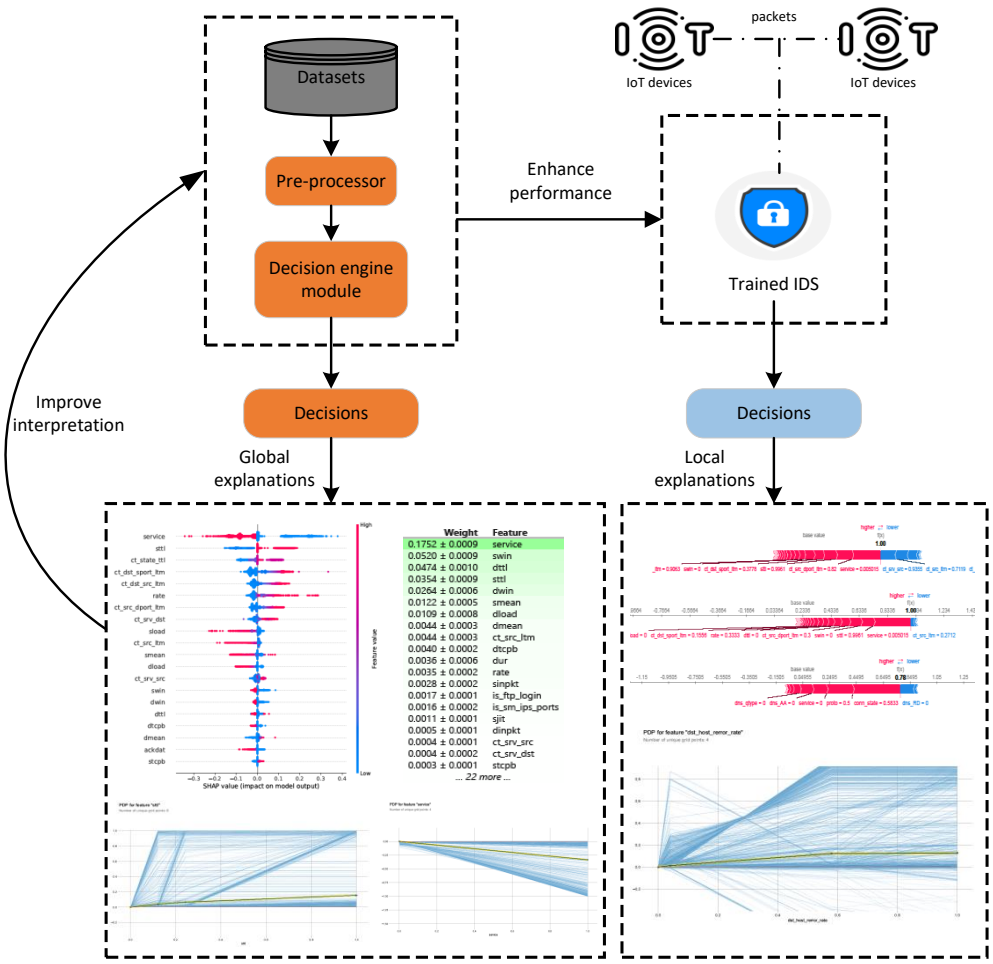
Graphical Abstract

An Explainable Deep Learning-enabled Intrusion Detection Framework in IoT networks

Nam Pham, Nour Moustafa, and Benjamin Turnbull, Marwa Keshk, and Albert



Highlights

An Explainable Deep Learning-enabled Intrusion Detection Framework in IoT networks

Nam Pham, Nour Moustafa, and Benjamin Turnbull, Marwa Keshk, and Albert

- We propose a novel SPIP framework that provides global and local explanations to IDS regardless of the underlying algorithm used.
- We use the novel set of input features extracted by the proposed framework to train and evaluate the AI-based IDS. The utilisation of a customized set of input features increase the performance and reduce the training time as well as the total detection time of the IDS.
- We demonstrate the proposed framework's ability to effectively enhances the interpretability and explainability of cyber defence systems.

An Explainable Deep Learning-enabled Intrusion Detection Framework in IoT networks

Nam Pham, Nour Moustafa, and Benjamin Turnbull, Marwa Keshk, and Albert

[a]*School of Engineering and Information Technology, University of New South Wales, Canberra, 2612, ACT, Australia*

Abstract

Although the field of eXplainable Artificial Intelligence (XAI) is garnering significant interest, its implementation within cyber security applications still needs further investigation to understand its effectiveness in discovering attack surfaces and vectors. In cyber defence, especially anomaly-based Intrusion Detection Systems (IDS), the emerging applications of machine/deep learning models require the interpretation of the models' architecture and the explanation of models' prediction to examine how cyber-attacks would occur. This paper proposes the novel SPIP framework for AI-based IDS in Internet of Things (IoT) networks. We have developed an IDS using a Long Short-Term Memory (LSTM) - a variant of Recurrent Neural Network (RNN) model, which identifies cyber-attacks and explains the model's decisions. This uses a novel set of input features extracted by SPIP framework to train and evaluate the LSTM model. The framework was validated by the NSL-KDD, UNSW-NB15 and TON_IoT datasets. Based on the results obtained from the experiments, the SPIP framework achieves high performance in terms of detection accuracy, processing time, and high interpretability of data features and model outputs when compared with other peer techniques. The proposed framework has the potential to assist administrators and decision-makers in understanding complex attack behaviour.

Keywords: Intrusion Detection System (IDS), Artificial Intelligence (AI), Explainable AI (XAI), Internet of Things (IoT), Cyber defence

1. Introduction

Emerging technologies such as the Internet of Things (IoT), Smart Cities, mobile devices and the Internet have fundamentally changed how modern society operates. However, such technologies are increasingly complex and pose unique challenges to secure, in addition to increasingly serious impacts [1]. There are several underlying reasons for this; such systems are comprised of a wide variety of low-cost devices, and generate significant data across multiple networks, protocols and for different use-cases [2]. Due to this proliferation of heterogeneous devices, IoT networks generate high-dimensional and multimodal data, which requires the capability of analysing big data. Artificial Intelligence (AI) is one paradigm increasingly sought after to help solve these issues, as AI technologies, especially Machine Learning (ML) and Deep Learning (DL), have been utilised across industries and achieve excellent performance with large scales of data.

IoT networks have unique properties that require different approaches to defence than used in traditional corporate environments. To prevent malicious activities and protect IoT networks, multiple security mechanisms have been proposed. Traditional approaches, such as anti-malware, firewalls, user authentication and data encryption, are all well-known, and each of them fits different purposes [3]. However, traditional mechanisms are less effective as they are less flexible and dynamic in the face of the rapid growth of attack techniques [4]. Intrusion Detection Systems (IDSs) are a mature and prominently used cybersecurity detection control that can identify diverse cyber attacks and even zero-day attacks [5]. Due to the application of AI techniques, AI-based IDSs can achieve high performance with benchmark datasets. Deep learning techniques can analyse complex data and learn from previous attack patterns in the dataset to detect zero-day attacks. However, such techniques typically have a high false-positive rate and their decisions are opaque to users [6].

Anomaly-based detection is important to discover unknown attacks and protect information technology systems. Therefore, many ML algorithms have been proposed to design efficient IDSs with high accuracy and low false-positive rates. Moreover, the need to explain the functioning and predictions of ML-based IDS arises as different types of users are benefited from understanding the root cause of intrusion detection. Therefore, the field of explainable artificial intelligence (XAI) has been developed in recent years to address this concern. XAI illuminates the black-box model by providing

explanations on their functioning and predictions. One of the key limitations of existing deep learning-based IDS models is the discovery of zero-day attacks and interpretation of how the models successfully discovered them [4, 6]. Current signature-based IDSs rely on known attacks signatures and suffer from high false negative and false positive rate, hence limiting their ability to detect unknown attacks and hindering their practical use [7]. Such systems are known to be efficient, but have a high probability of a motivated attacker being able to bypass identification. Additionally, signature-based IDS platforms are unable to discover new or zero-day attacks - they are limited to signatures for attacks that have been previously analysed and are within the database. Moreover, researchers are encouraged to work on interpreting machine learning models as the need to explain their decisions has been written in regulations [8].

This paper intends to promote the interpretability and explainability of IDSs. To do this, we propose the novel SPIP (S: SHapley Additive exPlanations, P: Permutation Feature Importance, I: Individual Conditional Expectation, P: Partial Dependence Plot) framework. The proposed framework provides both global and local explanations to serve different purposes. The generated explanations are intuitive and useful for lay users, security experts and researchers. Its ultimate aim is to promote the interpretability and explainability of IDSs, hence enhancing the performance of cyber defence systems.

The major contributions of this paper are structured as follows:

- We propose the novel SPIP framework that generates global and local explanations to the decisions of IDS regardless of the underlying algorithms. This framework promotes the interpretation and explainability of the IDS, hence building users' trust in the IDS. Moreover, this framework support experts in analyzing the IoT datasets and characteristics of various cyber-attacks, which would enhance the performance of cyber defence systems.

- We employ a set of the most important features extracted by the proposed framework and the original set of input features to train the AI-based IDS and compare the performance. The utilisation of a customized set of input features increases the performance and reduce the training time as well as the total detection time of the IDS.

The remainder of this paper is organized as follows. Section 2 discusses

the related works. The methodology of this framework is described in section 3, including Long Short-Term Memory (LSTM), Shapley Additive Explanations (SHAP), Permutation Feature Importance (PFI), Individual Conditional Expectation (ICE) and Partial Dependence Plot (PDP). Section 4 proposes SPIP framework. Section 5 presents the experiments and the results' discussions. Finally, section 6 summarises this work and discusses the future research direction.

2. Related Work

Intrusion Detection System (IDS) is a prominent control in cyber defence, and used extensively to effectively detect and prevent cyberattacks. IDS was first introduced by Denning [9] in 1987. There are various approaches to divide IDS into different categories; with network-based IDS and host-based IDS being the two main types, which are categorized based on places of deployment. Whilst network-based IDS detect attacks by capturing and analysing network traffic between devices, host-based IDS monitor processes and system events on the host to identify malicious activities [10]. Moreover, an IDS can be design based on two different detection methods, including signature-based and anomaly-based. Whilst signature-based IDS rely on a database consisting of known attacks' signatures to detect them, anomaly-based IDS identify malicious activities by comparing them to a pre-built baseline of normal behaviours [11]. Hybrid IDS employ both methodologies.

