Towards Anomaly Detection using Explainable AI

Manh-Dung Nguyen[0000−0001−8760−3258], Vinh-Hoa La[0000−0003−1554−4847], Wissam Mallouli[0000−0003−2548−6628], Ana Rosa Cavalli[0000−0003−2586−9071], and Edgardo Montes de Oca[0000−0001−6771−0689]

Montimage EURL, France
firstname.lastname@montimage.com
https://www.montimage.com/

Abstract. Anomaly detection in networks is an important aspect of network security, enabling organizations to identify and respond to unusual patterns of activity that may indicate a security threat or performance issue. By identifying and addressing anomalies in real time, organizations can reduce the risk of data breaches and other security incidents, as well as ensure the optimal performance and reliability of their network infrastructure. However, implementing effective anomaly detection in networks with good quality is a significant challenge, requiring careful consideration of several key factors.
One of the main challenges of anomaly detection in networks is the sheer volume of data that must be processed and analyzed. Networks generate vast amounts of traffic data, making it difficult to identify patterns and anomalies in real time. To address this challenge, anomaly detection systems must be able to handle large amounts of data and operate at high speeds, while also minimizing false positives and false negatives.
In this chapter, we present MMT a monitoring framework developed by the Montimage research team to perform anomaly detection. This framework is being extended with Explainable AI (XAI) capabilities to better understand the classification done by AI/ML-based algorithms. First experimentations are presented in this book chapter using SHAP, LIME and SHAPASH technologies.

Keywords: Network Monitoring · Anomaly detection · Explainable AI.

1 Introduction

Anomaly Detection (AD) techniques can be used to identify a wide range of network anomalies, including network intrusions, malware infections, Denial-of-Service (DoS) attacks, and other forms of malicious activities [4], [9]. There are several different approaches to anomaly detection in networks, including rule-based methods, statistical methods, and machine learning methods. *Rule-based methods* involve defining a set of rules that describe normal network activity, and then flagging any activity that deviates from these rules as anomalous. *Statistical methods* involve using probability distributions and statistical models to identify deviations from normal network activity. *Machine learning* methods involve training algorithms on large datasets of network activity to identify patterns and anomalies.

One of the key challenges of anomaly detection in networks is minimizing false positives, that occur when normal network activities are incorrectly classified as anomalous, and false negatives, obtained when anomalous activities are not detected [1]. To address this challenge, many anomaly detection systems use a combination of multiple techniques, as well as feedback loops and manual review by security analysts.

Other challenges are the need of network activities continuous monitoring and the quality of detecting anomalies. In fact, anomaly detection systems have not only to be able to identify anomalies, but also to accurately detect true ones and quickly respond to them, in order to minimize the risk of security incidents or performance issues. The first challenge can be faced through the combination of automated detection methods and human oversight, as well as ongoing analysis and refinement of the detection algorithms. Moreover, AD systems must avoid misclassifications, *e.g.*, normal network activities detected as anomalous. This requires a deep understanding of network behavior and the ability to adapt to changing patterns of activity over time [2].

One approach to anomaly detection is to use *eXplainable Artificial Intelligence* (XAI) techniques [3], which are designed to provide transparency and interpretability into the decisions made by the algorithm. This can help network administrators understand why certain network activity is classified as anomalous, and can provide insights into potential security threats or performance issues. For example, an XAI-based anomaly detection system might identify a sudden surge in network traffic from a particular IP address as anomalous. By providing explanations of how the algorithm came to that decision, the system can help the network administrator understand that the IP address is engaged in potentially malicious activity, such as a distributed denial-of-service (DDoS) attack.

In this chapter, we rely on *Montimage Monitoring Tool* (MMT) [18], *i.e.*, a set of modules to perform real-time or post analysis of captured traffic, combined with AI/Machine Learning (ML) algorithms to classify sessions and detection deviations from learned behaviours. Through this tool, we will perform network monitoring and anomaly detection, also introducing the possibility of using XAI for network traffic classification. The preliminary outcomes of our experimentation will help to extend the AI-based MMT monitoring framework with transparency and interpretability.

The chapter is organized as follows: Section 2 will present the MMT architecture and its usage for anomaly detection and network classification. Section 3 will propose the usage of XAI for the classification of network traffic. Several algorithms like SHAP and LIME are presented and included in MMT as plugins to existing Deep Learning (DL) algorithms. Section 4 will present the first results demonstrating the interest of using XAI in network traffic classification in general and in anomaly detection in particular.

2 Network monitoring approaches: MMT monitoring framework example

Network monitoring is the process of observing and analyzing the performance and security of a computer network [5]. It involves collecting and analyzing data on network activity, such as the amount of data being transmitted, the types of data being transmitted, and the sources and destinations of the data.

Classification is an important aspect of network monitoring. It involves identifying network traffic and categorizing it into different types, such as email, web browsing, or file sharing. This can be done through the use of machine learning algorithms, which analyze patterns in the network traffic to identify different types of activity. By classifying network traffic, network administrators can identify potential security threats, such as suspicious or unauthorized activity, and take appropriate action to mitigate the risk. They can also gain insight into network usage, identifying trends and patterns that can help optimize network performance and improve user experience.

Overall, network monitoring and classification are critical components of maintaining a secure and efficient computer network. Through the use of advanced algorithms and analysis techniques, network administrators can gain a deeper understanding of their network and take proactive steps to ensure its continued success.

In the reminder of this section, we present: i) classification techniques, i.e, rule-based classification and AI-based classification, in subsection 2.1, ii) the global architecture of Montimage monitoring tool and its application for anomaly detection respectively in subsections 2.2 and 2.3.

2.1 Classification techniques

Rule-based network classification: it is a method of categorizing network traffic based on a set of pre-defined rules, that are typically based on attributes such as the used protocol, source and destination IP addresses, and port numbers. The process of rule-based classification involves: i) examination of data packets as they move through the network and ii) comparison of their attributes to a set of predefined rules. When a packet matches a rule, it is classified accordingly. For example, a packet that is identified as HTTP traffic (based on the used protocol) with a destination port of 80 (which is typically used for web traffic) might be classified as "web browsing".

Rule-based classification can be effective in identifying certain types of network traffic, such as web browsing, email, or file sharing. However, it can also be limited by its *inflexibility*, since it relies on predefined rules that may not capture all types of network traffic. In addition, rule-based classification can be *vulnerable to evasion techniques* used by attackers to disguise their activities, e.g., using non-standard ports or encryption.

Despite these limitations, rule-based classification is still a widely used method of network traffic analysis, especially in situations where the network environment is well understood and the types of traffic are relatively stable. It can be an efficient way to identify and filter out unwanted or malicious traffic, as well as provide insights into network usage and performance.

