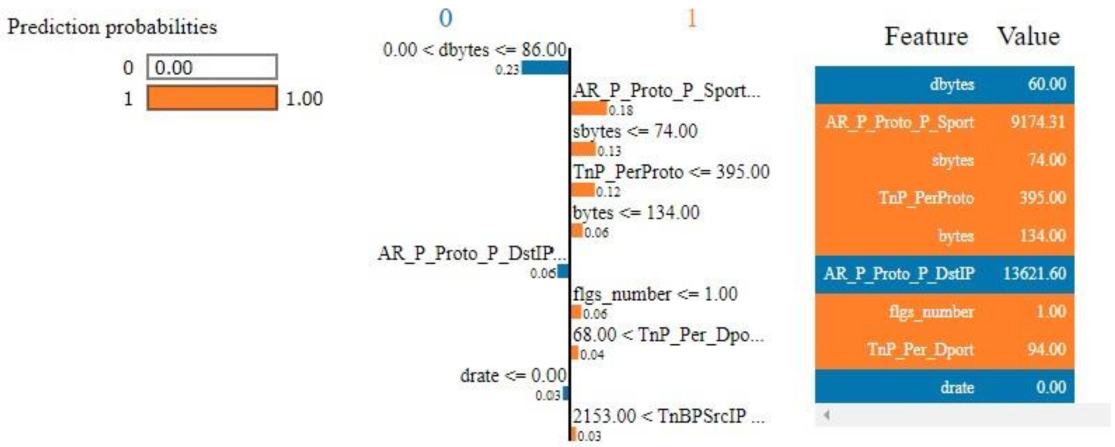
4.1.2 Effect of XAI technique on feature ranking

SHAP and LIME differ in the way they calculate the contribution of a feature to the final prediction. This results in differences in the ranking of the most influential features for the same instance in a dataset. Figures 15 and 16 illustrate the top features ranked by LIME for a NIDS using Logistic Regression and the information theft subset. Figure 15 illustrates an instance where the prediction is for benign network connection while Figure 16 illustrates prediction for malignant network connection.



Feature	Value
sbytes	276.00
dbytes	0.00
TnBPSrcIP	3474376.00
TnP_PerProto	124664.00
bytes	276.00
AR_P_Proto_P_Sport	1.96
TnP_PSrcIP	11742.00
flgs_number	1.00
TnP_PDstIP	17.00

Figure 15: LIME output for benign connection (Information Theft)



Feature	Value
dbytes	60.00
AR_P_Proto_P_Sport	9174.31
sbytes	74.00
TnP_PerProto	395.00
bytes	134.00
AR_P_Proto_P_DstIP	13621.60
flgs_number	1.00
TnP_Per_Dport	94.00
drate	0.00

Figure 16: LIME output for malignant connection (Information Theft)

Figures 17 and 18 illustrate the top features ranked by SHAP for a NIDS using Logistic Regression and the information theft subset. Figure 17 illustrates an instance where the prediction is for benign network connection while Figure 18 illustrates prediction for malignant network connection.

In Figure 17 the models predicted probability *f(x)* is -35.932. This value is evaluated using shapley values and the negative value indicates that the prediction is for zero or benign connection. The base value *E[f(x)]* at the bottom of the figure indicates the value that would be predicted if the model did not have any features for the current instance. The base value is calculated by averaging the model output over the training dataset. The numbers in the centre with the arrows indicate the SHAP value while the numbers next to the feature labels on the left indicate the actual value in the dataset. The red arrows in the graph indicate positive contribution of the feature towards the predicted output. The blue features indicate negative contribution of the feature towards the predicted output. The larger the arrow, the greater the impact of the feature towards the output.

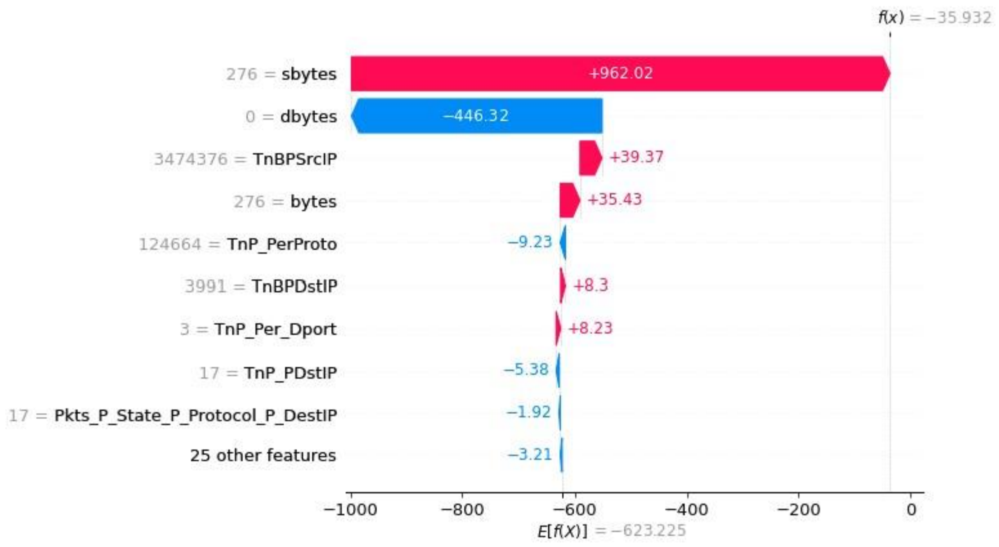


Figure 17: SHAP output for benign connection (Information Theft)

If the Figures 16 and 18 are compared, it can be observed that they both are prediction of the same malignant connection instance from the dataset. Yet, the top impactful features in both scenarios are slightly different. For the case of LIME, the average rate per protocol per sport, the source-to-destination byte count and total number of packets per protocol are the three most impactful positive features respectively while in the case of SHAP, the source-to-destination byte count, the total number of bytes per source IP and the total number of bytes in the transaction are the three most impactful positive features. In both cases, the destination-to-source byte count is the most impactful negative feature. A similar case can be observed in the Figures representing the benign case. This difference is observed as the two methods have different approaches to explainability. LIME creates local interpretable models that approximate the complex algorithm while SHAP works by applying concepts of cooperative game theory and replacing players with features that are used to calculate the shapley values. Thus, there are some subtle differences in the ranking of the most influential features.

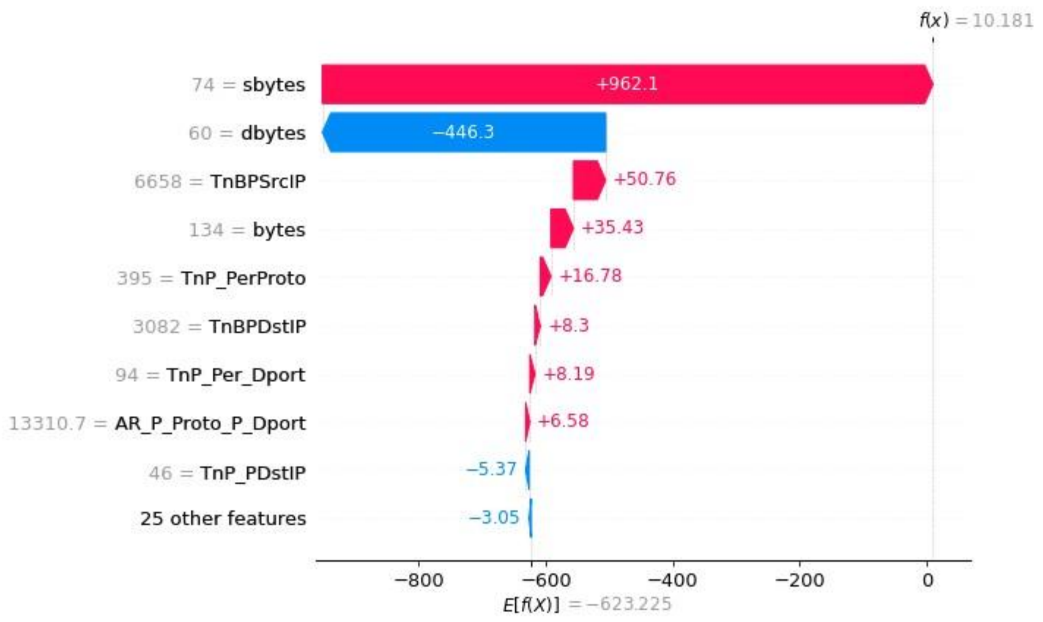


Figure 18: SHAP output for malignant connection (Information Theft)