Explainable Artificial Intelligence (XAI) was first introduced in the 1980s [12]. However XAI has seen less notice as AI's growth has focused on predictive performance. In recent years, the need of XAI is increasing as AI models are widely used in critical sectors where explanations are required for any decisions. Morevover, XAI can enhance the use of AI models as it aids researchers in model debugging, data collection, trust building and human decision-making. There are two main approaches of XAI methods; intrinsic XAI, and post-hoc XAI. Intrinsic XAI approaches provide explanations to the structure and functioning of the model, which limit its application to a specific type of AI techniques. Post-hoc approaches provide explanations to the model's final decision by analysing the set of input and the set of output; thus, it is applicable to any type of models. XAI methods are also divided into global or local. Global XAI methods aim to explain the general characteristics of the models by analyzing all of its decisions. Whilst local XAI

methods focus on giving explanations to individual decisions that the model has made.

The large scale development of the Internet of Things (IoT) has caused significant new challenges in the areas of cyber defence. IoT deployments are Large number of Internet-connected devices generates huge amount of traffic and digital data generated, which challenges existing solutions' ability to detect and prevent cyberattacks [13]. Therefore, AI-based IDS is a prominent method to deal with such large-scale data [14]. A huge body of researches have been focused on finding the best IDS in IoT enviroments [13]. Additionally, XAI methods would be applied in this field to enhance the performance, interpretability and explanability of AI-based intrusion detection models in the heterogeneous IoT networks.

XAI methods have been created and applied to many applications and fields. The most popular works focus on model explanations for computer vision [15], Natural Language Processing (NLP) [16] and voice analysis [17]. Of particular note, the authors of [16] developed and deploy a framework, LIME, in the field of NLP and [18] used SHAP to give explanations in the field of biology. In contrast, the utilization of XAI methods in the field of cyber defence is very new and rare.

In the field of cyber defence, AI is mainly utilised to build a model for automated detection. Numerous researchers have shown the prominence of AI-based IDS. Of note amongst these, Rahul et al. [14] used a benchmark dataset to compare the performance of several classical ML algorithms and Deep Neural Network (DNN) models. In this experiment, AI-based IDSs obtained excellent performance, particularly those built upon DNNs outperformed other classical ML algorithms. Andalib and Vakili [19] proposed an IDS combining three different types of ML techniques. Based on their observations, ML models have a short training time and only need low computational resources [19]. Despite the prominent performance, the AI-based intrusion detection models are becoming more complex and their functions and outcomes are difficult to interpret. This makes them into black-box models and hinders their applications in critical sectors including cyber defence.

In order to confront this problem, several works related to explainable AI-based IDS have emerged. Islam et al. [20] proposed a method to gather and utilize domain knowledge to automate the defence response and improve the explainability of the IDS model. In the experiment, domain knowledge (including fundamental principle of Confidentiality, Integrity, and Availability (CIA)) is instilled into the ML models. This was analyzed with the CIDS2017

dataset. Results from this experiment highlighted an increase in generalizability and explainability of the models, which promoted the utilization of AI-based IDS in IoT networks. Marino et al. [21] produced explanations for incorrect samples using an adversarial approach. This approach works by finding the minimum modification to correctly label the incorrect records then visualize the most critical features to explain the model's wrong decision. Le et al. [22] propose a IDS built upon an RNN and explain the outcomes of the model to generate Software Defined Networking (SDN) flow rules. They use a linear regression model as a local surrogate describes in LEMNA [23], the k most relevant features are chosen to generate network access control policies.

3. Proposed Methodology

This section introduces the underlying mathematical models of the IDS and SPIP framework. Long Short-Term Memory (LSTM) - a modified version of Recurrent Neural Networks (RNN) is utilised to analyse complex and large-scale datasets to detect cyber attacks. The proposed framework applies various XAI methods to produces reasonable and intuitive explanations that would be used to study the functioning of the model, the quality of datasets and the characteristics of many attack classes.

3.1. Recurrent Neural Networks

Before discussing LSTM, we will briefly describe RNN to understand how LSTM is more advanced than RNN. RNN consists of multiple layers with a feedback loop, which makes RNN effective in analysing sequence data. Each neuron in RNN has a short-term memory to pass information to itself in the future. Combining the internal memory and feedback loop helps to link the current output with the preceded output, propagating the data from the past onward to the present time. In other words, information travels through time in RNNs; thus, the weights between neurons are updated by using backpropagation through time (BPTT) [24]. However, BPTT suffers from vanishing and exploding gradient problems as Bengio et al. demonstrated in [25]. Therefore, RNN can only retain information for a few time steps and it fails to capture long-term data dependencies. Due to the vanishing gradient problem, RNN cannot learn the context information across a long time and become inefficient in long data sequence analysis.

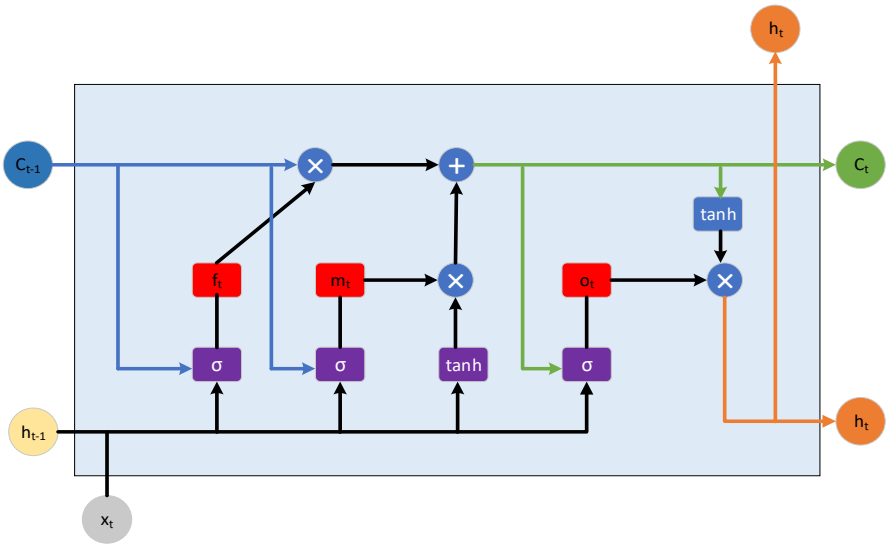


Figure 1: Long Short-Term Memory

3.2. Long Short-Term Memory

Long-Short Term Memory (LSTM) is a modified version of RNN that was first introduced in 1997 [26]. Technically, LSTM prevents the vanishing gradient problem by employing memory cells and gates. The LSTM cell contains three gates including memory, forget and output gate. The gate mechanism helps LSTM capture long-term dependencies and allows the retention of information for a long period of steps. The gates control the information that flows into and out of the cell state. The memory gate controls the flow of new information, the forget gate deletes information in the cell state and the output gate regulates the output flow of the internal memory cell state.

Firstly, the forget gate decides which information to delete or keep in the cell state by a sigmoid activation function. The sigmoid layer considers h_{t-1}, x_t and c_{t-1} and generates a real number in the range of 0-1 for each number in the cell state c_{t-1}, as shown in equation (1). For example, 0 indicates "delete all information" and 1 indicates "keep all information". Secondly, the cell processes new information and decides which information to be stored in the cell state. The sigmoid layer chooses information to update as shown in equation (2) and the *tanh* activation function generate a vector of new values. The update to the cell state is the combination of these two values,

including information that is decided to be stored in the cell state and the vector of new values generated by *tanh* activation function.

Thirdly, after computing values in the first two steps, the new cell state c_t is created in equation (4). It may keep some information from c_{t-1} (decided by f_t) and add with new information calculated in the second step. Lastly, the output flow of information is regulated using equation (5). Output gate also uses a sigmoid activation function. The cell state c_t goes through *tanh* function and then it is multiplied by the result from the sigmoid layer.

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{1}$$

$$m_t = \sigma(W_{xm}x_t + W_{hm}h_{t-1} + W_{cm}c_{t-1} + b_m) \tag{2}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{3}$$

$$c_t = f_t{}^{\odot}c_{t-1} + m_t{}^{\odot}tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{4}$$

$$h_t = o_t \odot tanh(c_t) \tag{5}$$

In the above equations, m_t, f_t, and o_t correspond to values of memory, forget, and output gate at time t, respectively. x_t, h_t, and c_t represent the input layer, hidden layer, and cell state at time t. b terms indicate the bias vectors and W terms indicate weight matrices. σ and *tanh* are the sigmoid and *tanh* activation functions, $\odot$ is the element-wise multiplication.

3.3. Explainable Artificial Intelligence (XAI) methods

Model-agnostic XAI techniques can be classified into 2 categories, including local and global explanation methods. Local interpretability would use only one data record to generate the explanation for a single model's decision. This type of explanation is useful for users and people that are affected by the model's decision in helping them to trust the model and verify a single decision that affects them. Global interpretability would take the whole dataset as input to generate explanations, which demonstrates the overall structure and functioning of the black-box model. This technique is especially useful for researchers and developers to analyse and detect any bias to improve the model's performance. Moreover, in the case of IDS deployments, global XAI methods are important in studying and analysing different types of cyberattacks based on features of network traffics.

In this paper, different XAI methods are used to study the functioning and predictions of the proposed IDS both locally and globally. On one hand,

we apply Shapley Additive Explanations (SHAP) [27] and Individual Conditional Expectation (ICE) [28] techniques for visualising individual prediction made by the model. On the other hand, Permutation Feature Importance (PFI) [29], SHAP [27] and Partial Dependence Plot (PDP) [30] are utilised as global XAI methods to find the customized set of relevant input features, which can be used to explain the functioning of the proposed model as a whole. Those XAI techniques are discussed and detailed as follows.