AI-based network classification: it involves the use of AI and ML algorithms to identify and categorize network traffic. These algorithms are trained on large datasets of network traffic and use statistical models to classify new data based on their patterns and their features. AI-based network classification can be more *flexible* and *accurate* than rule-based classification, since it can learn from data and adapt to new and evolving types of network traffic. For example, an AI-based classification system might be able to identify previously unknown types of traffic, such as a new type of malware or an emerging application protocol.

There are several different types of AI-based classification techniques, including supervised learning, unsupervised learning, and deep learning. Supervised learning involves training the algorithm on a labeled dataset, where the correct category of each data point is known. Unsupervised learning involves discovering patterns and relationships in unlabeled data, which can be useful for identifying new and previously unknown types of traffic. Deep learning involves training neural networks with multiple layers to learn complex data representations.

AI-based network classification has several advantages over traditional classification methods. It can provide greater accuracy and speed, allowing network administrators to quickly identify and respond to potential security threats. It can also be more scalable, since it can learn from large datasets and adapt to new types of traffic over time.

However, AI-based classification also requires significant computational resources and expertise to develop and maintain. It also raises concerns around privacy and security, since large amounts of sensitive network traffic data are required to train the algorithms. Therefore, it is important to carefully consider the risks and benefits of AI-based network classification before implementing it in a network environment.

2.2 Global MMT monitoring architecture

The MMT monitoring framework is an open source monitoring solution developed by Montimage and freely available for the research community on github [18]. Its workflow is presented in the Fig. 1 and hereinafter we analyze each MMT component.

Features Extraction: it is the functionality of the module "***MMT-Extract***" that allows to parse the network traffic, identify sessions and compute packet and session attributes called features. This module is implemented as a C library that analyzes network traffic to extract network and application-based events. Extraction is powered by a plugin architecture that allows adding new protocols or application message formats to parse. In the current development, more than 600 plugins for classical protocols and applications are already implemented.

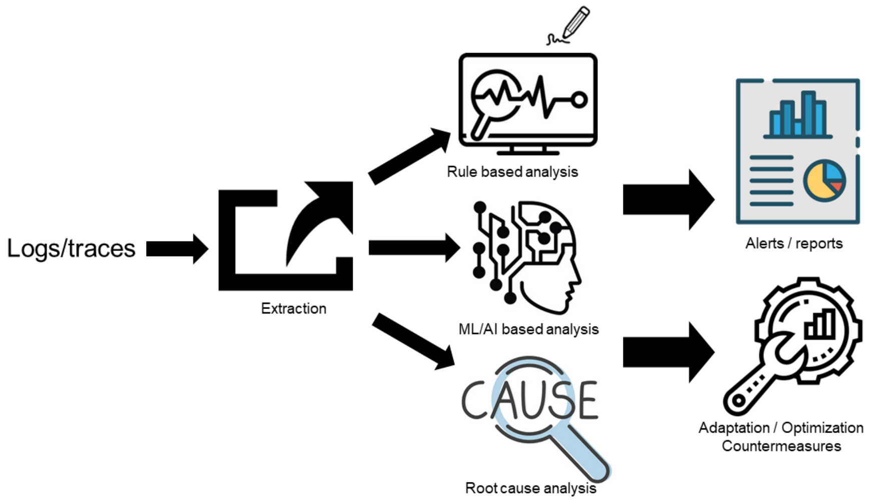


Fig. 1: Monitoring components of MMT.

Rule-based analysis: "***MMT-Security***" is a signature-based monitoring solution, that allows analysing network traffic according to a set of properties. These properties contain signatures that formally specify security goals, or malicious behaviours related to the monitored system. The MMT-Security property model is inspired by Linear Temporal Logic (LTL) and can be referred to the following two types of properties.

1. Properties that describe the *normal, legitimate behaviour* of the application or protocol under analysis. Consequently, the non-respect of the property indicates a potential violation of a safety or security requirement; e.g., all the ports in a computer must be closed unless they are being used by an authorised application.
2. Attacks that describe *malicious behaviour* corresponding to an attack model, a vulnerability or misbehaviour. In this case, the respect of the property indicates the detection of a potential incident; e.g., a big number of requests in a short period of time could be a DoS attack.

The chosen language of "MMT-Security" properties is XML format, due to its simplicity and straightforward structure verification. A property is a general ordered tree as shown in Fig. 2, where the leaf nodes are the atomic events captured in the traces. Each property is composed of a context, in the left branch, and a trigger, in the right branch. Then, a property is valid when the trigger is valid, and the trigger is inspected only if the context is valid.

Machine Learning based anomaly detection: "***MMT-AI***" allows to perform AI-based analysis of the collected features applying one or several AI/ML

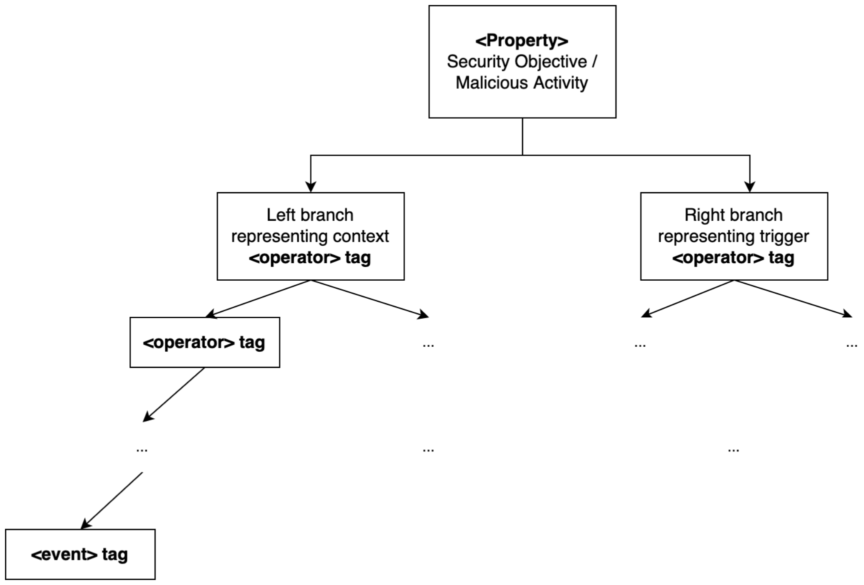

<Property>
Security Objective / Malicious Activity
Left branch representing context **<operator> tag**
Right branch representing trigger **<operator> tag**
<operator> tag
<event> tag

Fig. 2: Security property structure in MMT [18].

algorithms. It is responsible for building a model (which depends on the data and the chosen ML algorithm), as well as utilising existing ones. We can therefore distinguish its two modes of operating, i.e., training and prediction.