4.1.3 Local vs Global Explanations

A major benefit of using SHAP over LIME is its ability to create global explanations. Global explanations describe the overall behaviour of the NIDS. Figure 19 illustrates the global explanation for a NIDS using the XGBoost algorithm and information theft subset. In this figure, the high values of the protocol number and total number of packets per protocol have a negative impact on the model. On the contrary, low values of the protocol number have a positive impact on the model. The features are placed in the figure according to their impact on the model output. Points on the left of the vertical axis have a negative impact on the model, while point son the right have a positive impact on the model.

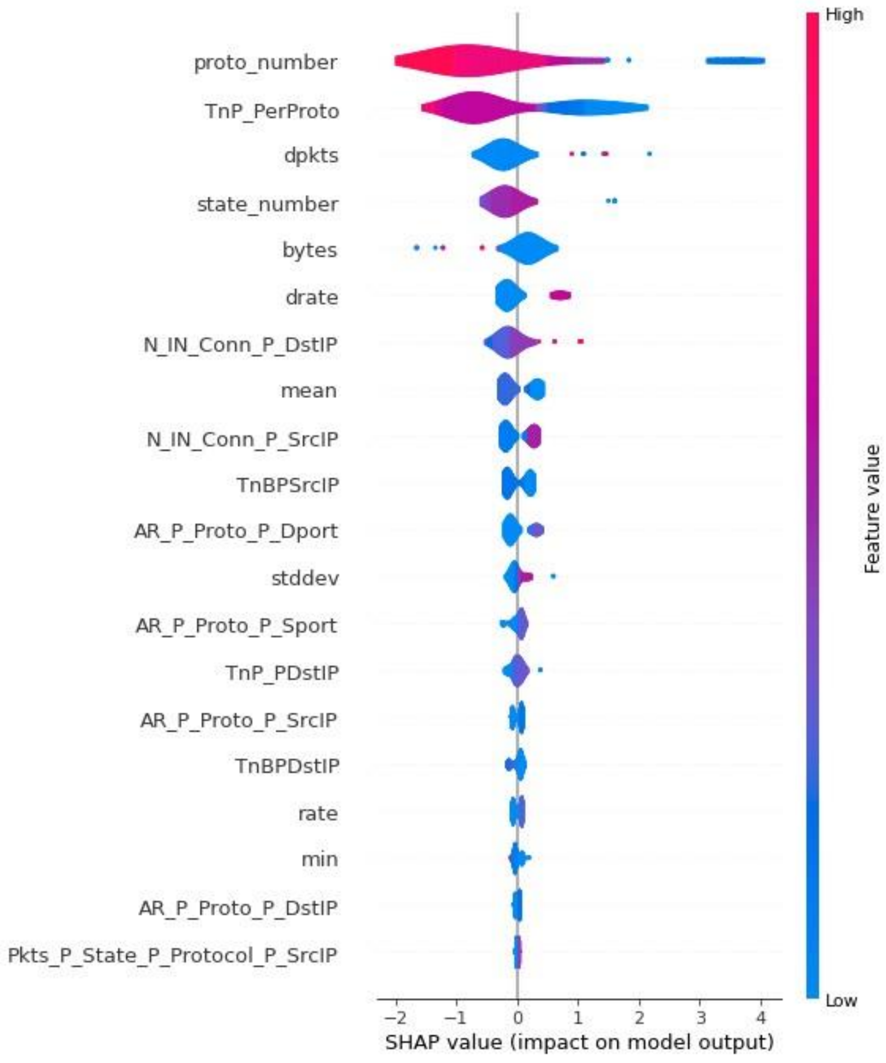


Figure 19: SHAP global explanation for XGBoost (Information Theft)

Figures 20 and 21 illustrate local predictions for a benign and malignant instance, respectively. In both cases, the top two most influential features are the same. The protocol number and total number of packets per protocol are also the most impactful features in the global explanation. Hence, a direct correlation can be observed from these three figures as global explanations are created by compiling all the local explanations.

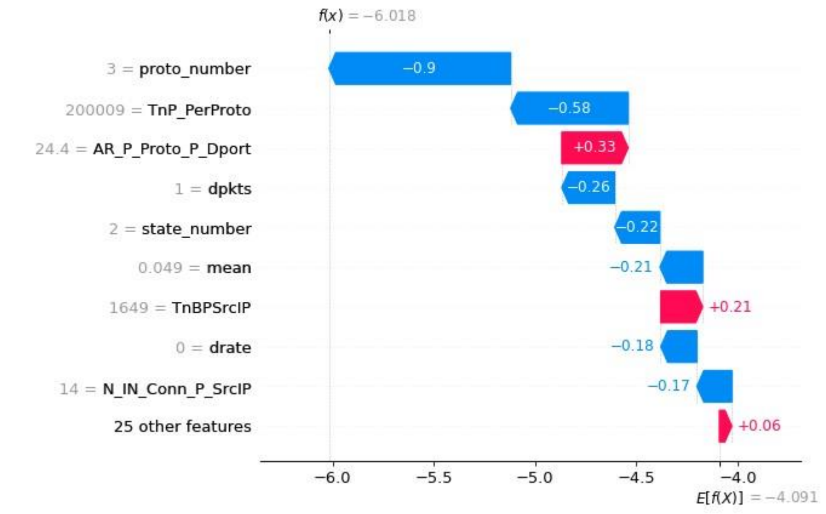


Figure 20: SHAP explanation for XGBoost (Benign)

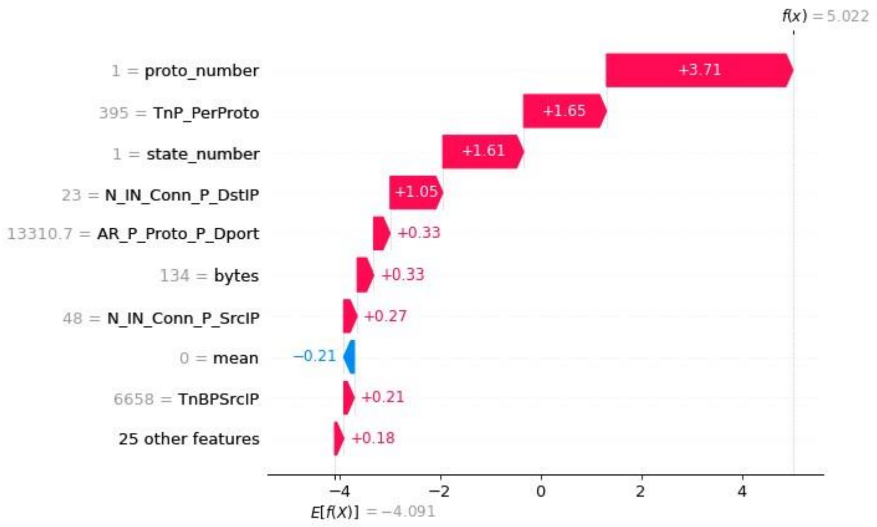


Figure 21: SHAP explanation for XGBoost (Malignant)

4.2 Multiclass Dataset Explanations

In this section the complete BoT-IoT dataset is evaluated using XAI. Due to its large size, generating explanations for the entire dataset is a computationally expensive task. This section will be used to compare the output of LIME and SHAP when dealing with large multiclass datasets. The effect of using different machine learning algorithms will also be evaluated. Next, the dataset will be cut back to only include the features with the most impact. The effect of this pruning on accuracy and processing time will be reviewed.