3.3.1. Shapley Additive Explanations (SHAP)

SHAP was proposed by Lundberg et al. [27] based on the game theoretically optimal Shapley values in [31]. Therefore, SHAP has a strong theoretical foundation. SHAP is a combination of Local Interpretable Model-agnostic Explanations (LIME) [32] and Shapley value [31]. Thus, we will briefly introduce LIME and Shapley's value before analysing SHAP.

LIME derives from the concept of local surrogate models. In the concept of surrogate model, an interpretable model is chosen to be trained and approximate the predictions of the black-box model that need to be explained. Different from global surrogate models, LIME's key concept is fitting an interpretable model around a specific instance to visualise significant features of that data instance in contribution towards an attack or normal prediction of the model. In LIME, the surrogate model can be of any type of interpretable model as long as it is a good approximation of the black-box model's predictions locally [32]. In LIME, the explanation of an observation x is computed by the following formula:

$$E(x) = \underset{g \in G}{\operatorname{argmin}} \left\{ L\left(f, g, d^x\right) + \Omega(g) \right\} \tag{6}$$

In equation (6), g denotes an interpretable model and G represents the class of all possible explanation models. These can be any type of interpretable ML whose results can be understood by a human. The original model is denoted by f and the distance between the sample and original data is presented as d^x. Thus, L(f, g, d^x) is the loss function that measures the difference in decisions of the surrogate model and the original model. LIME aims to calculate this loss function L. Additionally, $\Omega(g)$ denotes the complexity of model g and it is defined by human. For example, $\Omega(g)$ can be point for the depth of the decision tree model [32].

The LIME algorithm begins by creating non-existing data samples around the chosen instance of interest x. Then a local surrogate model g is trained on

this data samples as the loss function L and the complexity Ω are minimized. Eventually, a local model denoted by *E(x)* can explain the given decision of instance x. The Shapley value is a technique in the cooperative game theory method. The value was named after its inventor Lloyd Shapley as he introduced it in [31]. Shapley value has a strong theoretical foundation to measure the importance of each player in a collaborative game and indicate how much each player has contributed to the success. Specifically, each feature's value has its own Shapley value that represents its average contribution of to the model's outcome. This value is computed and summed over all combinations of feature value for a given set of features and calculated predictions:

$$\phi_j(val) = \sum_{S \subseteq M \backslash \{j\}} \frac{|S|!\,(m - |S| - 1)!}{m!} [val(S + j) - val(S)], \quad j = 1 \ldots m \quad (7)$$

where:

- S is the subset of features used and —S— is its number.
- M is the complete set of features and m is its size.
- j is the j^{th} feature the vector of the feature values of the instance to be explained
- *val(S)* and *val(S+j)* are the functions assigning only subset of features S and subset of features S with j^{th} feature present, respectively.

The Shapley value has four meaningful properties; **Efficiency, Symmetry, Dummy** and **Additivity**. The property of **Efficiency** states that the feature contributions must add up to the prediction for a single instance which equals to the difference between the actual output and the average prediction:

$$\sum_{i=1}^{m} \phi_i(val) = val(M) \quad (8)$$

Symmetry: Considering two different features i^{th} and k^{th} in the subset of features S, so that:

$$val(S + i) = val(S + k), \quad S \subseteq M \backslash \{i, k\} \quad (9)$$

Equation 11 indicates that two features i^{th} and k^{th} contribute equally to all possible coalitions. Therefore, the contribution of two features should be equal and the function ϕ is symmetric:

$$\phi_i(val) = \phi_k(val) \tag{10}$$

Dummy: if contribution of a given feature does not change the prediction, its Shapley value should be equal to zero:

$$val(S + i) = val(S), \quad S \subseteq M \backslash \{i\} \tag{11}$$

then:

$$\phi_i(val) = 0 \tag{12}$$

Additivity: the gain from combining two functions *val* and *val′* is equal to sum of individual gains from each function for every feature ith:

$$\phi_i(val + val') = \phi_i(val) + \phi_i(val') \tag{13}$$

SHAP was first introduced in 2017 [27] as a unified approach for explaining black-box models' predictions. SHAP is a type of relevance-based approach that measures the importance of each feature by computing its contributions to the model's decision. SHAP can connect LIME and Shapley value by representing Shapley value explanation as a linear model. SHAP value can be calculated using individual data instances. For example, explanation for an instance x can be calculated using the following formula:

$$g\left(z'\right) = \phi_0 + \sum_{i=1}^{m} \phi_i z'_i, \quad z' \in \{0,1\}^m \tag{14}$$

where:

- m is the size of all possible coalitions.
- g is the explanation model.
- z' is simplified features. The 1 in z′ indicates that features in the new data are the same as those of the original data - the instance z.
- ϕ_i is the Shapley value for feature $i^{th}$of instance x, hence indicates the impact of feature i on the model's prediction.

A key advantage of SHAP over LIME is that effects are distributed fairly in SHAP yet behaviour of the model is assumed to be linear locally in LIME. Specifically, LIME weighs the instances based on their distance to the original instance. Meanwhile, instances in SHAP are weighted so that the coalition is in the Shapley value estimation. Shapley compliant weighting is calculated by a method called SHAP kernel [27]:

$$\pi_x(z') = \frac{(m-1)}{\binom{m}{|z'|}\,|z'|\,(m-|z'|)} \tag{15}$$

where:

- m is the size of all possible coalitions.
- $—z'—$ is the total number of present features in instance z'

To conclude, the process of calculating SHAP values for an instance x is divided into five steps:

1. Sample data $z'_k \in \{0,1\}^m$, k $\in \{1 \ldots \mathrm{K}\}$
2. Get prediction for each z'_k. This requires 2 steps, including converting each z'_k to the original feature space, and then applying model $f(h_x(z'_k))$. $f(x)$ represents original model, and $h_x(z') = z$ where h_x: $\{0,1\}^m \rightarrow \mathbb{R}$ is used to get valid data instances from coalitions of feature values.
3. Calculate the weight for each z'_k using equation (15)
4. Train the linear model g by optimizing the loss function L and using Z as the training data:

$$L(f, g, \pi_x) = \sum_{z' \in Z} \left[f\left(h_x(z')\right) - g(z')\right]^2 \pi_x(z') \tag{16}$$

5. Return ϕ_k as Shapley values and coefficients from the linear model calculated in equation (16).

3.3.2. Permutation Feature Importance (PFI)

PFI is another type of relevance-based model-agnostic method that computes the changes in prediction error of a fitted model when permuting the input feature values. This technique shows the importance of each feature by breaking the relationship between the feature and the desired output. Therefore, the model's error would increase if the feature of interest is considered as

being important by the model. PFI was first introduced in 2001 by Breiman [33]. Based on this concept, Fisher et al. created a model-agnostic version of the PFI [29] named *Model Class Reliance* in 2018.

There are three steps in the PFI algorithm proposed by Fisher et al. (2018). Considering the trained model *f*, feature matrix *X*, target vector *y* and error measure $\mathcal{E}$*(y, f)*:

1. Compute the original model's error: $e^{orig} = \mathcal{E}(y, f(X))$
2. For each $i = \{1 \ldots k\}$:
   - Create X^{perm} by permuting feature i in the data X to break the relationship between feature i and true outcome y.
   - New error value $e^{perm} = \mathcal{E}(y, f(X^{perm}))$
   - Calculate the PFI value for feature i^{th}: $S_i = \frac{e^{perm}}{e^{orig}}$
3. Sort and present features in descending order of value S_i

3.3.3. Partial Dependence Plot (PDP)

Partial Dependence Plot (PDP) is one of the most widely adopted visualisation-based XAI methods. It was first introduced by Friedman (2001) to show the relationship between feature values and the black-box model outcome by visualising on the plot [34]. PDP is a type of global model-agnostic approach, which means that it considers all records and demonstrates the global relationship between a specific feature and the prediction.

$$\mathcal{PD}(x_s) = E(x_c)[f(x_s, x_c)] = \int f(x_s, x_c) dP(x_c) \tag{17}$$

where:

- f refers to the ML model
- $x_s \in S$ - set of input features for which PDP function is plotted.
- $x_c \in C$ - set other input features *f*, which means that: $x_c = x \setminus x_s$
- *dP(x_c)* is the marginal distribution of x_c

According to equation (17), PDP function marginalise model f outcome over the distribution of set C in order to shows the relationship between the

features in set S and model f output. Another way to compute the partial function $\mathcal{PD}(x_s)$ by using Monte Carlo method [35]:

$$\mathcal{PD}(x_s) = \frac{1}{k}\sum_{i=1}^{k} f(x_s, x_c^i) \tag{18}$$

where:

- k refers to the total number of data records
- x_c^i are feature values from the set of features C

Equation (18) demonstrates the average marginal effect of given feature values in a set of features S on the model outcome. Additionally, PDP assumes that there is no correlation between features in S and features in C. However, this assumption is likely violated in real life; thus, the function $\mathcal{PD}$ would take invalid data points as arguments and demonstrate misleading explanations. Moreover, PDP can have a hidden heterogeneous effect as it only visualise the average marginal effect. Specifically, each data point can have a positive or negative association with the model's prediction; thus, only presenting the average effects of all points would hide such relationships [35].

3.3.4. Individual Conditional Expectation (ICE)

Individual Conditional Expectation (ICE) is calculated similarly to PDP, but it is designed to inspect individual observations of the model. Compared to one line overall in PDP, ICE curves display one line for each record that demonstrates the record's prediction changes when a feature changes. In other words, PDP represents the average of all lines in an ICE plot. PDP covers the heterogeneous relationships created by interactions between features, therefore, it only works well if these interactions are weak. ICE uncovers such relationships, hence providing more insight than PDP. ICE would take each record in the observations set $\{(x_s^{(i)}, x_c^{(i)})\}_{i=1}^{k}$ and plot the curve $f_s^{(i)}$ against $x_s^{(i)}$. Thus, the relationship of the feature with prediction is shown for k observations, and the average of all curves results in PDP.