1. *Training*: it is designed to create and parameterize the model based on already cleaned and transformed data. This means that it executes the algorithm selected by user and a model is built by using the training data in order to find its weights and biases that would lead to the best results. Loss function, with penalised bad prediction, is used as a metrics of result during training. Depending on the selected algorithm, this step also includes the experimentation of algorithm parameters different values, such as learning rate, activation functions, batch size and so on. In the proposed system, it is assumed that this step is either done by a user a bit familiar with hypertuning, or it is done by utilisation of the values directly suggested by the system.
   – *Model evaluation.* In order to evaluate the sufficient amount of parameters and the model training, the training needs to be done using a separate dataset from the testing dataset, thus the model will be tested on the completely new samples. In this case the evaluation checks whether the model is generalised enough. To investigate the results, the correctness of classification is verified using the following terms: i) *True Positive* (TP) and *True Negative* (TN) are samples that are correctly assigned to the normal and anomaly classes respectively; ii) *False Positive* (FP), *False Negative* (FN) are samples that are incorrectly assigned to positive or negative classes.

2. *Prediction*: it is the activity done after the model is trained (that's to say the algorithm is executed in training mode), and the satisfactory results are obtained. It involves utilising the model directly on new, unseen data (in a real-life case scenario these are just production data) and obtaining the results, such as probabilities, classifications, etc. Importantly, the accuracy of the prediction results can also be used in order to further hyper-tune the model.

As the system aims to simplify the prototyping and utilisation of AI/ML algorithms for practical applications, it is assumed that the user may want to create multiple different models. Therefore, instead of selecting one model's predictions from one particular model, it can be beneficial to combine the results of different models together. Thus, this final (and optional) step of the ML module consists of the ensemble part that is capable of joining the results together.

Root Cause Analysis: "***MMT-RCA***" relies on machine learning algorithms to identify the most probable cause(s) of detected anomalies based on the knowledge of similar observed ones. It enables the systematization of the experience in dealing with incidents to build a historical database and verify whether a newly detected incident is similar enough to an observed one with known causes. Thanks to MMT-RCA's suggestions, remediation actions could be timely and wisely taken to prevent or mitigate the damage of the recurrence of problems.

2.3 Application of MMT for Anomaly detection

MMT-AI has been used in several projects e.g., for differentiating bots and human activities in the net [35], for anomaly detection in industrial systems (e.g., Load Position System of ABB) [7]. In the following, we present a classical usage of MMT-AI on an open-source database CSE-CIC-IDS2018 provided by the Canadian Institute for Cybersecurity [15].

Settings *Stacked AutoEncoders* (SAE) [8] and *Convolutional Neural Network* (CNN) [6] are used to train and classify the network traffic with the Canadian dataset. More in detail, SAE are multiple encoders stacked on top of one another. The number of neurons in each decoder and encoder are the same. They aim at dimensionality reduction, i.e., filtering the essential features from the data. Then, CNNs are used and they are a specialized type of artificial neural networks that use, in at least one of their layers, a mathematical operation called convolution in place of general matrix multiplication. It consists of an input layer, hidden layers, which perform convolutions, and an output layer. The general implemented architecture can be seen in Fig. 3. The advantage of this architecture is its flexibility, as both modules can be easily added to the structure of the the global system and integrated in the final solution.

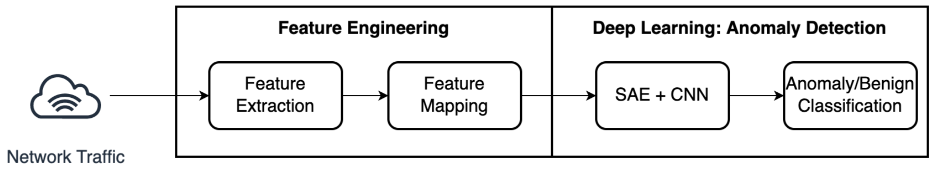


Fig. 3: *The AI-based anomaly detection architecture.*

Features extraction: the feature extractor module is used in both generating the training/testing datasets, and on the input file for prediction.

Scaling: data scaling is the treatment data process in order to obtain standard format data, thus the training is improved, accurate and faster. Indeed, a model with large weight values is often unstable, which means that it may give poor performance during learning and have a high sensitivity to the input values, that leads to a higher generalization error. Column normalization involves bringing the column values to a common scale, which is usually done for columns with varying ranges.

MinMaxScaler from Scikit learn library [34] transforms features by scaling each feature to a given range. This estimator individually scales and translates each feature, in this way values are limited to a given range in the training set, e.g., between zero and one. The transformation is given by the following equations:

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

$$X_{scaled} = X_{std} \times (X_{max} - X_{min}) + X_{min} \tag{2}$$

where (X_{min}, X_{max}) represents the desired range of scaled data, e.g., $(0, 1)$.

Training: in the learning phase, the model is fed with the so-called training dataset, and the model is tested in order to obtain the best performance and highest accuracy of the final classification. The input files pass through the features extractor module which runs the MMT-Extract. Then, it creates a training and testing .csv files with balanced 0/1 classes. More in detail, the dataset is divided in this way: 70% for training and 30% for testing.

At this level, there is the possibility that we must do multiple experiments with the use of different parameters of the models, e.g., CNN or SAE. Thus, additional model adaptation towards specific conditions or changes are recommended in order to obtain higher performance and accuracy of the model that will be saved for prediction purpose later. The model's structure is hybrid, composed of two auto-encoders and one dimensional CNN as shown in Fig. 4.

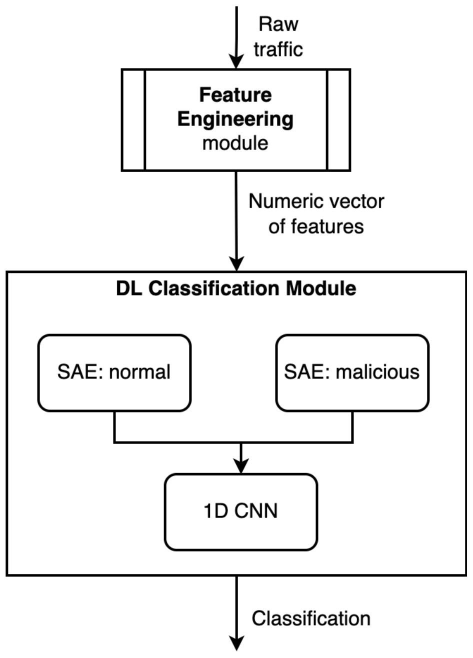


Fig. 4: *Overview of the deep learning modules.*

Evaluation We applied the learned model to the 30% of remaining datasets. The classification is done and the results are presented in the next subsection.