4.2.1 LIME vs SHAP

When dealing with large datasets, both LIME and SHAP have their advantages and disadvantages. SHAP is a more complex XAI method and hence requires greater time and computational resources as compared to LIME. This cost is mitigated by the act that SHAP can be used to develop global explanations. The advantage of LIME is that it does not require as much time to generate explanations, but it does not perform as well as SHAP when dealing with complex algorithms.

Figure 22 illustrates the global explanation for the XGBoost algorithm using the Bot-IoT dataset generated by SHAP. Here, the impact of a feature on the classes is stacked to create the plot. This plot illustrates what a multiclass classification NIDS learned from the features.

In the given case, it can be observed that Denial of Service (DoS) attacks cannot be identified by using the features, dport, sbytes and TnBPSrcIP. It can also be observed that if the N_IN_Conn_P_DstIP and N_IN_Conn_P_SrcIP were the only available features then differentiating between Normal connections and DoS attacks would be exceedingly difficult as both use the two features equally. In order to separate between the two classes, new features would have to be generated that can uniquely be dedicated towards these classes. Static features do not have any predictive power and thus do not have an impact on predictions.

Figure 23 illustrates the output for LIME when used for multiclass classification. The figure presents the class the particular instance is predicted to belong to and the features of significance. When compared with the local explanation generated from SHAP, illustrated in Figure 24, it can be observed that with the exception of the topmost impactful feature, the explanations generated by SHAP, and LIME still have differences. Figure 25 illustrates the SHAP output for the global explanation for each subcategory. In this figure, the impact of each feature on every subcategory of attack can observed.

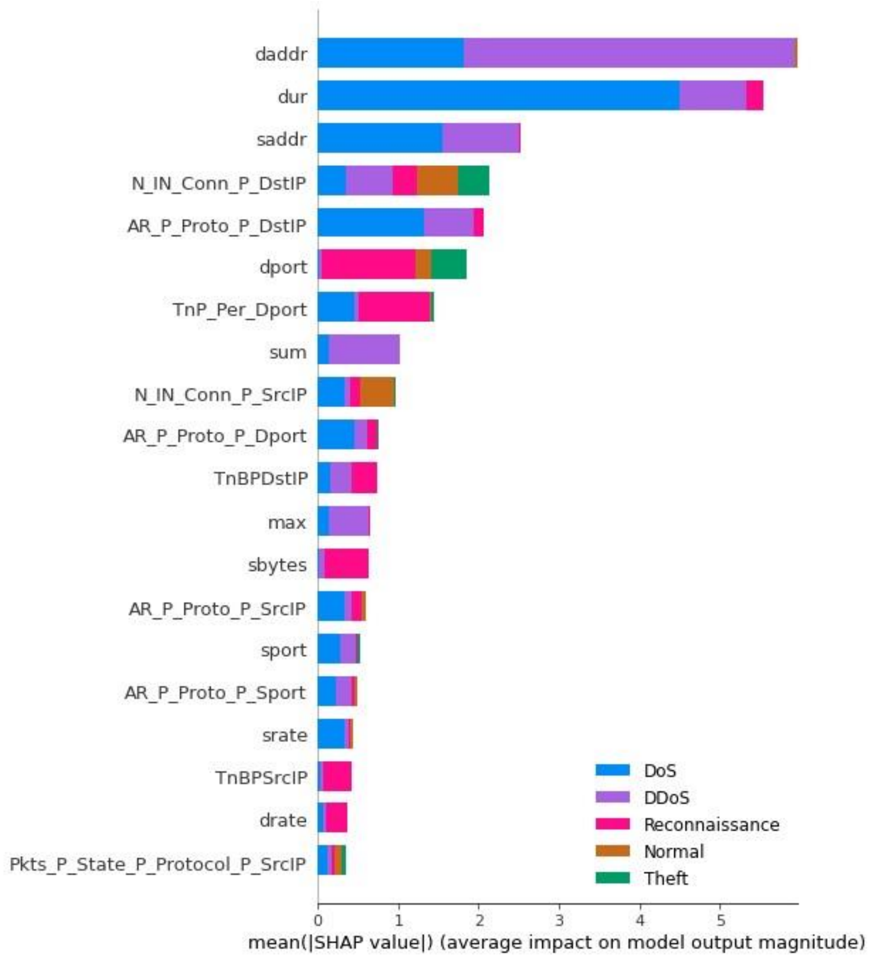
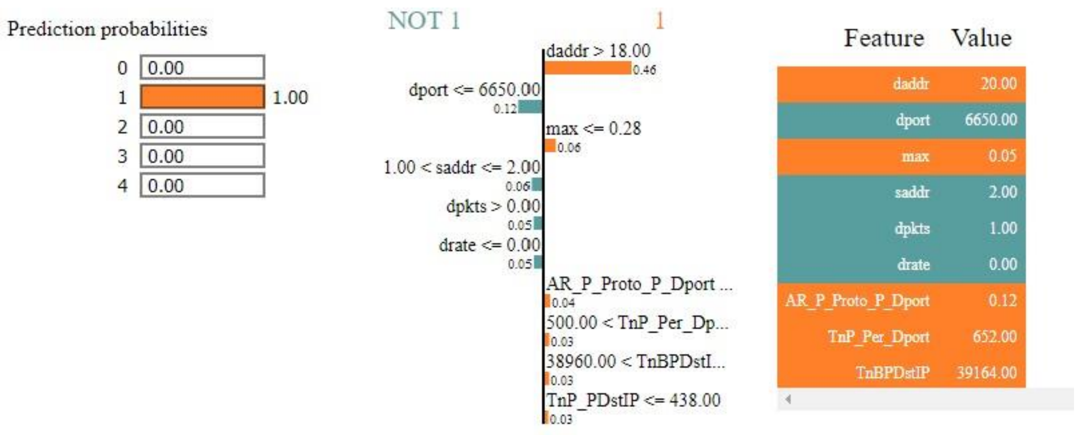


Figure 22: SHAP output for XGBoost (Multiclass)

When dealing with multiclass classification, the LIME output provides information on the exact class that has been predicted while the SHAP local explanation does not.



Feature	Value
daddr	20.00
dport	6650.00
max	0.05
saddr	2.00
dpkts	1.00
drate	0.00
AR_P_Proto_P_Dport	0.12
TnP_Per_Dport	652.00
TnBPDstIP	39164.00

Figure 23: LIME output for XGBoost (Multiclass)

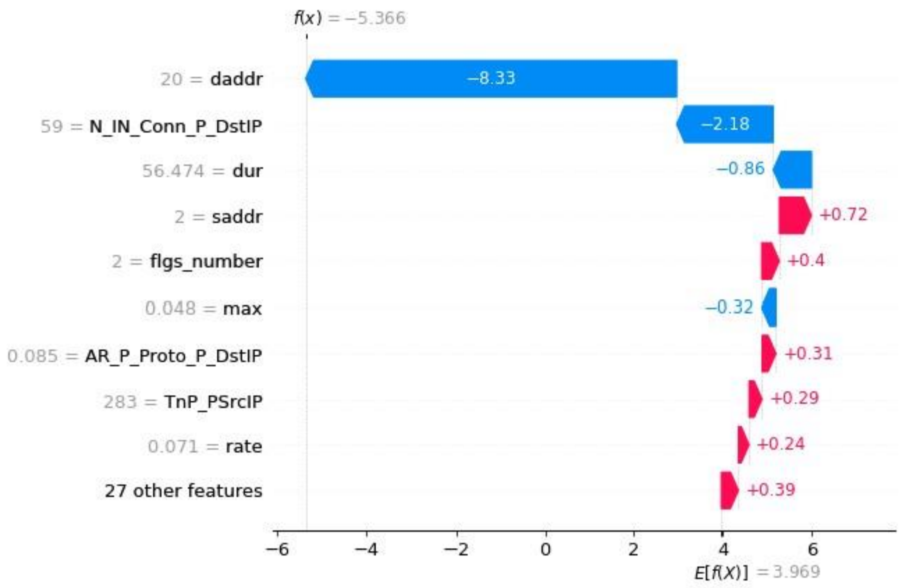


Figure 24: SHAP local explanation for XGBoost (Multiclass)