4. Proposed Method

This section demonstrates the proposed framework for an explainable AI-based IDS platform. In IDS, identifying malicious behaviour is only the

first step because understanding such a decision is crucial for a solution. An insight into the decision helps administrators identify the part of the network, the part of features and the security policies compromised by attackers [36]. With the information provided by XAI, the IDS operator can give the correct actions, whether it is to debug the IDS model or apply new security policies to prevent the same attacks in the future.

The proposed framework aims to develop an effective explainable AI-based IDS, therefore, various XAI methods are used to generate both local and global explanations. The local approach focuses on explaining single network traffic that was classified as intrusion, thus, it increases the security expert's and user's trust in the IDS. Meanwhile, the global approach gives insights into the complex functioning of the IDS, hence helping the IDS operators to improve the model's efficiency and providing the experts with deep knowledge about cyber attacks.

4.1. Local Explanation

The proposed framework utilises two different methods to generate local explanations, including Shapley Additive Explanations (SHAP) and Individual Conditional Expectation (ICE). Shapley values of individual prediction measure the effect of each input feature on the model's prediction. The explanations are visualised to be more intuitive. Additionally, XAI recipients can easily find how that value of a particular feature affects the predicted probabilities made by the IDS. Meanwhile, ICE inspects each data record and displays one line for each instance demonstrating the prediction's changes according to a specific feature's changes. Therefore, ICE can support Shapley's values very well in showing the impact of a feature on the final prediction.

4.2. Global Explanation

The proposed framework utilises three different methods to generate global explanations, including Partial Dependence Plot (PDP), Permutation Feature Importance (PFI) and SHAP. PDP visualises the global impact of specific features on the predictions of a fitted ML model. This helps security experts to study the importance of each feature in a particular benchmark dataset. PFI computes the change in prediction error of a fitted model over all possible permutation of the feature's values; thus, it shows the global impact of each feature on the model's predictions.

PFI typically shows the twenty most important features from the set of input features, therefore, it would be useful in the process of feature selection.

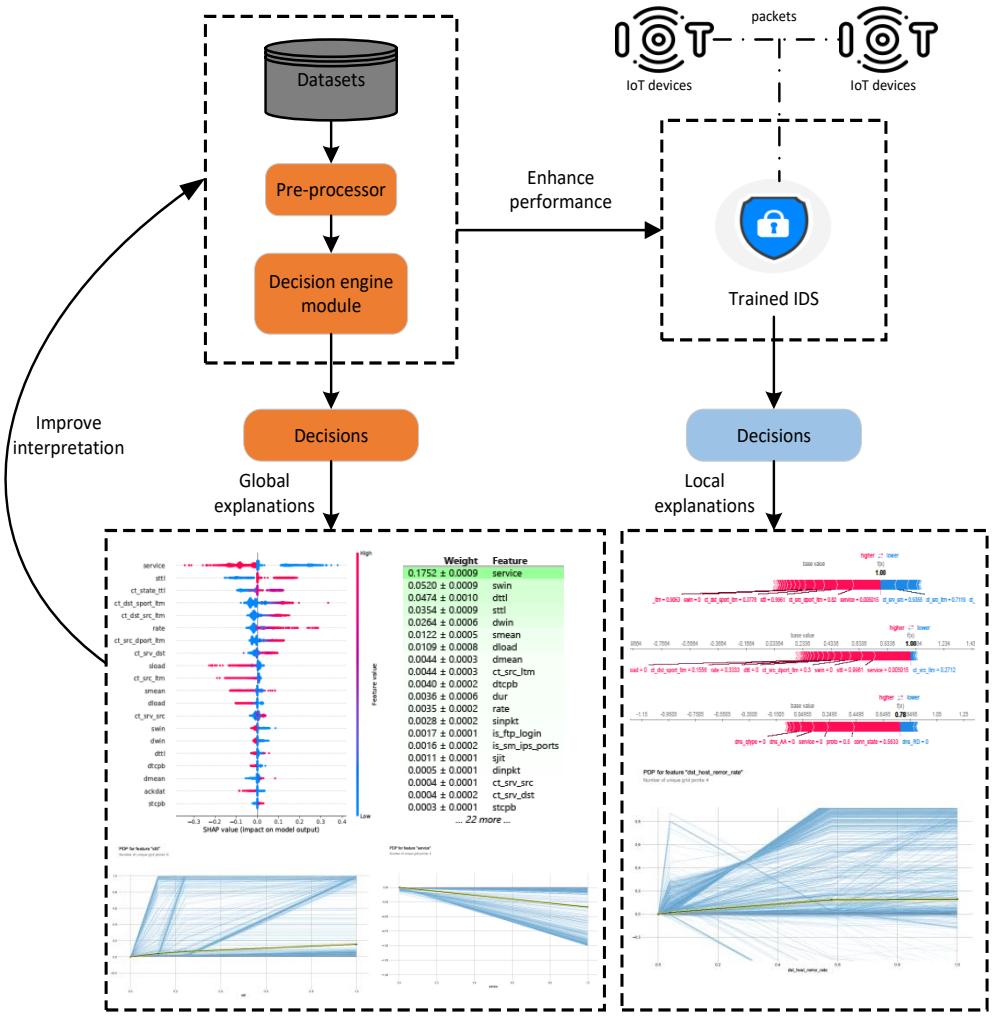


Figure 2: Overview of the structure of SPIP framework

SHAP can also be used to show the most important features by combining Shapley values. Calculating SHAP for each data instance would generate a matrix whose rows represent for instances and columns denote features. Then, the importance score of each feature can be obtained by computing the average of the absolute Shapley values per feature across this matrix:

$$\mathcal{A}_j = \sum_{i=1}^{k} ||\phi_j(x_i)|| \tag{19}$$

where:

- $\mathcal{A}_j$ is the average Shapley value of the j^{th} feature
- k refers to the number of instances in the dataset
- $\phi_j(x_i)$ is the Shapley value of the j^{th} feature in the i^{th} instance

After obtaining these values, the features are typically sorted based on the importance score and the twenty most important features are shown.

5. Experimentation and Results

In this section, the experiment configurations, including the datasets used and the structures of the models, are discussed. We also evaluate the performance of different classifiers and the proposed explainable AI-based IDS framework. As discussed above, our framework generates both global and local explanations using multiple XAI methods. The experiments aim to evaluate the efficiency of generated explanations. Additionally, different classifiers are customised to utilise the results obtained from the proposed framework, hence improving the model's performance and giving more insight into the datasets used as well as the functioning of the AI-based IDS.

5.1. Datasets

Several statistical analyses have revealed the drawbacks of the benchmark dataset KDD-CUP99 and shown that the dataset may adversely affect the performance, evaluation and comparison of different classifiers. NSL-KDD is a purified version of KDD-CUP99 as it has significant improvement. Firstly, redundant and duplicate records are removed to prevent the IDS from being

biased [37]. Secondly, the number of selected records from each difficulty-level group is inversely proportional to the percentage of records in the KDD-CUP99. As a result, the evaluation of IDS is more accurate because the range of classification rates is wider [37]. Lastly, the NSL-KDD dataset contains a sufficient and reasonable number of records to ensure reliable results from different classifiers. Each data instance has a total of 41 features and one label classifying it into attack or normal network flow. The features can be basic, content-related, time-related or host-based traffic features [38]. There are three datasets in NSL-KDD, including (1)KDDTrain+ for training, (2)KDDTest+ for testing, and (3)KDDTest-21 for advanced testing. Researchers designed 21 machine learning models to evaluate the NSL-KDD datasets. Then, they removed all records in KDDTest+ that were correctly labelled by 21 models and make a new dataset from the leftover called KDDTest-21[37].

Moustafa and Slay (2015) created the UNSW-NB15 dataset in a synthetic setting built in the UNSW cyber security lab. UNSWNB-15 is a modern NIDS benchmark dataset as the creators utilize novel methods to generate modern normal and synthetic attack network activities [39]. Unlike NSL-KDD/KDDCup-99 datasets, UNSW-NB15 contains new attack and standard network traffic patterns. IXIA is utilized to generate network traffic then Argus, Bro-IDS tools and twelve algorithms are used to extract a total of 47 features from the generated activities. Depending on the IXIA tool's report, each record has 2 labels including (1) *attack_cat*: name of each attack category and (2) *label*: 0 for normal, 1 for attack records. UNSW-NB15 has multiple attacks and up-to-date information of packets, which are significant advantages over old datasets.

Moustafa [6] developed a new dataset to validate Industrial IoT (IIoT) systems named the ToN_IoT dataset. The dataset contains telemetry data, which are collected from IoT devices, to detect intrusions that manipulate IoT devices [40]. The IoT telemetry data was generated in a testbed environment with three layers *Edge, Fog* and *Cloud* to represent real-life data from current production IoT/IIoT networks. The NSX-VMware platform was utilized to provide the features of Software-defined Network (SDN) and Network Function Virtualization (NFV) to manage the interaction between the three layers [40]. ToN_IoT was generated on a realistic simulation of an IoT environment, incorporating both normal and various types of attacks in IIoT applications.

Table 1: List of symbolic and binary features in the three datasets

Dataset	Symbolic features	Binary features
NSL-KDD	protocol_type, service, flag	land, logged_in, root_shell, su_attempted, is_host_login, is_guest_login, logged_in
UNSW-NB15	proto, state, service	is_sm_ips_ports, is_ftp_login
ToN_IoT	conn_state, dns_query, ssl_version, ssl_cipher, ssl_subject, ssl_issuer, http_method, http_uri, http_version, http_orig_mime_types, http_resp_mime_types, weird_name, weird_addl, type	dns_AA, dns_RD, dns_RA, dns_rejected, ssl_resumed, ssl_established, weird_notice

5.2. Data Pre-processing

Each dataset above has several symbolic and binary input features. Names of such features are listed in table 1. Symbolic features from these datasets are converted using a label encoder and binary features remained unchanged. Other input features in the three datasets are continuous which can be integer or float. In this experiment, we use min-max normalization method so that all continuous features are in the range [0,1] after being scaled:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{20}$$

There are 17 classes of attack in the three datasets. Among those, *Denial of Service (DoS)* attack appears in all three datasets. *Probe* attack in NSL-KDD is equivalent to *Analysis* in UNSW-NB15 and *Scanning* in ToN_IoT. Additionally, both UNSW-NB15 and ToN_IoT have the *Backdoor* attack class. In the NSL-KDD dataset, classes of attack are categorised into different types that are shown in table 2. The *label* column in NSL-KDD contains these attack types' names, therefore, we must convert them into the original attack classes' names before using the NSL-KDD dataset.