Results and interpretation: using the default parameters, the training on these datasets gives the following results, as shown in Table 1 and Table 2.

Table 1: *Confusion matrix 1.*

	0	1
0	6779	14
1	3	6790

Table 2: *Metrics of model using default parameters.*

	Precision	**Recall**	**F1-score**	**Support**
0 (normal traffic)	0.999558	0.997939	0.998748	6793
1 (malware traffic)	0.997942	0.999558	0.998750	6793
Accuracy	0.998749	0.998749	0.998749	0.998749
Macro average	0.998750	0.998749	0.998749	13586
Weighted average	0.998750	0.998749	0.998749	13586

The obtained results are quite impressive and are more than 99% for the precision, recall and F1-score, as shown in Table 3 and Table 4. However, better results are still possible by refining parameters.

Table 3: *Confusion matrix 2.*

	0	1
0	6778	5
1	3	6790

Table 4: *Metrics of model using advanced parameters.*

	Precision	**Recall**	**F1-score**	**Support**
0 (normal traffic)	0.999558	0.999264	0.999411	6793
1 (malware traffic)	0.999264	0.999558	0.999411	6793
Accuracy	0.999411	0.999411	0.999411	0.999411
Macro average	0.999411	0.999411	0.999411	13586
Weighted average	0.999411	0.999411	0.999411	13586

The same methodology has been applied to real traffic data collected in the Montimage internal network (private network). The train and test datasets present 15000 samples together and as already said, the dataset has been split into 70% for training and 30% for testing. The data are shuffled in order to give different patterns of the presence of 0/1 (normal/malicious) samples in the files. After the training process on these data, we obtained the following results, as shown in Table 5 and Table 6.

Table 5: *Confusion matrix 3.*

	0	1
0	7500	0
1	87	7413

The results were good enough to affirm that the obtained model is efficient. Thus we further investigated the classification phase in order to check the accuracy of the prediction of the model. We used a portion of the raw normal traffic data that we used for training. In this way, we know that the model is correctly functioning when we see zeros in the predicted malware value. However, the results were not compatible with the expected values and the predictions are thus not correct. The model predicts that there were attacks in the known normal traffic. Therefore, further investigation and testing need to be done.

Table 6: *Metrics of model using real network traffic.*

	Precision	**Recall**	**F1-score**	**Support**
0 (normal traffic)	0.999558	0.999264	0.999411	6793
1 (malware traffic)	0.999264	0.999558	0.999411	6793
Accuracy	0.999411	0.999411	0.999411	0.999411
Macro average	0.999411	0.999411	0.999411	13586
Weighted average	0.999411	0.999411	0.999411	13586

Discussion: the manual investigation shows that the classification using AI-based anomaly detection provides in some cases false positives (e.g., 0.1% for malware prediction using default parameters) which are difficult to interpret mainly with theoretical metrics that are more than 99% (precision, recall, f1-score etc.). The need to have more transparency is needed in such context to better interpret the results and understand why we have such decisions. That's why we will use XAI to have a better insight on network traffic classification using Explainable AI. These results are still preliminary.

3 Interpreting ML models for user network activities classification

3.1 Motivation

Context: network traffic classification becomes more and more challenging due to the growth in network traffic. As there are new applications with different characteristics and network requirements, it is crucial to identify the requirements to provide the appropriate resource to each application. In the literature, several approaches have been proposed for network traffic classification based on the well-known ports (e.g., TCP or UDP port numbers), and on Deep Packet Inspection (DPI) technique [27]. However, port-based classification technique is ineffective because mapping between ports and applications using dynamic ports is not well defined. Moreover, the growing popularity of *encrypted traffic* HTTPS and Virtual Private Networks (VPN) increases user security and privacy, but also becomes a big challenge for traditional traffic analysis, making DPI-based service classification unfeasible. Therefore, it raises the need for advanced analysis techniques based on other criteria, such as behavior analysis. With the introduction of network encryption techniques, such as the TLS protocol, the accuracy and efficiency of conventional Network Intrusion Detection Systems (NIDS) that were using rule- and signature- based monitoring detection methods is greatly reduced. Consequently, in the last decade, research efforts have moved towards new analysis methods based on AI techniques for network traffic classification. Indeed, various AI algorithms have been used in the literature, such as supervised [27,28], unsupervised [29], and hybrid machine learning approaches [30,31,32].

Nowadays, apart from accuracy and performance, new requirements concerning trustworthy, transparency, unbiasedness, privacy, robustness also need to be taken into account in the development of AI-based systems. Nonetheless, existing AI methods, especially complex ones like Deep Neural Networks, are seen as black boxes, thus have a common limitation of *lacking explainability*. Indeed, the classification results of existing work do not provide the users with any information of how the dataset, input features or selected models contribute to the predicted classification. In this context, user network activities classifiers, as well as other traffic analysis applications, must be improved and optimized not only in terms of performance but also for other properties listed above. Recently, *eXplainable Artificial Intelligence (XAI)* has become a hot research topic in the AI community [10]. It provides a rationale that allows users to understand why an AI-based system has produced a given output and increases trust of end-users. Different approaches [21,24] are proposed to providing and improving the understanding, in the global and local manner, of what the models have learned and how the models make individual predictions.

Proposal: our work aims at characterising and classifying user network activities using machine learning techniques. We use popular supervised techniques, such as Random Forest, neural networks, XGBoost[1] and LightGBM[2], and unsupervised techniques, such as K-Means, for classification. Furthermore, we want not only to understand why our application produces promising results, but also why it makes some wrong predictions in some cases to further improve the performance. To achieve this goal, we add an extra explainability layer on top of our AI-based classification system by applying different popular XAI methods, such as SHAP and LIME. The full dataset and the code of the AI-based system will be published along with this paper at [14].

3.2 Classification of user network activities

Overview Our classification system takes as input network traffic data with IP and TCP/UDP headers fields. Figure 5 illustrates an overview of the workflow of our AI-based classification consisting of four main phases: *dataset generation*, *dataset preprocessing*, *feature extraction* and *classification*. The dataset generation process is indeed important for training and testing our application. The dataset preprocessing phase is required to describe and transform the input network traffic data into a set of features suitable for the classification task. Then, different classification models are executed using the feature selection output to predict the user activity in one of three groups: Web, Interactive and Video. Below, we describe the four phases in more details.