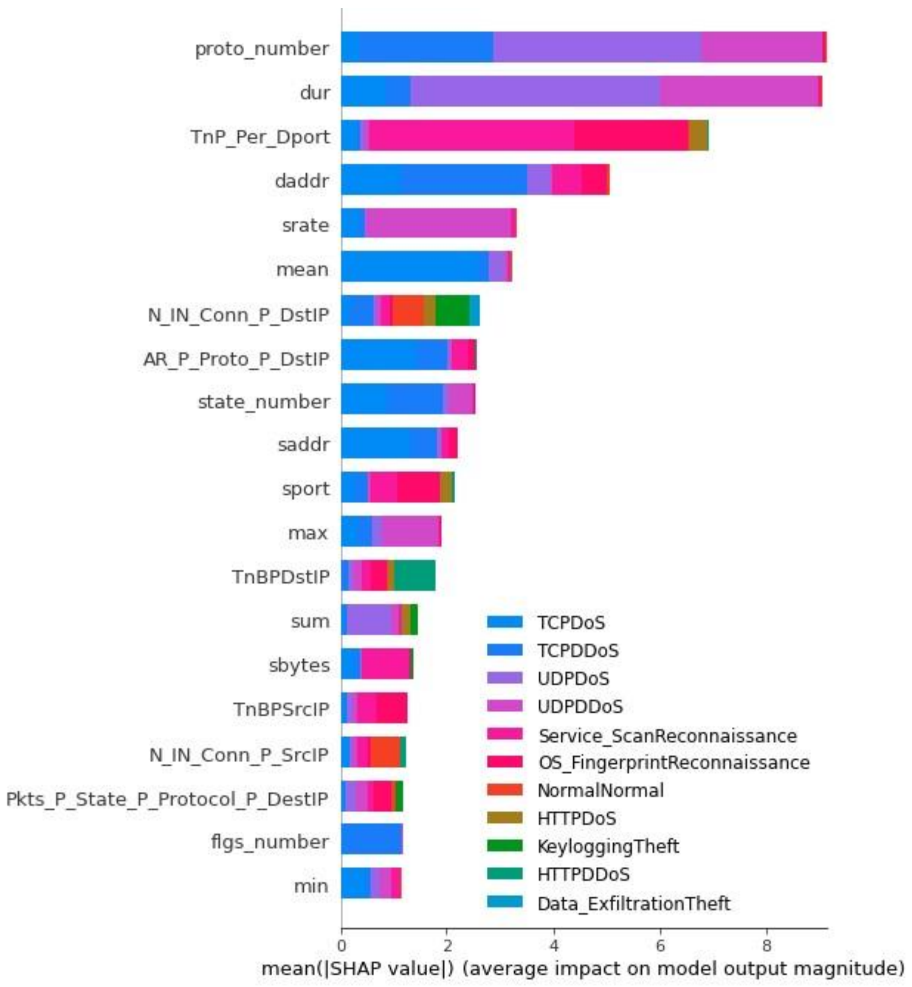


Figure 25: SHAP output for global explanation of subcategories

4.2.2 Performance Optimisation

XAI presents the opportunity to reduce computation time and increase accuracy of a machine learning algorithm, especially when dealing with class specific NIDS. It has already been observed that some features have a negative influence on the prediction of an algorithm and thus by selecting only the most impactful features, it can be shown that the computation time and accuracy can be improved. For this section, XGBoost and LightGBM will be tested as they are the more complex algorithms and performance can be more significant.

4.2.2.1 Multiclass classification

In Figures 26 and 22 the global impact of each feature can be observed when using a LightGBM and XGBoost NIDS, respectively.

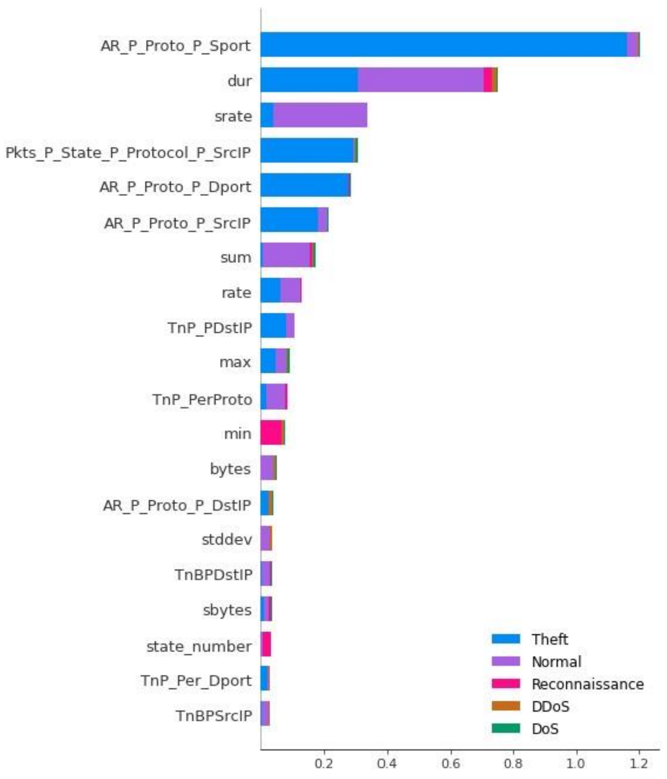
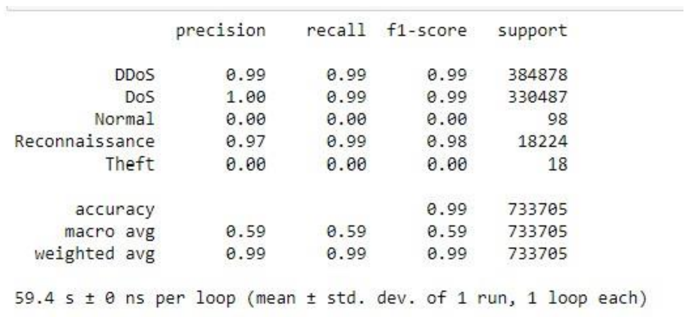


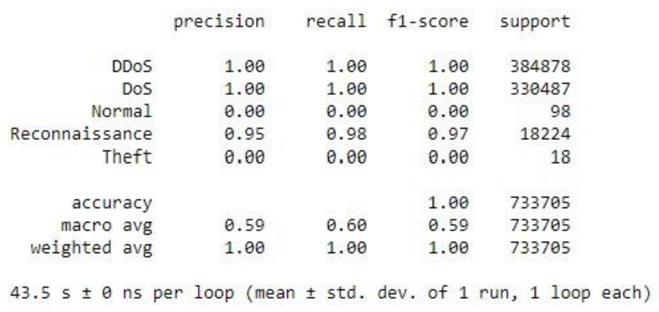
Figure 26: SHAP output for LightGBM (Multiclass)

Based on the plots, for global feature impact, the NIDS were retrained with the top 10, 5 and 2 most impactful features. Figures 27, 28, 29 and 30 illustrate the accuracy and time required for each experiment using LightGBM.