5.3. Evaluation Metrics

Accuracy, precision, recall, and F1-score are evaluation indicators to evaluate the model's performance. These statistical measures are calculated from ground truth values including True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). These evaluation indicators explain

Table 2: List of attack types in NSL-KDD

Class of attack	Attack types
probe	ipsweep, nmap, portsweep, satan, saint, mscan
DoS (Denial of Service)	back, land, neptune, pod, smurf, teardrop, apache2, udpstorm, processtable, mailbomb
u2r (user to root)	buffer_overflow, loadmodule, perl, rootkit, xterm, ps, sqlattack
r2l (root to local)	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster, snmpgetattack, named, xlock, xsnoop, sendmail, httptunnel, worm, snmpguess

the performance of the model; thus, they provide feedback to make improvements and design an IDS in IoT networks with desirable performance. The definition of ground truth values are listed as follow:

- **True Positive (TP)**: number of instances correctly classified as the target outcome
- **True Negative (TN)**: number of instances correctly classified as others
- **False Positive (FP)**: number of instances wrongly classified as the target outcome
- **False Negative (FN)**: number of instances wrongly classified as others

Based on above values, evaluation indicators are calculated as follow:

1. **Accuracy**: ratio of the number of correctly classified instances to the total number of instances in the test set.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{19}$$

2. **Precision**: ratio of the number of instances correctly classified as the target outcome to the number of instances classified as the target outcome

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{20}$$

3. **Recall**: ratio of the number of instances correctly classified as the target outcome to the total number of target records.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{21}$$

4. **F1-score**: this value measures precision and recall at the same time by using the harmonic mean in place of the arithmetic mean

$$F1 \text{ -score } = \frac{2 * \text{ Recall } * \text{ Precision}}{\text{Recall } + \text{ Precision}} \tag{22}$$

5.4. Computer Environment

The models were developed using the Python programming language on Microsoft Windows 10 Home Version 20H2 . These were evaluated on a system with with 16 GB RAM. In this experiment, we use TensorFlow - a major deep learning framework to train and evaluate the model. TensorFlow is GPU-accelerated and this framework utilised the NVIDIA GeForce GTX 1060 3GB on the evaluation system to effectively train the DL-based models.

5.5. Intrusion Detection System

The first stage of this work is to build and train different types of classifiers. Firstly, binary classifiers are built to detect attacks without identifying the class of such attacks. From there, in this work we apply global XAI methods (SHAP, PDP) to study the most important features in each dataset and select the most relevant features to build a more efficient model. Secondly, we build one-vs-all classifiers to detect each class of attack separately. After obtaining results from fitted one-vs-all classifiers, the proposed framework will generate both local and global explanations, hence giving insights into the importance of each feature and the functioning of models. Lastly, we build target-vs-normal classifiers that distinguish between a targeted attack class and the normal traffic. This will help domain experts in analysing the distinctive characteristics of each attack class or attack type.

We establish the models based on fully connected networks with ReLU activation. The input dimension of each classifier depends on the dataset used as it equals the number of input features. Each classifier has 3 hidden layers, and each hidden layer contains 50 neurons. The output dimension of each classifier is 2. The TensorFlow framework is used to effectively build the intrusion detection models. We use Adam optimizer, learning rate of 0.001 and dropout rate of 0.1. The batch size is 32 with a total of 10 epochs.

Table 3: Results about Binary Classifiers

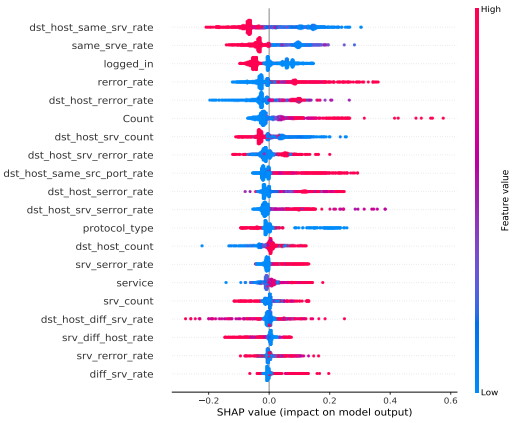
Dataset	Accuracy	Precision	Recall	F1-score
NSL-KDD	0.811	0.921	0.732	0.815
UNSW-NB15	0.866	0.811	0.988	0.891
ToN_IoT	0.873	0.784	0.880	0.829

5.6. Binary Classifiers

We construct and train binary classifiers on the three datasets using the complete set of input features. Table 3 shows the evaluation indicators obtained from these classifiers. These values indicated that the performance of the models is sufficient to apply the proposed XAI framework and study the relevant features that models have learned to identify malicious activities. From the fitted model, the framework calculates the importance score of each feature using SHAP and PFI. The most important features of attacks that the model has learned from the three datasets are shown in Figure 3.

As shown in Figure 3, lists of the top 20 important features that are generated by SHAP and PFI are quite different in all datasets. The outputs from PFI are easier to understand since each feature has a simple numeric value that indicates its overall impact on the model's decision. Therefore, the comparisons between features are relatively simple, which makes PFI's lists suitable for lay-users to understnad. However, explanations generated by PFI are so simple that they only calculate the medium permutation importance without showing how much each features matters. This would lead to drawbacks in the case of features that have a large impact on a few predictions but no impact in general or features that have an average impact on all predictions.

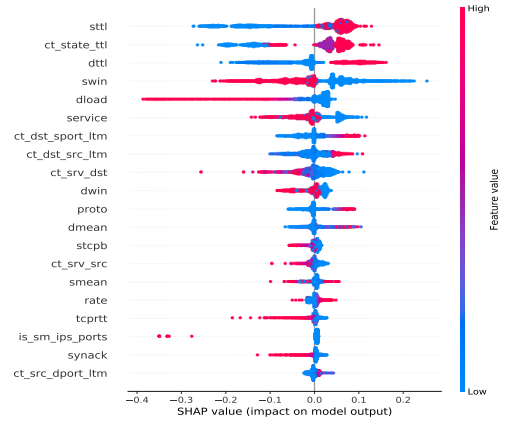
SHAP presents the list of the most important features by plotting positions on the graphs shown in Figures 3a, 3c and 3e. To understand the plot, we need to analyse each dot since each dot represents an instance in the dataset. The dot's vertical position indicates the feature it is depicting, its horizontal position refers to the impact of that value on the model's prediction. The colors of the dot represent the value of that feature for that instance of the dataset, indicating whether it is high, medium or low (red, purple or blue, respectively). For example, in Figure 3a, a high value of the feature *'Count'* increased the probability that the activity is attack by 20% to 60%. SHAP gives a different view of feature importance and shows how each features matters. Therefore, it mitigates the major disadvantage



(a) NSL-KDD features extracted by SHAP

Weight	Feature
0.0587 ± 0.0023	dst_host_srv_count
0.0420 ± 0.0013	protocol_type
0.0217 ± 0.0011	logged_in
0.0203 ± 0.0007	dst_host_serror_rate
0.0140 ± 0.0007	hot
0.0101 ± 0.0012	service
0.0097 ± 0.0010	srv_serror_rate
0.0090 ± 0.0010	dst_host_same_src_port_rate
0.0084 ± 0.0014	dst_host_srv_rerror_rate
0.0078 ± 0.0004	dst_host_srv_serror_rate
0.0067 ± 0.0014	dst_host_rerror_rate
0.0064 ± 0.0009	dst_host_count
0.0052 ± 0.0025	dst_host_same_srv_rate
0.0049 ± 0.0008	same_srve_rate
0.0041 ± 0.0019	Count
0.0026 ± 0.0009	serror_rate
0.0008 ± 0.0002	root_shell
0.0007 ± 0.0004	is_guest_login
0.0007 ± 0.0017	rerror_rate
0.0002 ± 0.0001	num_shells
... 21 more ...	

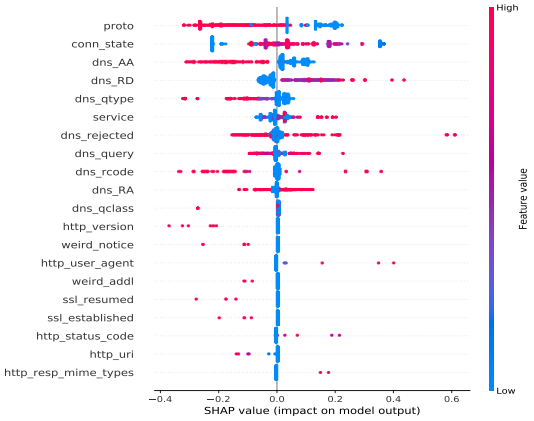
(b) NSL-KDD features extracted by PFI



(c) UNSW features extracted by SHAP

Weight	Feature
0.0660 ± 0.0012	swin
0.0449 ± 0.0014	dttl
0.0432 ± 0.0018	sttl
0.0395 ± 0.0010	ct_state_ttl
0.0175 ± 0.0014	dload
0.0121 ± 0.0009	service
0.0079 ± 0.0003	smean
0.0071 ± 0.0014	ct_srv_dst
0.0051 ± 0.0007	ct_dst_src_ltm
0.0044 ± 0.0003	ct_dst_ltm
0.0035 ± 0.0003	is_sm_ips_ports
0.0033 ± 0.0006	ct_dst_sport_ltm
0.0026 ± 0.0003	ct_src_dport_ltm
0.0024 ± 0.0002	synack
0.0022 ± 0.0004	rate
0.0022 ± 0.0003	proto
0.0021 ± 0.0003	dwin
0.0008 ± 0.0001	tcprtt
0.0002 ± 0.0001	spkts
0.0002 ± 0.0001	sbytes
... 22 more ...	