Types of activities: we choose the most common user activities on the Internet covering behaviors exhibited by the network traffic from different applications. The set of three classes are as follows:

[1] https://xgboost.readthedocs.io
[2] https://lightgbm.readthedocs.io

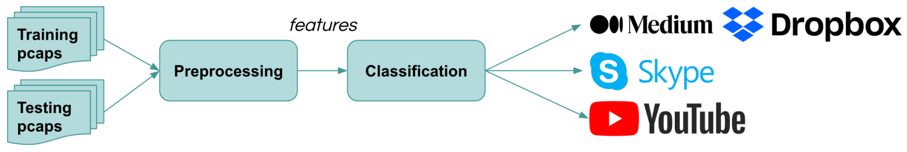


Fig. 5: Overview of AI-based classification for user activities.

1. **Web browsing** activity includes the network traffic generated when users search or view different web pages, including downloading of multimedia content such as text, images or advertising video. This activity can also include traffic of applications that transfer of big volume data over the network.
2. **Interactive** activity contains network traffic of applications that execute real-time interactions, for example chatting application like Discord, Messenger or remotely editing files on Google Docs.
3. **Video** activity contains network traffic of applications consuming video in streaming mode, for example watching online movies or Youtube videos.

Dataset generation: Figure 6 depicts the overview of our methodology used to capture network traffic to generate the dataset for training and evaluating the classification system. First, we capture the traffic using Wireshark [17] to produce pcap files whose size is 2.15 Gb when a user performs some normal activities on a single host. Concretely, for each class, we perform the following activities: Web (web browsing in blogs, social networks and shopping sites), Interactive (chatting application like Discord, Messenger), and Video (watching Youtube and movies). Therefore, the main dataset is composed of several network traffic traces, each belonging to a specific user activity. Then, we filter those pcap files so that there is one source IP and one destination IP for each pcap file. Also, all packets with an empty payload and non-TCP or -UDP packets should be filtered out.

We use our open source Montimage security monitoring framework (MMT) [18], in particular "***MMT-Probe***" module [19] to convert pcap files into csv reports. We can also visualize pcap files with "***MMT-Operator***" [20]. The .csv files characterise the network traffic using the following features: the timestamp, the protocols, the source and destination IP addresses, the payload size in bytes, the number of packets, etc. The full dataset in both pcap and csv formats is being published along with this paper at [14].

Dataset preprocessing: some activities may produce multiples traces, for instance, MMT-Probe converts a single pcap file capturing a Video activity into multiple csv files. Therefore, first of call we need to merge those csv files belonging to a single activity into a single csv file for further analysis. Next, we only select interesting data concerning the network traffic in the merged csv reports and also compute additional statistics values, like data aggregation. We come up with 21 features that will be described later, as shown in Table 7. Finally, the

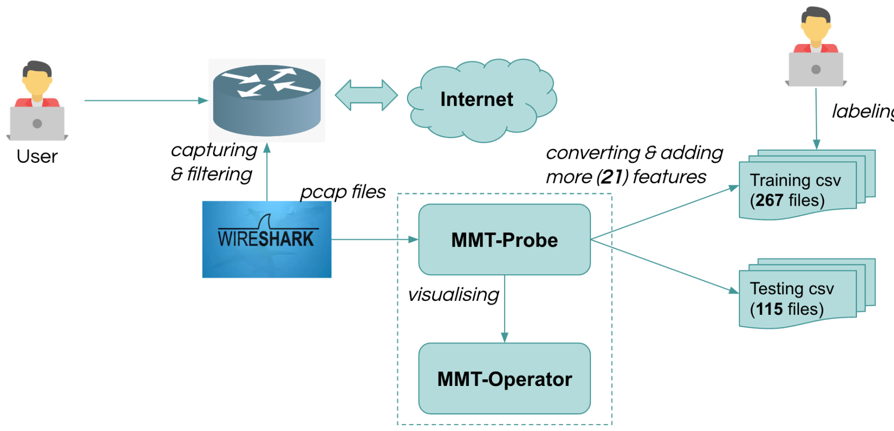


Fig. 6: Traffic dataset generation.

full dataset consists of 382 labeled traces for three traffic classes. The number of traces in Web, Interactive and Video activities are 304, 34, 44, respectively. The final set of processed csv files will be used through the analysis and evaluation of our AI-based classification system.

Feature extraction: Table 7 shows 21 features of 5 main categories. In the group *Duration*, the feature *session_time* shows the total time wherein a user interactions with applications when performing an activity. In the group *Protocol*, there are 2 features corresponding to the percentage of TCP or UDP traffic. Normally, the two features values always add up to 100%. In the group *Uplink* and *Downlink*, we use some common features of uplink and downlink communication, such as data volume in bytes, number of packets and percentage of packets to the total packets. In addition, we also compute other useful values, such as the maximum, minimum, mean and standard deviation. Finally, in the group *Speed*, we add two more features concerning the network speed. This set of features is optimal and suitable to describe network traffic in our scenario.

Classification: we implemented the classification application in Python version 3.10 using the open source popular ML libraries, such as *Scikit-learn*[3], *Keras*[4] and *TensorFlow 2*[5]. Data pre- and post- processing have been performed using the *numpy* and *pandas* libraries. The graphical plots have been obtained using popular libraries, like *matplotlib* and *seaborn*. For the deep learning model, we used a Sequential model as our network consists of a linear stack of layers from

[3] https://scikit-learn.org
[4] https://keras.io
[5] https://www.tensorflow.org

Table 7: Features selection.

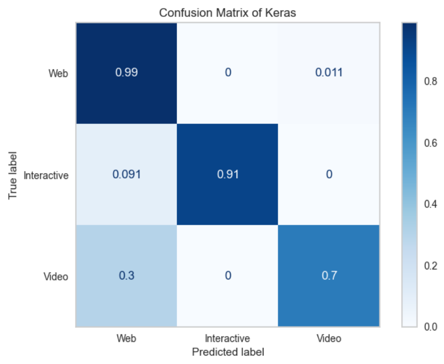
	Category	**Feature Id**	**Feature Description**
1	Duration	*session_ time*	total time wherein a user interacts with apps
2	Protocol	*%tcp_ protocol*	percentage of TCP traffic
3		*%udp_ protocol*	percentage of UDP traffic
4	Uplink	*ul_ data_ volume*	uplink data volume in bytes
5		*max_ ul_ volume*	maximum of uplink data volume
6		*min_ ul_ volume*	minimum of uplink data volume
7		*avg_ ul_ volume*	average of uplink data volume
8		*std_ ul_ volume*	standard deviation of uplink data volume
9		*%ul_ volume*	percentage of uplink data volume
10		*nb_ uplink_ packet*	number of uplink packets
11		*ul_ packet*	percentage of uplink packets
12	Downlink	*dl_ data_ volume*	downlink data volume in bytes
13		*max_ dl_ volume*	maximum of downlink data volume
14		*min_ dl_ volume*	maximum of downlink data volume
15		*avg_ dl_ volume*	average of downlink data volume
16		*std_ dl_ volume*	standard deviation of downlink data volume
17		*%dl_ volume*	percentage of downlink data volume
18		*nb_ downlink_ packet*	number of downlink packets
19		*dl_ packet*	percentage of downlink packets
20	Speed	*kB/s*	number of kB per second
21		*nb_ packet/s*	number of packets per second