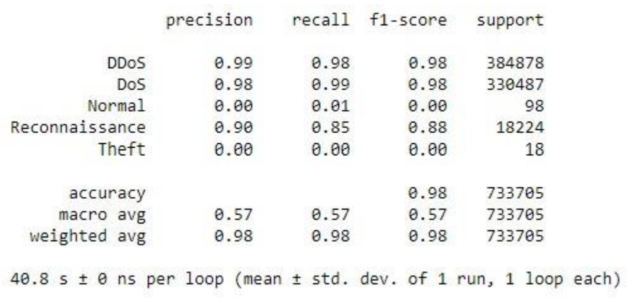
```
                precision    recall  f1-score   support

          DDoS       0.99      0.99      0.99    384878
           DoS       1.00      0.99      0.99    330487
        Normal       0.00      0.00      0.00        98
Reconnaissance       0.97      0.99      0.98     18224
         Theft       0.00      0.00      0.00        18

      accuracy                           0.99    733705
     macro avg       0.59      0.59      0.59    733705
  weighted avg       0.99      0.99      0.99    733705

59.4 s ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 27: Classification report for LightGBM (all features)



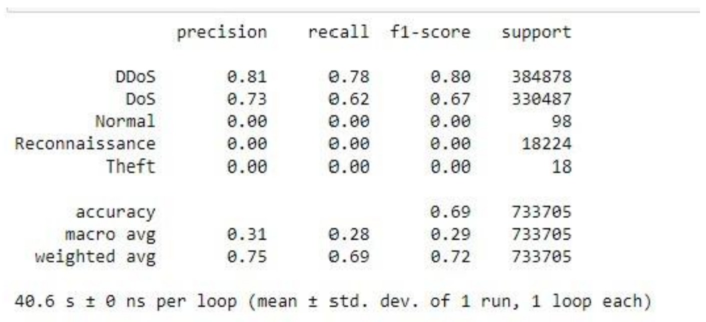
```
                precision    recall  f1-score   support

          DDoS       1.00      1.00      1.00    384878
           DoS       1.00      1.00      1.00    330487
        Normal       0.00      0.00      0.00        98
Reconnaissance       0.95      0.98      0.97     18224
         Theft       0.00      0.00      0.00        18

      accuracy                           1.00    733705
     macro avg       0.59      0.60      0.59    733705
  weighted avg       1.00      1.00      1.00    733705

43.5 s ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 28: Classification report for LightGBM (top 10 features)



```
                precision    recall  f1-score   support

          DDoS       0.99      0.98      0.98    384878
           DoS       0.98      0.99      0.98    330487
        Normal       0.00      0.01      0.00        98
Reconnaissance       0.90      0.85      0.88     18224
         Theft       0.00      0.00      0.00        18

      accuracy                           0.98    733705
     macro avg       0.57      0.57      0.57    733705
  weighted avg       0.98      0.98      0.98    733705

40.8 s ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 29: Classification report for LightGBM (top 5 features)

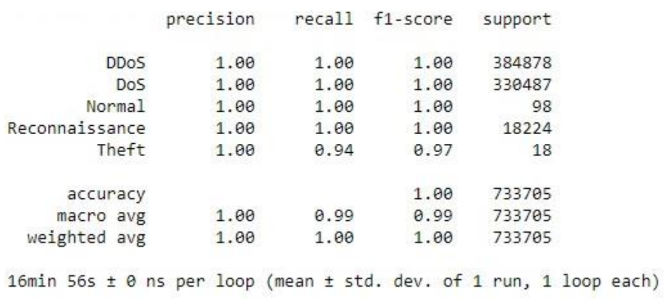


```
                precision    recall  f1-score   support

          DDoS       0.81      0.78      0.80    384878
           DoS       0.73      0.62      0.67    330487
        Normal       0.00      0.00      0.00        98
Reconnaissance       0.00      0.00      0.00     18224
         Theft       0.00      0.00      0.00        18

      accuracy                           0.69    733705
     macro avg       0.31      0.28      0.29    733705
  weighted avg       0.75      0.69      0.72    733705

40.6 s ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 30: Classification report for LightGBM (top 2 features)

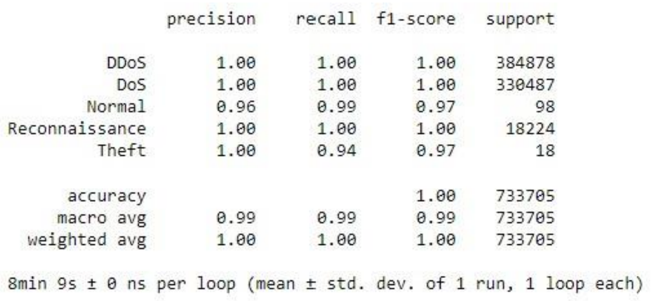
From the above figures it can observed that as the number of features is decreased progressively, the amount of processing time required is decreased as well. The decrease in time is less significant as the number features used is reduced. It is also to be noted that there is no major drop in accuracy until the number of features used is two. In the case of LightGBM it can also be noted from the classification reports and global impact plot that the Normal and Theft classes share many of the same features and seem to be impacted by many features. Hence, the algorithm often confuses between the two and presents incredibly low accuracy for both classes.

XGBoost is the most complex of the algorithms discussed in this report and the benefits of XAI are the most significant when using this algorithm. Figures 31, 32, 33, and 34 illustrate the classification report and time complexity of the XGBoost algorithm as the number of features is progressively reduced.



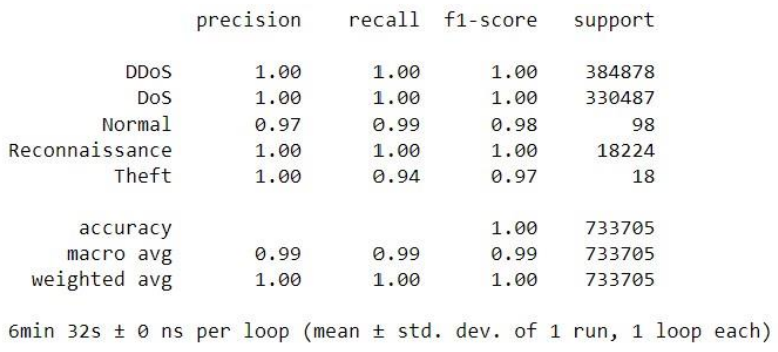
```
                precision    recall  f1-score   support

          DDoS       1.00      1.00      1.00    384878
           DoS       1.00      1.00      1.00    330487
        Normal       1.00      1.00      1.00        98
Reconnaissance       1.00      1.00      1.00     18224
         Theft       1.00      0.94      0.97        18

      accuracy                           1.00    733705
     macro avg       1.00      0.99      0.99    733705
  weighted avg       1.00      1.00      1.00    733705

16min 56s ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 31: Classification report for XGBoost (all features)



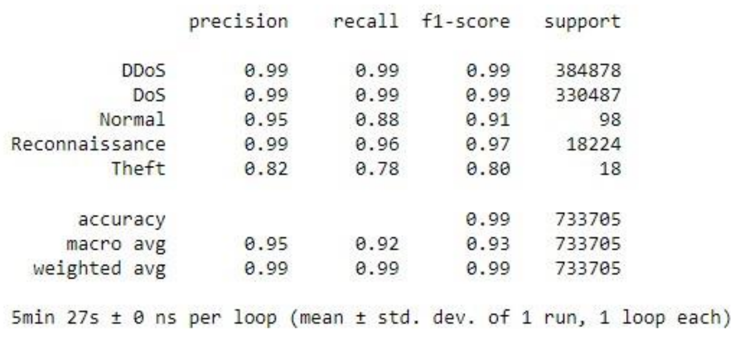
```
                precision    recall  f1-score   support

          DDoS       1.00      1.00      1.00    384878
           DoS       1.00      1.00      1.00    330487
        Normal       0.96      0.99      0.97        98
Reconnaissance       1.00      1.00      1.00     18224
         Theft       1.00      0.94      0.97        18

      accuracy                           1.00    733705
     macro avg       0.99      0.99      0.99    733705
  weighted avg       1.00      1.00      1.00    733705