(d) UNSW features extracted by PFI



(e) ToN_IoT features extracted by SHAP

Weight	Feature
0.1956 ± 0.0026	proto
0.1792 ± 0.0027	conn_state
0.0824 ± 0.0010	dns_AA
0.0737 ± 0.0015	service
0.0588 ± 0.0017	dns_RD
0.0434 ± 0.0009	dns_query
0.0225 ± 0.0009	dns_rejected
0.0163 ± 0.0009	dns_qtype
0.0104 ± 0.0008	dns_rcode
0.0082 ± 0.0005	dns_RA
0.0010 ± 0.0002	dns_qclass
0.0007 ± 0.0001	weird_notice
0.0004 ± 0.0002	ssl_resumed
0.0003 ± 0.0001	ssl_established
0.0003 ± 0.0001	weird_addl
0.0003 ± 0.0001	http_version
0.0002 ± 0.0001	weird_name
0.0002 ± 0.0001	ssl_version
0.0001 ± 0.0000	http_uri
0.0001 ± 0.0000	src_bytes
... 18 more ...	

(f) ToN_IoT features extracted by PFI

Figure 3: Top 20 relevant features of attacks that binary classifiers learned from the three datasets (extracted by SHAP and PFI)

Table 4: Results about Binary Classifiers

Dataset	Accuracy	Precision	Recall	F1-score
NSL-KDD with SHAP features	0.787	0.970	0.646	0.776
NSL-KDD with PFI features	0.817	0.963	0.706	0.815
UNSW-NB15 with SHAP features	0.864	0.808	0.987	0.889
UNSW-NB15 with PFI features	0.859	0.801	0.990	0.885
ToN_IoT with SHAP features	0.872	0.790	0.866	0.826
ToN_IoT with PFI features	0.873	0.787	0.876	0.829

Table 5: Results of Binary Classifiers with Combined Set of Features

Dataset	Accuracy	Precision	Recall	F1-score
NSL-KDD with combined features	0.787	0.971	0.645	0.775
UNSW-NB15 with combined features	0.858	0.797	0.994	0.885
ToN_IoT with combined features	0.873	0.785	0.879	0.829

of PFI's explanations that we discussed above.

Evaluation indicators in Table 4 show that when the classifiers are trained on subsets of input features, their performance only changes slightly and even gets better in a few cases. Models that are trained on the PFI subset of UNSW-NB15 and NSL-KDD datasets get higher recall and accuracy, respectively. Models that are trained on the SHAP subset of ToN_IoT and NSL-KDD datasets get higher precision. This means that the sets of input features collected in these three datasets are not yet optimal to detect attack activities. Therefore, we combine the most important features extracted by SHAP and PFI (Figure 3) and use them to re-train the binary classifier to detect attacks without altering the model's parameters. Results are shown in Table 5. Models that are trained on the combined subset of UNSW-NB15 and NSL-KDD datasets get higher recall and precision, respectively.

We also measure the time taken when using a different set of features for training and evaluating the models, as shown in Table 6. In all three datasets, using the set of features extracted by SHAP reduces both training time and detection time without significantly lower the model's performance. The training time tends to be reduced much more if its value is already high (when using the original features). Therefore, using SHAP features would reduce the network overhead and computational resources without lowering the effectiveness of IDS in IoT networks.

Table 6: Training time and detection time taken (using different set of features)

	Original	SHAP	PFI	Combined
NSL-KDD training time (ms)	88801	87006	87520	87815
Total detection time (ms)	507	434	448	470
UNSW-NB15 training time (ms)	128519	123377	122447	121950
Total detection time (ms)	1444	1268	1336	1292
ToN_IoT training time (ms)	257544	250684	252824	259633
Total detection time (ms)	1664	1458	1508	1493

Table 7: Results about one-vs-all classifiers

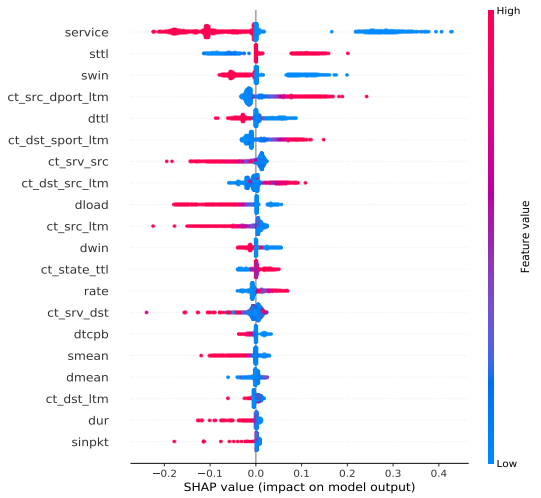
Attack class	Dataset	Accuracy	Precision	Recall	F1-score
DoS	NSL-KDD	0.931	0.958	0.829	0.889
Probe	NSL-KDD	0.946	0.819	0.636	0.716
Exploits	UNSW-NB15	0.920	0.763	0.596	0.669
Generic	UNSW-NB15	0.991	0.999	0.962	0.980
Reconnaissance	UNSW-NB15	0.979	0.812	0.664	0.731
DDoS	ToN_IoT	0.987	0.822	0.902	0.859
XSS	ToN_IoT	0.969	0.629	0.736	0.678

5.7. One-vs-All Classifiers

The next stage is to build one-vs-all classifiers to detect a specific class of attack and apply both global and local XAI methods. We expect that the explanations from the proposed framework would provide insight into the functioning of the models and help security experts in studying the characteristics of each cyber attack class. After building classifiers to detect individual attack classes, it is observed that a few attack classes can be well detected by the model, whilst the performance of classifiers in detecting other classes is insufficient for applying the proposed framework. Results from training one-vs-all classifiers are shown in Table 7.

Among all classes of cyber attack in Table 7, the models can obtain the best performance on *Generic*. Therefore, we apply the XAI methods in analysing the model's decision in detecting *Generic*. Based on the explanations generated by both SHAP and PFI in Figure 4, the following features are considered as the most important features to detect a *Generic* attack:

- *service*: Low values of *service* feature would not affect the model's decision, but in many other cases, they increases the probability that



(a) UNSW-NB15 features extracted by SHAP

Weight	Feature
0.1752 ± 0.0009	service
0.0520 ± 0.0009	swin
0.0474 ± 0.0010	dttl
0.0354 ± 0.0009	sttl
0.0264 ± 0.0006	dwin
0.0122 ± 0.0005	smean
0.0109 ± 0.0008	dload
0.0044 ± 0.0003	dmean
0.0044 ± 0.0003	ct_src_ltm
0.0040 ± 0.0002	dtcpb
0.0036 ± 0.0006	dur
0.0035 ± 0.0002	rate
0.0028 ± 0.0002	sinpkt
0.0017 ± 0.0001	is_ftp_login
0.0016 ± 0.0002	is_sm_ips_ports
0.0011 ± 0.0001	sjit
0.0005 ± 0.0001	dinpkt
0.0004 ± 0.0001	ct_srv_src
0.0004 ± 0.0002	ct_srv_dst
0.0003 ± 0.0001	stcpb
... 22 more ...	

(b) UNSW-NB15 features extracted by PFI

Figure 4: Top 20 important features of *Generic* attack class that models have learned from UNSW-NB15

the model identify the records as *Generic* attack by 25% - 40%. List of services in ascending order of value: *http, ftp, smtp, pop3, dns, snmp, ssl, dhcp, irc, radius, ssh, none.*

- *swin*: It refers to the values of source TCP window advertisement. Low values of *swin* feature would not affect the model's decision, but in many other cases, they increase the probability of a prediction as *Generic* attack by 10% - 20%.

- *dttl*: This feature measures the time to live of destination to source connections. Low values of *dttl* feature would slightly increase the chance that the record are detected as *Generic* attack.

- *sttl*: This feature measures the time to live of source to destination connections. In contrast to *dttl*, high values of *sttk* feature would increase the chance that the record is the *Generic* attack by 10% - 20%.

- *ct_src_dport_ltm*: This feature refers to the number of connections of the same source address and the destination port in 100 connections according to the last time. Interestingly, SHAP differs from PFI in computing the effect of this feature on the model's decisions. In this

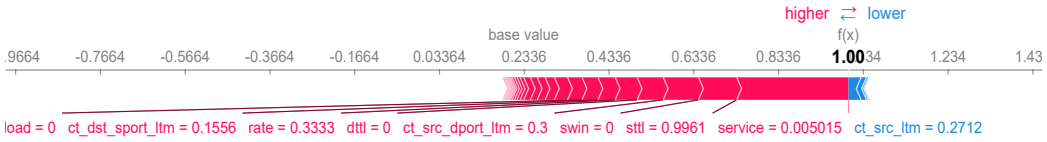
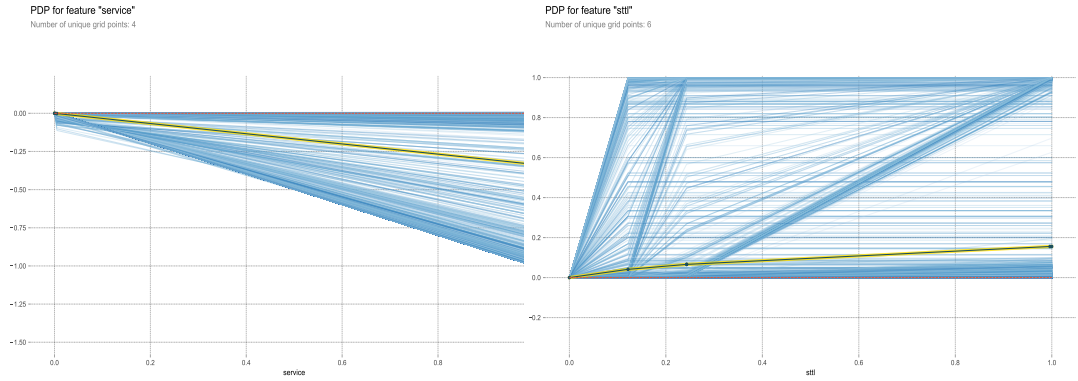


Figure 5: A local explanation on the *Generic* attack class



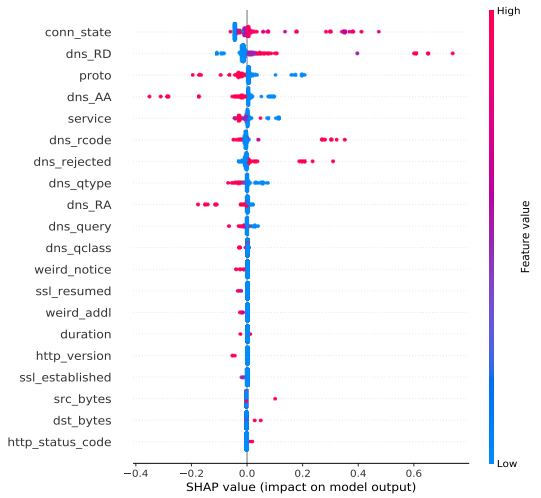
(a) feature *service* (b) feature *sttl*

Figure 6: PDP and ICE for features *service* and *sttl* feature in detecting *Generic* attack

case, PFI hinders the feature's impact because PFI does not demonstrate how much this feature matters. PFI only computes the medium permutation importance and neglects data records in which this feature has a great impact.