the Keras library. Concretely, a fully-connected network structure has three layers: in the first two layers we used the most widely used activation function, i.e., Rectified Linear Unit (ReLU), while in the output layer we used the Sigmoid activation function. In addition, we used the default models of XGBoost, LightGBM and Random Forest for classification.

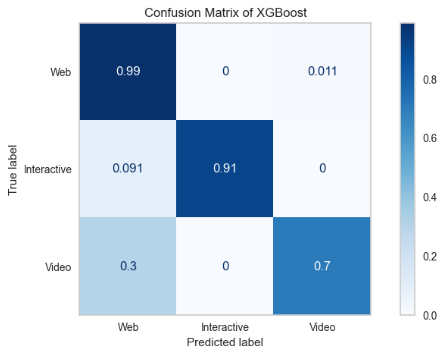
The full dataset and the code to perform the evaluation of the experiments can be found at [14]. For the evaluation, we randomly split the main dataset, including 382 traces, into the training and testing datasets with a probability of 70% to perform cross validation. Concretely, we used 267 traces to building and training the models and evaluate them against the testing dataset of 115 traces.

3.3 Evaluation

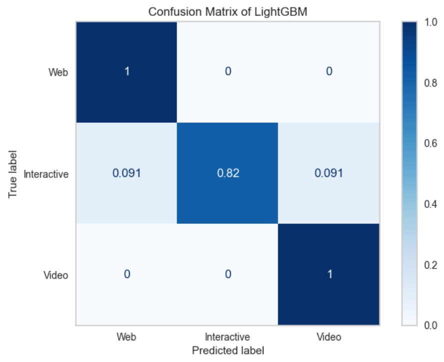
Metrics: we measure the performance of classification models using some popular metrics [16], such as the accuracy, precision, recall (or sensitivity) and F1-score metrics. Those metrics are defined in the following equations (3), (4), (5) and (6) and are applied to any classification models. More specifically, *TP*, *FP*, *FN* are the number of True Positive instances (correctly classified), the number of False Positive instances (incorrectly classified as a class), and the number of False Negative instances (incorrectly classified as another class), respectively.



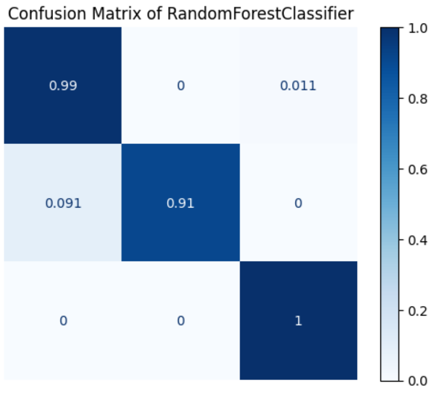
(a) Keras model.



(b) XGBoost model.



(c) LightGBM model.



(d) Random Forest Classifier model.

Fig. 7: Confusion matrices of 4 models for the testing dataset.

The F1 score metric takes both precision and recall into account. All those metrics values are in range from 0 to 1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{6}$$

Supervised classification models: Table 8 contains the metrics values of the classification results from the testing dataset for each user activity. Moreover, Figure 7 provides confusion matrices of three classification models to better visualize the relationships between different user activities. While the rows of the confusion matrix illustrate the predicted classification distribution for each user activity, the columns represent the true activity distribution for each predicted class. In addition, the recall for each class is shown in the main diagonal of the confusion matrix.

Table 8: Experimental results of activity classification for the testing dataset.

Activity	**Keras**			**XGBoost**			**LightGBM**			**Random Forest Classifier**		
	Precision	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Web	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	1.00	0.99	0.99	0.99
Interactive	1.00	0.91	0.95	1.00	0.95	0.91	1.00	0.90	0.82	1.00	0.91	0.95
Video	0.78	0.70	0.74	0.88	0.78	0.70	0.91	0.95	1.00	0.91	1.00	0.95
	Accuracy: 0.95			**Accuracy**: 0.96			**Accuracy**: 0.98			**Accuracy**: 0.98		

As shown in Table 8 and Figure 7, the accuracy values of four classification models, including the Keras model, the XGBoost model, the LightGBM model and the Random Forest Classifier model, are outstanding with 95% (5 wrong predictions), 96% (5 wrong predictions), 98% (2 wrong predictions) and 98% (2 wrong predictions), respectively. Other metrics like the precision, recall and F1 scores are mostly over 95%, for all except Video activities. Among four classification models, the LightGBM model and the Random Forest Classifier model have the best performance against the testing dataset as they predict correctly all instances of Web and Video activities, while the other two models Keras and XGBoost have the same 4 wrong predictions. Interestingly, for Interactive activities classification, the overall best classification model LightGBM performs worst with 2 wrong predictions, while others have only one. The results suggest that AI classification models are complements to each other, and combining those supervised models could give us better results.

One possible explanation of the worst results of those models in classifying Web and Video activities is that users may perform unintentionally those activities at the same time, e.g., users browser web pages that access the content in Video activities or some advertising videos pop up on web pages. Furthermore, Web activities have variable behaviors in different forms and share its feature space with other types of user activities.

3.4 Explainable AI (XAI)

State-of-the-art of XAI method: eXplainable AI (XAI) [10] is a promising set of technologies that increases the AI black-box models' transparency to explain why certain decisions were made. While AI plays a critical role in different domains, XAI is crucial to enhance trust and transparency for people to use future AI-based applications. For instance, in the previous subsections, we employed different AI models for user activity classification and achieve very good results but still incorrectly classify some instances. However, those models are complex with multiple input features and not readily interpretable by design, thus hindering users or even developers to understand and debug them to improve the performance of the AI-based system. Therefore, we need to build an explainability layer on top of the AI models to provide post-hoc explainability and enhance their interpretability. As depicted in Figure 8, some popular post-hoc explainability methods are *visual explanations*, *local explanations*, *explanations by example*, and *feature relevance explanations*.