8min 9s ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 32: Classification report for XGBoost (top 10 features)



```
                precision    recall  f1-score   support

          DDoS       1.00      1.00      1.00    384878
           DoS       1.00      1.00      1.00    330487
        Normal       0.97      0.99      0.98        98
Reconnaissance       1.00      1.00      1.00     18224
         Theft       1.00      0.94      0.97        18

      accuracy                           1.00    733705
     macro avg       0.99      0.99      0.99    733705
  weighted avg       1.00      1.00      1.00    733705

6min 32s ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 33: Classification report for XGBoost (top 5 features)



```
                precision    recall  f1-score   support

          DDoS       0.99      0.99      0.99    384878
           DoS       0.99      0.99      0.99    330487
        Normal       0.95      0.88      0.91        98
Reconnaissance       0.99      0.96      0.97     18224
         Theft       0.82      0.78      0.80        18

      accuracy                           0.99    733705
     macro avg       0.95      0.92      0.93    733705
  weighted avg       0.99      0.99      0.99    733705

5min 27s ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 34: Classification report for XGBoost (top 2 features)

In the case of XGBoost the benefit of reducing the number of features to ten is clearly evident. The processing time is reduced by half while the drop in accuracy is insignificant. As the number of features is reduced the reduction in processing time is less significant but the accuracy still remains at a respectable number, even when using just the top two features. In this case as well the confusion between the Normal and Theft classes exists. At higher number of features the algorithm performs well in mitigating this but as the number of features is reduced it can be observed that these features have small reductions in accuracy.

4.2.2.2 Single class classification

In this section, the benefit of using the most impactful features on model performance will be observed. The Theft subset will be used, and Figure 35 illustrates the impact of each feature using the LightGBM algorithm. From this plot it can also be observed why simpler algorithms do not perform well for the Theft and Normal classes. Features such as the total number of packets per destination IP and number of bytes influence the algorithm both positively and negatively for low values of those features. Thus, simpler algorithms cannot accurately model this complexity.

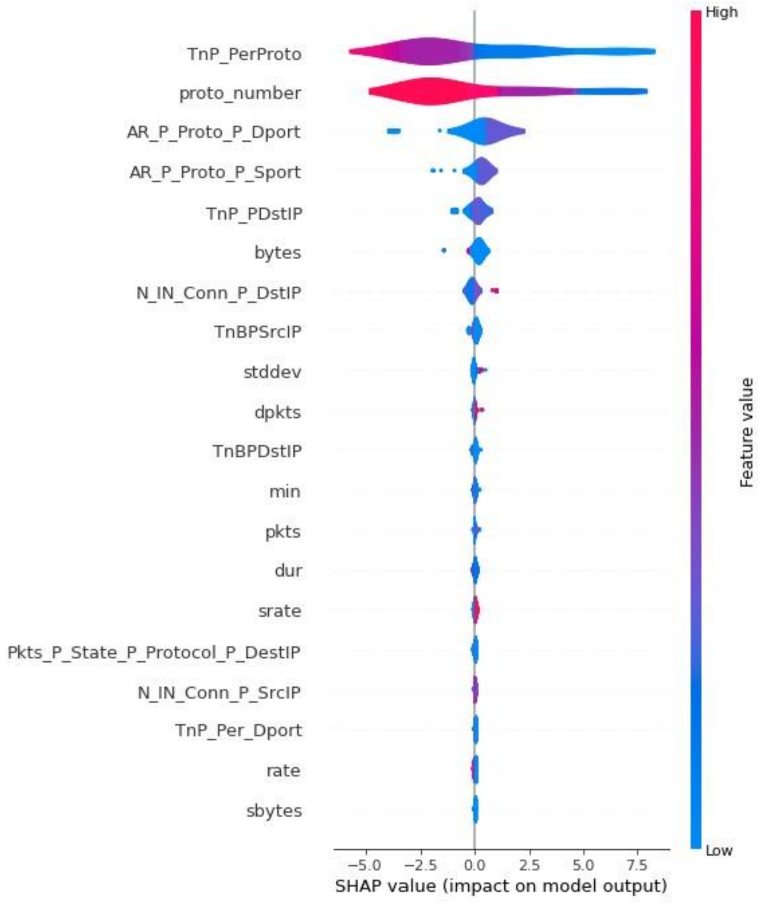
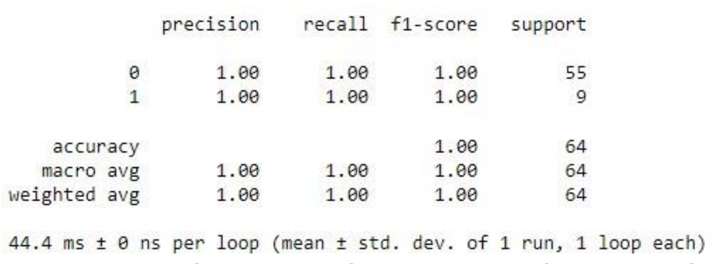


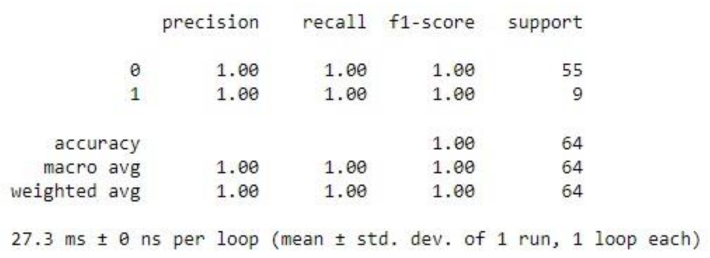
Figure 35: SHAP global explanation for LightGBM (Information Theft)

Based on the above figure, the top 5 and 2 most impactful features were selected to observe any improvement in performance. The classification report for the LightGBM algorithm using all the features is illustrated in Figure 36 while the classification report for the top 5 and 2 features is illustrated in Figures 37 and 38, respectively.



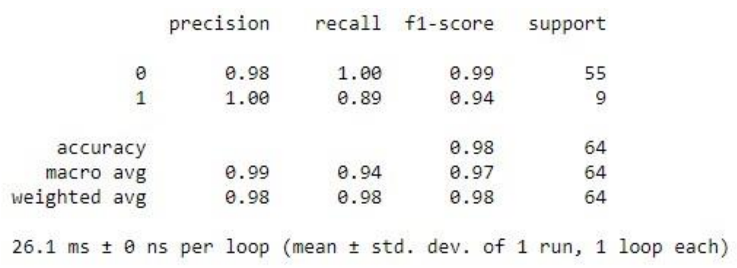
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        55
           1       1.00      1.00      1.00         9

    accuracy                           1.00        64
   macro avg       1.00      1.00      1.00        64
weighted avg       1.00      1.00      1.00        64

44.4 ms ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 36: Classification report for LightGBM (all features) (Theft Identification)



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        55
           1       1.00      1.00      1.00         9

    accuracy                           1.00        64
   macro avg       1.00      1.00      1.00        64
weighted avg       1.00      1.00      1.00        64

27.3 ms ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 37: Classification report for LightGBM (top 5 features) (Theft Identification)