SHAP can also be used to generate individual explanations. Figure 5 shows a data record that is classified as *Generic* attack. The model's decision is decomposed into the sum of effects of each feature value. In this specific data instance, *service* = 0.005015 (dns), *sttl* = 0.9961 (254) and *ct_src_dport_ltm* = 0.3 (16) are the three features that contribute the most to the final decision of the model.

To further analyse the effects of these features on the overall decision or individual decisions of the model, we apply PDP and ICE as shown in Figure 6a and 6a. ICE lines are blue and the average of all blue lines are yellow lines, which is also known as PDP. These methods demonstrate the model's prediction changes when a feature changes. Figure 6a tell us that the increase in value of *service* feature would lead to decrease in the probability that the model classify an activity as *Generic* attack. Moreover, it can be divided into



(a) ToN_IoT features extracted by SHAP

Weight	Feature
0.0498 ± 0.0010	dns_RD
0.0449 ± 0.0012	conn_state
0.0190 ± 0.0002	dns_query
0.0135 ± 0.0007	dns_rejected
0.0126 ± 0.0006	service
0.0102 ± 0.0002	dns_rcode
0.0086 ± 0.0010	proto
0.0049 ± 0.0004	dns_AA
0.0047 ± 0.0006	dns_RA
0.0043 ± 0.0004	dns_qtype
0.0003 ± 0.0000	http_version
0.0002 ± 0.0001	ssl_resumed
0.0002 ± 0.0001	dns_qclass
0.0001 ± 0.0001	weird_name
0.0001 ± 0.0001	weird_notice
0.0001 ± 0.0000	duration
0.0001 ± 0.0000	ssl_established
0.0001 ± 0.0000	src_bytes
0.0000 ± 0.0000	weird_addl
0.0000 ± 0.0000	http_method

... 18 more ...

(b) ToN_IoT features extracted by PFI

Figure 7: Top 20 important features of *DDoS* attack class that models have learned from ToN_IoT

two groups of instances depending on how much the value of *service* feature affects the model's prediction. One group slightly decreases the chance of a *Generic* attack by only 10%, whilst the other group significantly decreases such chance by 90% and up to 100%.

Figure 6b is more complex as the data instances can be divided into 4 groups. Data records in the first and second groups would be classified as *Generic* attack if the value of *sttl* feature is around 0.1 and 0.25, respectively. Whilst data records in the third group are classified as an attack if the value of *sttl* feature reaches its maximum, the effect of *sttl* feature on data instances' decisions in the fourth group is very small.

The one-vs-all classifier that detects *DDoS* attack in ToN_IoT dataset also perform well, which is shown in Table 7. This good performance allows us to further analyse the results by using XAI methods as we had used above to study *Generic* attack. Figure 10 show the list of the 20 most important features extracted by SHAP and PFI from the ToN_IoT dataset. Both SHAP and PFI extract some notable features, including:

- *conn_state*: this feature is a string that refers to the state of connections, such as S0 (connection without replay), S1 (connection established), and REJ (connection attempt rejected). Higher values of this

Figure 8: A local explanation on the *DDoS* attack class

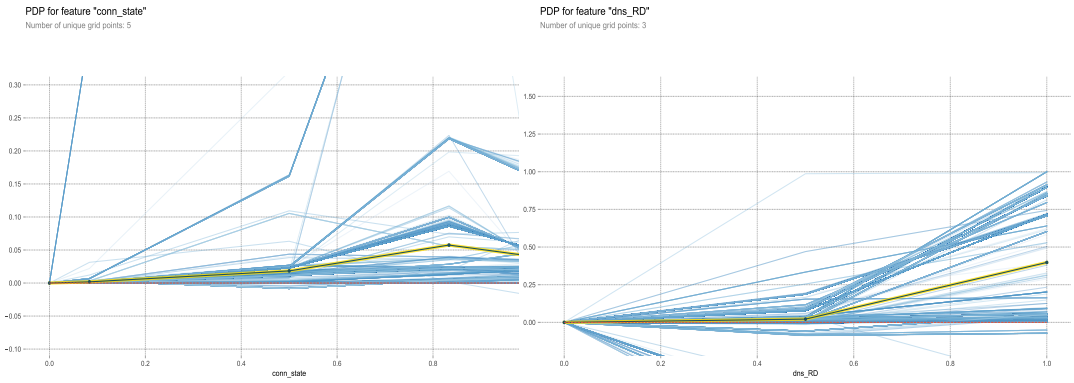
feature would lead to higher probability that the data instances are classified as *DDoS* attacks. Symbols of *conn_state* feature in ascending order of value: *OTH, REJ, RSTO, RSTOS0, RSTR, RSTRH, S0, S1, S2, S3, SF, SH, SHR*

- *dns_RD*: this feature has boolean data type that indicates the recursion desire of DNS. After applying label encoding and scaler, the data preprocessor assigns value of "-" as 0, value of *False* as 1 and value of *True* as 2. Same as *conn_state* feature, higher value of *dns_RD* increase the chance that an instance is classifed as *DDoS* attack.

Figure 8 demonstrates an individual explanation for a single data instance that is correctly classified as *DDoS* attack by the model. The value "0.5833" of feature *conn_state* contributes greatly to the final decision, which means that if *conn_state* equals to S1 the activity is likely classified as *DDoS* attack. Moreover, *proto* = 0.5 (TCP) and *service* = 0 (none) also contribute to such decision. Although *dns_RD* = 0 (none) decreases the chance of a *DDoS* attack, the model has computed that overall probability equals to 0.78. Therefore, this is a *DDoS* attack.

Figure 9a and 9b shows PDP and ICE for the two features *conn_state* and *dns_RD*. Overall, high values of both of these features would increase the chance of a *DDoS* attack. Specifically, the feature *dns_RD* significantly increase such probability if it reaches maximum value. However, compared to Figure 6a and 6b, there are many data instances in Figure 9a and 9b that do not follow the rules mentioned above, which signifies hidden relationships between input features. This requires us to further analyse the results and demonstrate any existing correlations between features that can affect the final decision of the model.

Figure 10 demonstrates SHAP dependence plot for the three most important features consisting of *conn_state, dns_RD* and *proto*. Each dot on these plots represents a data instance in the dataset. The horizontal position indicates the value of the feature on the x-axis and the colour represents the



(a) feature *conn_state* (b) feature *dns_RD*

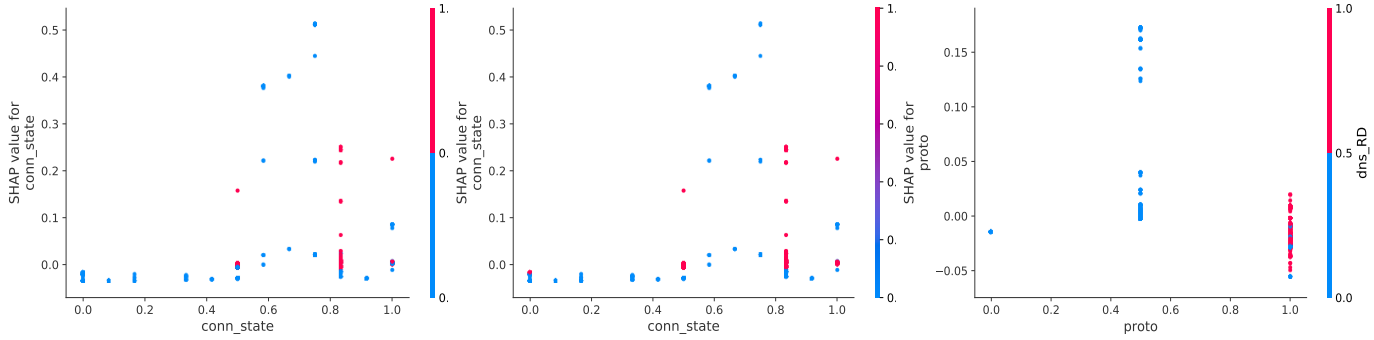
Figure 9: PDP and ICE for features *conn_state* and *dns_RD* feature in detecting *DDoS* attack

value of other features, which are shown on the right column. Whilst, the vertical position refers to the SHAP value that indicates how much these values of the two features affect the model's decision. Before analysing these plots, there are three different types of the protocol used in ToN_IoT, including ICMP (labeled as "0"), TCP (labeled as "0.5") and UDP (labeled as "1").

Figure 10a and 10b are quite similar in terms of distributions. Both plots observe high value of SHAP when the value of *conn_state* is in range of [0.5, 1]. Specifically, SHAP value is high when the value of *conn_state* is in range of [0.6, 0.8] and the values of *dns_RD* or *proto* are low. Whilst, high value of *dns_RD* or *proto* would significantly increase SHAP value when *conn_state*'s value is smaller than 0.6 or higher than 0.8. Figure 10c shows that all detected DDoS attacks use the UDP protocol.