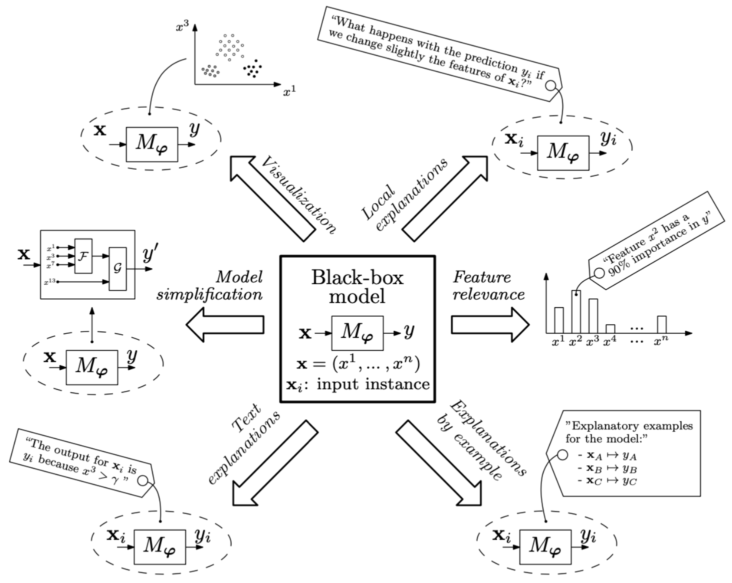

Black-box model
$\mathbf{x} \to M_\varphi \to y$
$\mathbf{x} = (x^1, \dots, x^n)$
$\mathbf{x}_i$: input instance
Visualization
Local explanations
Feature relevance
Explanations by example
Text explanations
Model simplification
"What happens with the prediction y_i if we change slightly the features of $\mathbf{x}_i$?"
"Feature x^2 has a 90% importance in y"
"Explanatory examples for the model:"
- $\mathbf{x}_A \mapsto y_A$
- $\mathbf{x}_B \mapsto y_B$
- $\mathbf{x}_C \mapsto y_C$
"The output for $\mathbf{x}_i$ is y_i because $x^3 > \gamma$"
y'
$\mathcal{F}$
$\mathcal{G}$
x^1 x^2 x^3 x^4 ... x^n

Fig. 8: Conceptual diagram showing different existing post-hoc explainability methods for ML models [10].

- ***Local explanations*** aim at approximating explanations to less complex solution sub-spaces for model predictions by only considering a subset of data. *Local Interpretable Model-agnostic Explanations* (LIME) [24] is a widely popular technique used in interpreting outputs of black-box models in several fields and applications.
- ***Feature relevance explanations*** compute relevance scores of the model features to quantify the contribution or sensitivity of each feature to the model's output. *Shapley Additive Explanations* (SHAP) [21] is a popular XAI technique that identifies the importance of each feature value in a certain prediction using popular cooperative game theory technique. *Permutation Feature Importance* is a global XAI method that measures the increase in the

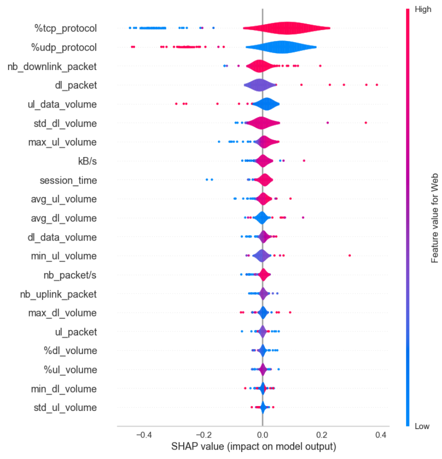
prediction error of the model after we permute the feature's tabular values. To assess how important a specific feature is, we compare the initial model with the new model on which the feature's values are randomly shuffled.

– ***Explanations by example*** consider the extraction of representative data examples that relate to the result generated by a certain model, allowing to get a better understanding of the model. Some XAI methods of this category are *counterfactual explanations* [25] and *adversarial examples.*

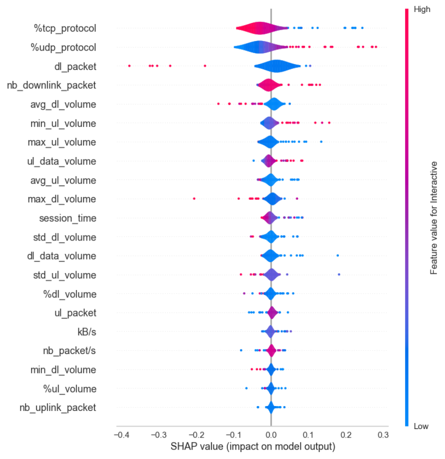
Each of those techniques covers a way in which humans explain an object and the combination of all methods provide us the whole explanations about the AI models. However, as many methods may be suitable for different types of the AI models or datasets, we need to consider the best appropriate methods for the concrete problem being solved. Next, we apply some popular XAI methods, such as SHAP and LIME, to provide both global and local explanations of the AI models that we used previously for user network activities classification. Since both SHAP and LIME are model-agnostic XAI methods, which implies that they can be applied to any ML models, we will discuss in details the explanations for the Keras model. The full results can be found at [14].

SHAP Lundberg *et al.* proposed the SHapley Additive exPlanations (SHAP) method which offer a high level of interpretability for a model [21]. The SHAP values, which are based on the concepts of game theory, provide both *global* and *local* explainability of any ML models. For global explainability, the SHAP values show how much each input feature contributes, either positively or negatively, to the model's global output. For local explainability, as each prediction has its own set of SHAP values, we can explain why the model makes a specific prediction and input features importance. Our implementation uses the *KernelExplainer* method of the SHAP library [13] to calculate SHAP values and build summary and dependence plots. Specifically, the *KernelExplainer* builds a weighted linear regression to compute the variable importance values using the dataset, the labeled outputs and the model predictions. In addition, we apply also the dedicated method *DeepExplainer* that performs calculations of the SHAP values for the Keras model faster than the previous one *KernelExplainer.*