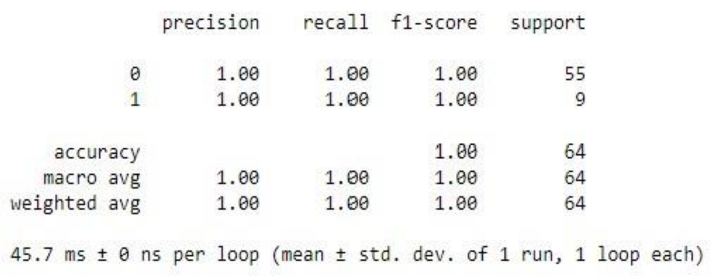


```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        55
           1       1.00      0.89      0.94         9

    accuracy                           0.98        64
   macro avg       0.99      0.94      0.97        64
weighted avg       0.98      0.98      0.98        64

26.1 ms ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 38: Classification report for LightGBM (top 2 features) (Theft Identification)

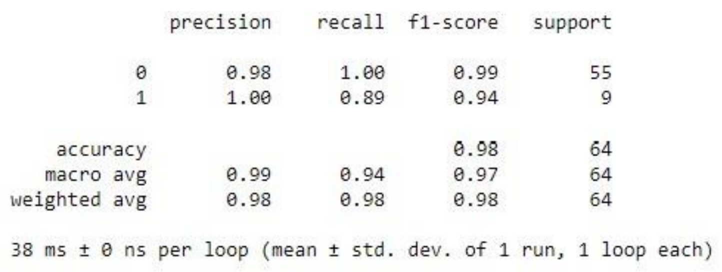
From the above figures it is observed that there is a small reduction in time complexity when the number of features is reduced to five. The reduction in duration is insignificant as well as the decrease in accuracy when the number of features is reduced to two.

The same experiments were conducted using the XGBoost classifier as it is the most complex of the algorithms used in this project. Figure 19 illustrates the impact of each feature using the XGBoost algorithm. The classification report for the XGBoost algorithm using all the features is illustrated in Figure 39 while the classification report for the top 5 and 2 features is illustrated in Figures 40 and 41, respectively.



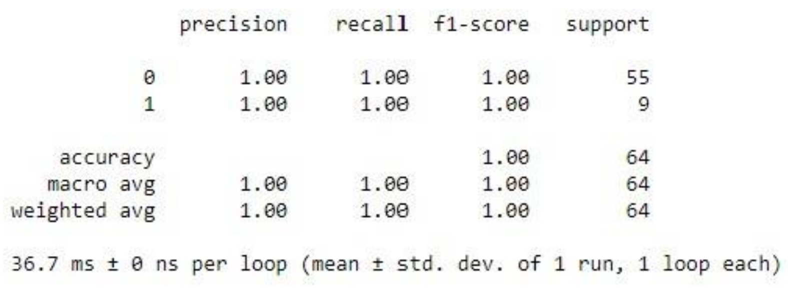
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        55
           1       1.00      1.00      1.00         9

    accuracy                           1.00        64
   macro avg       1.00      1.00      1.00        64
weighted avg       1.00      1.00      1.00        64

45.7 ms ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 39: Classification report for XGBoost (all features) (Theft Identification)



```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        55
           1       1.00      0.89      0.94         9

    accuracy                           0.98        64
   macro avg       0.99      0.94      0.97        64
weighted avg       0.98      0.98      0.98        64

38 ms ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 40: Classification report for XGBoost (top 5 features) (Theft Identification)



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        55
           1       1.00      1.00      1.00         9

    accuracy                           1.00        64
   macro avg       1.00      1.00      1.00        64
weighted avg       1.00      1.00      1.00        64