5.8. Relationship Between Feature Values and Attacks

Applying both local and global XAI methods on one-vs-all classifiers provides us with interesting results that reflect insight into the datasets and characteristics of each attack class. The relationships between the input features and individual class of attack are demonstrated through visualisation using our proposed framework. These explanations are intuitive and would be easy to understand for different types of XAI recipients with various levels of knowledge about cyber security or artificial intelligence. Therefore, the



(a) conn_state and dns_RD (b) conn_state and proto (c) proto and dns_RD

Figure 10: Dependence contribution plots of features *proto*, *dns_RD* and *conn_state*

SPIP framework can effectively enhance the benefits that XAI models could bring to audiences.

The experiments in this paper have shown a strong relationship between input features and attack classes. By applying the proposed framework on well-performing one-vs-all classifiers, we can analyse such relationships and the decision process of the AI-based models. For example, the framework studies that features *conn_state, dns_RD* and *proto* have strong relationships with *DDoS* attacks, which matches the real characteristics of *DDoS* attacks. This indicates that the classifier has given valid predictions as it has studied the right patterns of the particular attack class. Moreover, this has proven that the explanations generated by the SPIP framework can be used to effectively enhance the interpretability and explainability of AI-based intrusion detection models.

5.9. Discussion

There are several distinct and significant benefits to applying the proposed framework. These are:

- The model assists developers and researchers in evaluating the quality, analysing and detecting any bias in the dataset. This is done by combining the foundation knowledge about individual attack classes and the explanations generated from the SPIP framework.

- SPIP's reasonable explanations support the interpretability of the IDS. It provides the recipient with the demonstration of relationships between features' value and the prediction as well as relationships between input features. Moreover, the framework is post-hoc, which can be applied to any model regardless of the underlying algorithms.

- SPIP evaluates the quality of the collected input features in detecting different attack classes. Each attack class has specific characteristics that are shown in the features' value. A quality set of input features would demonstrate these characteristics, hence enhancing trust in the models' decisions. Therefore, outputs from SPIP can help researchers in evaluating the input features and discover novel features to collect from network activities.

- Zero-day attacks exploit unknown vulnerabilities in the system; therefore, it is difficult to detect and analyse them. AI-based IDS can detect

zero-day attacks and the SPIP framework can explain such decisions. SPIP's outputs would enhance the investigation process by helping security experts in discovering these unknown attacks' specific characteristics.

That said, there are also limitations to employing such a framework. The drawbacks of the SPIP framework:

- As outlined, strong IDS performance requires a comprehensive training set with valuable collected features from network activities. This is a significant challenge in building an effective IDS, as it is impossible to construct a dataset that involves all normal and malicious behaviours in a heterogeneous environment of IoT networks. Moreover, many existing datasets collect an incomplete set of features, and network information is captured without including both headers and payloads [41]. However, the SPIP framework can support security experts in collecting more relevant features from the network traffic, hence enhancing classifiers' performance.

- The generated explanations are limited to presenting notable features and their correlations without determining the security vulnerabilities that the attack exploits. However, security experts would utilise those results to discover and demonstrate the existing vulnerabilities based on their knowledge about the victim system and the notified attacks.

6. Conclusion & Future Work

This work has proposed the SPIP framework, developed for the purposes of evaluating explainable DL algorithms for IDS deployments in IoT environments. The SPIP framework generates both local and global explanations, and combines various XAI methods, includingLong Short-Term Memory (LSTM), Shapley Additive Explanations (SHAP), Permutation Feature Importance (PFI), Individual Conditional Expectation (ICE) and Partial Dependence Plot (PDP). The generated explanations by our framework are intuitive and easy for audiences of different technical levels to understand. The SPIP framework extracts a customized set of input features that outperforms the original set in the three datasets, including NSL-KDD, UNSW-NB15 and ToN_IoT. More importantly, the relationships extracted by SPIP

matches the characteristics of specific attack classes. The framework proposed in this research would greatly enhance the utilisation of AI-based IDS in the cyber defence system.

SPIP has some limitations which would be improved in future research. This framework's outputs greatly rely on the performance of intrusion detection models, which means that the outputs are not accurate when applying on an IDS with poor performance. Moreover, the proposed framework cannot identify the existing vulnerability that an attack class is exploiting. Future research can focus on building a framework that can suggest potential vulnerabilities by studying the features of network traffic generated during cyber-attacks.

The field of explainable AI has significant uses in cyber defence, and the development of IDS for IoT platforms is a necessary area that would significantly benefit from this work. The SPIP framework seeks to begin to address this issue.

References

[1] Government regulations in cyber security: Framework, standards and recommendations, Future Generation Computer Systems 92 (2019) 178–188.

[2] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, A. Wahab, A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions, Electronics 9 (7) (2020).

[3] I. H. Sarker, M. H. Furhad, R. Nowrozy, Ai-driven cybersecurity: an overview, security intelligence modeling and research directions, SN Computer Science 2 (3) (2021) 1–18.

[4] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee, H. Karimipour, Cyber intrusion detection by combined feature selection algorithm, Journal of information security and applications 44 (2019) 80–88.

[5] N. Moustafa, G. Misra, J. Slay, Generalized outlier gaussian mixture technique based on automated association features for simulating and detecting web application attacks, IEEE Transactions on Sustainable Computing (2018).

[6] N. Moustafa, A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets, Sustainable Cities and Society 72 (2021) 102994.

[7] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, X. Bellekens, Utilising deep learning techniques for effective zero-day attack detection, Electronics 9 (10) (2020) 1684.

[8] C. Wu, A. Qian, X. Dong, Y. Zhang, Feature-oriented design of visual analytics system for interpretable deep learning based intrusion detection, in: 2020 International Symposium on Theoretical Aspects of Software Engineering (TASE), IEEE, 2020, pp. 73–80.

[9] D. E. Denning, An intrusion-detection model, IEEE Transactions on software engineering (2) (1987) 222–232.

[10] A. Drewek-Ossowicka, M. Pietrołaj, J. Rumiński, A survey of neural networks usage for intrusion detection systems, Journal of Ambient Intelligence and Humanized Computing 12 (1) (2021) 497–514.

[11] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, M. K. Khan, Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review, in: Procedia Computer Science, Vol. 171, Elsevier B.V., 2020, pp. 1251–1260.

[12] J. D. Moore, W. R. Swartout, Explanation in expert systemss: A survey, Tech. rep., UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST (1988).

[13] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, A. Razaque, Deep recurrent neural network for iot intrusion detection system, Simulation Modelling Practice and Theory 101 (2020) 102031.

[14] V. K. Rahul, R. Vinayakumar, K. Soman, P. Poornachandran, Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security, in: 2018 9th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2018, Institute of Electrical and Electronics Engineers Inc., 2018.

[15] B. Zhou, Y. Sun, D. Bau, A. Torralba, Interpretable basis decomposition for visual explanation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 119–134.

[16] S. Anjomshoae, K. Främling, A. Najjar, Explanations of black-box model predictions by contextual importance and utility, in: International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems, Springer, 2019, pp. 95–109.

[17] N. Seedat, V. Aharonson, Y. Hamzany, Automated and interpretable m-health discrimination of vocal cord pathology enabled by machine learning, in: 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), IEEE, 2020, pp. 1–6.

[18] T. R. Wood, C. Kelly, M. Roberts, B. Walsh, An interpretable machine learning model of biological age, F1000Research 8 (17) (2019) 17.

[19] A. Andalib, V. T. Vakili, An autonomous intrusion detection system using an ensemble of advanced learners, in: 2020 28th Iranian Conference on Electrical Engineering (ICEE), 2020, pp. 1–5.

[20] S. R. Islam, W. Eberle, S. K. Ghafoor, A. Siraj, M. Rogers, Domain Knowledge Aided Explainable Artificial Intelligence for Intrusion Detection and Response, arXiv (11 2019).

[21] D. L. Marino, C. S. Wickramasinghe, M. Manic, An adversarial approach for explainable ai in intrusion detection systems, in: IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, 2018, pp. 3237–3243.

[22] H. Li, F. Wei, H. Hu, Enabling dynamic network access control with anomaly-based ids and sdn, Association for Computing Machinery, New York, NY, USA, 2019.

[23] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, X. Xing, Lemna: Explaining deep learning based security applications, Association for Computing Machinery, New York, NY, USA, 2018.

[24] P. Werbos, Backpropagation through time: what it does and how to do it, Proceedings of the IEEE 78 (10) (1990) 1550–1560.

[25] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks 5 (2) (1994) 157–166.

[26] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[27] S. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, arXiv preprint arXiv:1705.07874 (2017).

[28] D. Apley, Visualizing the effects of predictor variables in black box supervised learning models. arxiv, arXiv preprint arXiv:1612.08468 (2016).

[29] A. Fisher, C. Rudin, F. Dominici, All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously., Journal of Machine Learning Research 20 (177) (2019) 1–81.

[30] Q. Zhao, T. Hastie, Causal interpretations of black-box models, Journal of Business & Economic Statistics 39 (1) (2021) 272–281.

[31] L. S. Shapley, A value for n-person games, Contributions to the Theory of Games 2 (28) (1953) 307–317.

[32] M. T. Ribeiro, S. Singh, C. Guestrin, " why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.

[33] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[34] J. H. Friedman, Greedy function approximation: a gradient boosting machine, Annals of statistics (2001) 1189–1232.

[35] C. Molnar, Interpretable Machine Learning, 2019, `https://christophm.github.io/interpretable-ml-book/`.

[36] B. Mahbooba, M. Timilsina, R. Sahal, M. Serrano, Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model, Complexity 2021 (2021).

[37] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6.

[38] L. Dhanabal, S. Shantharajah, A study on nsl-kdd dataset for intrusion detection system based on classification algorithms, International Journal of Advanced Research in Computer and Communication Engineering 4 (6) (2015) 446–452.

[39] N. Moustafa, J. Slay, Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1–6.

[40] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, A. Anwar, Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems, IEEE Access 8 (2020) 165130–165150.

[41] N. Moustafa, J. Hu, J. Slay, A holistic review of Network Anomaly Detection Systems: A comprehensive survey, Journal of Network and Computer Applications 128 (2019) 33–55.