36.7 ms ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)
```

Figure 41: Classification report for XGBoost (top 2 features) (Theft Identification)

The above figures represent the same trend of reducing time complexity while maintaining insignificant changes to the accuracy of the model. There is one interesting point to note here, there is a drop in accuracy when going from all features to the top five features but a rise in accuracy when going to the top two features. From the global impact plot, it can be observed that features such as the number of dropped packets and bytes give rise to confusion in the algorithm as both high and low values of these features affect the output both negatively and positively. When all the features are used, the algorithm has enough features to rely on to differentiate between an attack or normal connection but when the number of features is reduced to five, the model makes some errors. If only the two most impactful features are used, the model regains its accuracy as the features that were causing the confusion are removed and the top two features have high predictive power.

5 CONCLUSION

In this report, we have discussed the need and approaches to network intrusion detection systems using various ML algorithms. We have observed the limitations of an IoT environment and the need for simple yet accurate classification models. In addition to this, the report presents results for some novel NIDSs when evaluated on the Bot-IoT dataset. Although these algorithms provide reliable results, they do not provide an insight into the reasoning behind the predictions of the NIDS. Additionally, the computation requirements of the more complex machine learning algorithms and larger datasets can be very taxing. Thus, with the aid of XAI methods it was shown that the interpretability of NIDS can be improved, to increase trust in the system as well as the ability to debug the system. In addition, the computation time and accuracy of NIDs can also be increased by focusing on features that have the most impact on the decisions of the algorithm.

5.1 Evaluation

The objectives in section 1.2 guided the tasks that were conducted in this project. For the first objective, the best-in-class NIDS using machine learning were evaluated along with the some of the most complex machine learning algorithms yet developed. By selecting these papers that showed great accuracy and a range of algorithms and dataset permutations, the efficacy of using XAI to improve upon these systems could be demonstrated.

For the second objective, the two most popular XAI techniques were selected to show how the methods available in XAI differ in their capability and complexity. These methods were used to show why some algorithms perform better and why certain features are more important than others. These methods illustrate what drives the predictions of an algorithm and by analysing the results of these methods, debugging can be performed to remove features that negatively influence the output.

Finally, by having a better understanding of the impact of each feature, large datasets can be trimmed down to improve the time complexity of complex machine learning algorithms with no significant loss in accuracy. This becomes especially useful in an IoT environment where resources and time are limited.

5.2 Future Work

There are multiple avenues that can be expanded upon by building upon the work done for this project. The first point of exploration can be to link the theory behind network security with the explanations provided by XAI techniques. This can be done to verify that the features that the XAI methods rely on are backed up by the theory of networking and network security. Next, more complex algorithms and especially, NIDS based on neural networks can be evaluated. Deep Learning remains a black box even though it has seen widespread adoption in multiple fields. With XAI the decisions of these black boxes can be better understood and potentially even optimised. In further work it could also be evaluated to see if the variation in datasets could be normalised in an ideal way that contributes to the task of classification. By developing an understanding of the range of values for a feature that most impact the output, the range of values for every feature can be reduced, thus reducing the amount of pre-processing that would be needed as well as reducing the size of the dataset.

Finally, newer XAI methods such as GraphLIME, SHAPFlow and ASV can be evaluated to gain deeper insights into the decisions behind complex algorithms as well making explanations more user friendly and accessible to non-professionals.

REFERENCES

[1] H. N. Saha, A. Mandal, and A. Sinha, ‘Recent trends in the Internet of Things’, in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2017, pp. 1–4. doi: 10.1109/CCWC.2017.7868439.

[2] Khraisat, Gondal, Vamplew, Kamruzzaman, and Alazab, ‘A novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks’, *Electronics (Basel)*, vol. 8, no. 11, p. 1210, Oct. 2019, doi: 10.3390/electronics8111210.

[3] J. Li, Y. Qu, F. Chao, H. P. H. Shum, E. S. L. Ho, and L. Yang, ‘Machine Learning Algorithms for Network Intrusion Detection’, in *Intelligent Systems Reference Library*, vol. 151, 2019, pp. 151–179. doi: 10.1007/978-3-319-98842-9_6.

[4] C. L. Ignat, Q. V. Dang, and V. L. Shalin, ‘The Influence of Trust Score on Cooperative Behavior’, *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 4, Sep. 2019, doi: 10.1145/3329250.

[5] L. A. Maglaras *et al.*, ‘Cyber security of critical infrastructures’, *ICT Express*, vol. 4, no. 1, pp. 42–45, Mar. 2018, doi: 10.1016/j.icte.2018.02.001.

[6] B. Reis, E. Maia, and I. Praça, ‘Selection and Performance Analysis of CICIDS2017 Features Importance’, in *Foundations and Practice of Security*, 2020, pp. 56–71.

[7] P. Hase and M. Bansal, ‘When Can Models Learn From Explanations? A Formal Framework for Understanding the Roles of Explanation Data’, in *Proceedings of the First Workshop on Learning with Natural Language Supervision*, 2022, pp. 29–39. doi: 10.18653/v1/2022.lnls-1.4.

[8] M. Turek, ‘Explainable Artificial Intelligence’, Arlington, Aug. 2016. Accessed: Sep. 08, 2022. [Online]. Available: https://www.darpa.mil/program/explainable-artificial-intelligence

[9] S. M. Lundberg, P. G. Allen, and S.-I. Lee, ‘A Unified Approach to Interpreting Model Predictions’, in *Advances in Neural Information Processing Systems*, 2017, vol. 30. Accessed: Sep. 06, 2022. [Online]. Available: https://github.com/slundberg/shap

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, ‘“Why Should I Trust You?”: Explaining the Predictions of Any Classifier’, in *NAACL-HLT 2016 - 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Demonstrations Session*, Feb. 2016, pp. 97–101. doi: 10.48550/arxiv.1602.04938.

[11] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek, ‘Explainable AI Methods - A Brief Overview’, in *xxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*, A. Holzinger, R. Goebel, R. Fong, T. Moon, K.-R. Müller, and W. Samek, Eds. Cham: Springer International Publishing, 2022, pp. 13–38. doi: 10.1007/978-3-031-04083-2_2.

[12] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, ‘RDTIDS: Rules and Decision Tree-Based Intrusion Detection System for Internet-of-Things Networks’, *Future Internet*, vol. 12, no. 3, p. 44, Mar. 2020, doi: 10.3390/fi12030044.

[13] J. L. Leevy, J. Hancock, T. M. Khoshgoftaar, and J. M. Peterson, ‘IoT information theft prediction using ensemble feature selection’, *J Big Data*, vol. 9, no. 1, p. 6, Dec. 2022, doi: 10.1186/s40537-021-00558-z.

[14] H. J. Liao, C. H. Richard Lin, Y. C. Lin, and K. Y. Tung, ‘Intrusion detection system: A comprehensive review’, *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, Jan. 2013, doi: 10.1016/J.JNCA.2012.09.004.

[15] S. Mukkamala, A. Sung, and A. Abraham, ‘Cyber security challenges: Designing efficient intrusion detection systems and antivirus tools’, *Vemuri, V. Rao, Enhancing Computer Security with Smart Technology.(Auerbach, 2006)*, pp. 125–163, 2005.

[16] A. L. Buczak and E. Guven, ‘A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection’, *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1153–1176, Apr. 2016, doi: 10.1109/COMST.2015.2494502.

[17] E. Conrad, S. Misenar, and J. Feldman, ‘Domain 7: Security operations’, in *Eleventh Hour CISSP®*, Syngress, 2017, pp. 145–183. doi: 10.1016/B978-0-12-811248-9.00007-3.

[18] J. Burton, I. Dubrawsky, V. Osipov, C. Tate Baumrucker, and M. Sweeney, Eds., ‘Introduction to Intrusion Detection Systems’, in *Cisco Security Professional’s Guide to Secure Intrusion Detection Systems*, Burlington: Elsevier, 2003, pp. 1–38. doi: 10.1016/B978-193226669-6/50021-5.

[19] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, ‘Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset’, *Future Generation Computer Systems*, vol. 100, pp. 779–796, Nov. 2018, doi: 10.48550/arxiv.1811.00701.

[20] P. Amini, M. A. Araghizadeh, and R. Azmi, ‘A survey on Botnet: Classification, detection and defense’, in *Proceedings - 2015 International Electronics Symposium: Emerging Technology in Electronic and Information, IES 2015*, Jan. 2016, pp. 233–238. doi: 10.1109/ELECSYM.2015.7380847.

[21] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, ‘An Evaluation Framework for Intrusion Detection Dataset’, in *2016 International Conference on Information Science and Security (ICISS)*, Dec. 2016, pp. 1–6. doi: 10.1109/ICISSEC.2016.7885840.

[22] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita, ‘Network attacks: Taxonomy, tools and systems’, *Journal of Network and Computer Applications*, vol. 40, no. 1, pp. 307–324, Apr. 2014, doi: 10.1016/J.JNCA.2013.08.001.

[23] ‘1.9. Naive Bayes — scikit-learn 1.1.1 documentation’. https://scikit-learn.org/stable/modules/naive_bayes.html (accessed Jun. 27, 2022).

[24] Q.-V. Dang, ‘Improving the performance of the intrusion detection systems by the machine learning explainability’, *International Journal of Web Information Systems*, vol. 17, no. 5, pp. 537–555, Sep. 2021, doi: 10.1108/IJWIS-03-2021-0022.

[25] F. Divina, A. Gilson, F. Goméz-Vela, M. G. Torres, and J. F. Torres, ‘Stacking Ensemble Learning for Short-Term Electricity Consumption Forecasting’, *Energies 2018, Vol. 11, Page 949*, vol. 11, no. 4, p. 949, Apr. 2018, doi: 10.3390/EN11040949.

[26] M. Naghiloo, J. J. Alonso, A. Romito, E. Lutz, and K. W. Murch, ‘Information Gain and Loss for a Quantum Maxwell’s Demon’, *Phys Rev Lett*, vol. 121, no. 3, p. 030604, Jul. 2018, doi: 10.1103/PHYSREVLETT.121.030604/FIGURES/4/MEDIUM.

[27] J. Brownlee, ‘Classification And Regression Trees for Machine Learning’, *Machine Learning Mastery*, Apr. 08, 2016. https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/ (accessed Sep. 06, 2022).

[28] P. Gupta, ‘Decision Trees in Machine Learning’, *Towards Data Science*, May 17, 2017. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (accessed Sep. 06, 2022).

[29] T. Chen and C. Guestrin, ‘XGBoost: A Scalable Tree Boosting System’. doi: 10.1145/2939672.

[30] ‘XGBoost’, *GeeksforGeeks*, Jul. 11, 2022. https://www.geeksforgeeks.org/xgboost/ (accessed Sep. 06, 2022).

[31] G. Ke *et al.*, ‘LightGBM: A Highly Efficient Gradient Boosting Decision Tree’, *Adv Neural Inf Process Syst*, vol. 30, 2017, Accessed: Sep. 06, 2022. [Online]. Available: https://github.com/Microsoft/LightGBM.

[32] ‘LightGBM (Light Gradient Boosting Machine)’, *GeeksforGeeks*, Dec. 22, 2021. https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/?ref=gcse (accessed Sep. 06, 2022).

[33] M. Collins, R. E. Schapire, and Y. Singer, ‘Logistic Regression, AdaBoost and Bregman Distances’, *Mach Learn*, vol. 48, no. 1, pp. 253–285, 2002, doi: 10.1023/A:1013912006537.

[34] ‘Explainable AI | IBM’. https://www.ibm.com/watson/explainable-ai (accessed Jun. 27, 2022).

[35] ‘Explainable AI’. https://www.pwc.co.uk/services/risk/insights/explainable-ai.html (accessed Jun. 27, 2022).

[36] A. Barredo Arrieta *et al.*, ‘Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI’, *Information Fusion*, vol. 58, pp. 82–115, Jun. 2020, doi: 10.1016/J.INFFUS.2019.12.012.

[37] S. Mane and D. Rao, ‘Explaining Network Intrusion Detection System Using Explainable AI Framework’, Mar. 2021, doi: 10.48550/arxiv.2103.07110.

[38] M. E. Morocho-Cayamcela, H. Lee, and W. Lim, ‘Machine Learning for 5G/B5G Mobile and

Wireless Communications: Potential, Limitations, and Future Directions', *IEEE Access*, vol. 7, pp. 137184–137206, 2019, doi: 10.1109/ACCESS.2019.2942390.

[39] 'Going on the Defensive: Intrusion-Detection Systems | Types of IDSs |', *InformIT*. https://www.informit.com/articles/article.aspx?p=29601 (accessed Jun. 27, 2022).