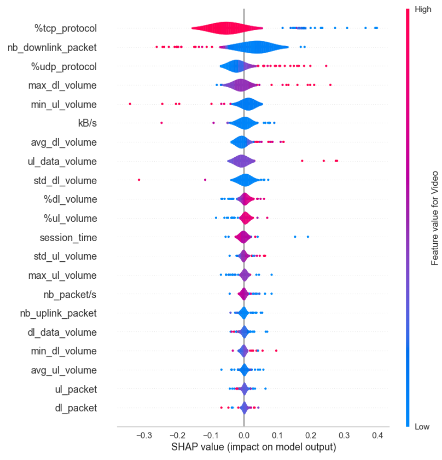
SHAP summary plots: they show the positive and negative relationships of the AI models with its outcome. Figure 9 shows SHAP summary plots for Web, Interactive, Video and all activities classification using the Keras model. The summary plot consists of many dots representing instances of the dataset. Vertical location shows the input features that are ranked in descending order in terms of feature importance. The horizontal location shows whether the effect of a single feature is associated with a higher or lower model prediction. Color illustrates whether that feature has a high (in red) or low (in blue) impact on that prediction. As depicted in Figure 9 (a), the feature %*tcp_protocol* has a positive and high impact on predicting an instance as a Web activity because of a large number of red dots on the X-axis. Similarly, we can say the feature %*udp_protocol* is negatively correlated with Web activities, but highly contributed to predict



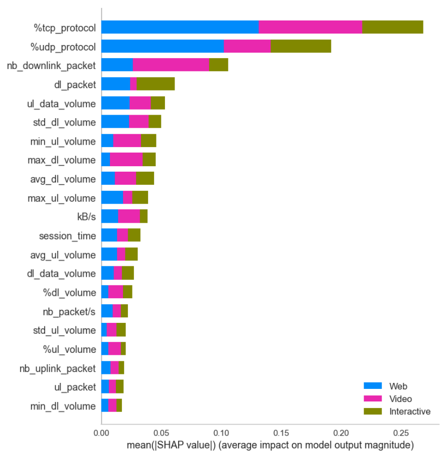
(a) Web activity.



(b) Interactive activity.



(c) Video activity.



(d) All three activities.

Fig. 9: SHAP summary plots for the feature importance of the Keras model.

Interactive and Video activities, as shown in Figure 9 (b) and Figure 9 (c). Furthermore, from Figure 9 (d), we observe that the most five important features contributing globally to our AI-based classification are *%tcp_protocol*, *%udp_protocol*, *nb_downlink_packet*, *dl_packet* and *ul_data_volume*.

SHAP dependence plots: they show the effect of a single input feature across the whole dataset. Figure 10 shows some interesting SHAP dependence plots for Web activities classification using the Keras model. Each dot in the plot represents a single prediction from the dataset. The x- and y-axis show the values of an input feature from the dataset and its SHAP values underlying how much this feature has contributed to the prediction, respectively. Similar to SHAP summary plots, the color of SHAP dependence plots corresponds to an interaction effect, like high in red and low in blue, between the input feature we are plotting and the second feature. For instance, the SHAP dependence plot of the feature *session_time* shows that this feature interacts mostly with the feature *nb_downlink_packet*, as depicted in Figure 10 (a). In addition, the Figure 10 (b) shows that there is an approximately linear and negative trend between the feature *nb_uplink_packet* and the second feature *%udp_protocol* that interacts with *nb_uplink_packet* frequently. Interestingly, from Figure 10 (c) and Figure 10 (d), the most two important features *%tcp_protocol* and *%udp_protocol* interact most with other features of the *Downlink* category, such as *%dl_volume* and *dl_data_volume*. This also explains why the *Downlink* features have great impact on the final prediction of the Keras model for user activities classifcation, as discussed above.

LIME: Ribeiro *et al.* proposed Local Interpretable Model-agnostic Explanations (LIME) method that aims to explain individual predictions of black-box AI models. While the SHAP values of a feature represent their contribution to one or several sets of features, LIME aims to provide *local* explainability that are locally faithful within the surroundings or vicinity of the sample data being explained. The LIME method is compatible with many different classifiers and can be used with image, tabular and text data. Similar to the SHAP method, LIME does not take the model into account, thus can be applied to any models. Our implementation uses the *LimeTabularExplainer* method of the LIME library [12] to calculate values and build plots.

Figure 11 shows the explanation for a single instance from the testing dataset. The leftmost values are the prediction probabilities of our classifier, that in this case is the Keras model. Concretely, the Keras model predicts correctly this particular instance as Web activity with 100% of confidence. The numbers on the right reflect the average influence of that particular feature value in the final prediction, for example as a Web/Interactive/Video activity or not. This set of values encapsulates the behavior of the LIME's linear model in the neighbourhood of the sample data that we try to explain.

Given the training dataset, having *ul_packet* > 0.84, *min_dl_volume* $<= 0.12$ and *%tcp_protocol* > 0.47 would increase on average the prediction probability of that instance being a Web activity by 0.13, 0.13 and 0.10, respectively.

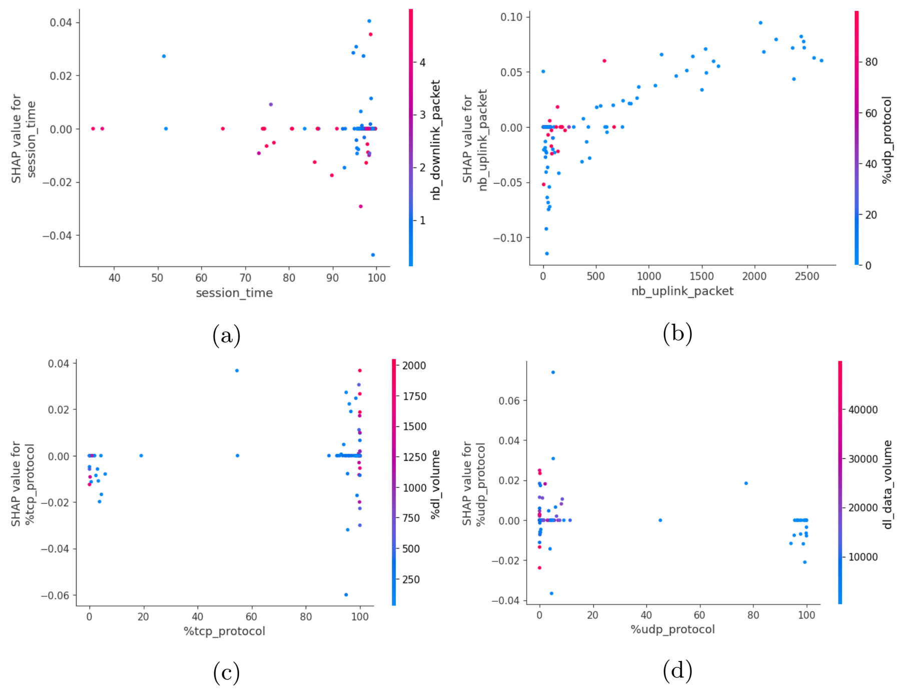
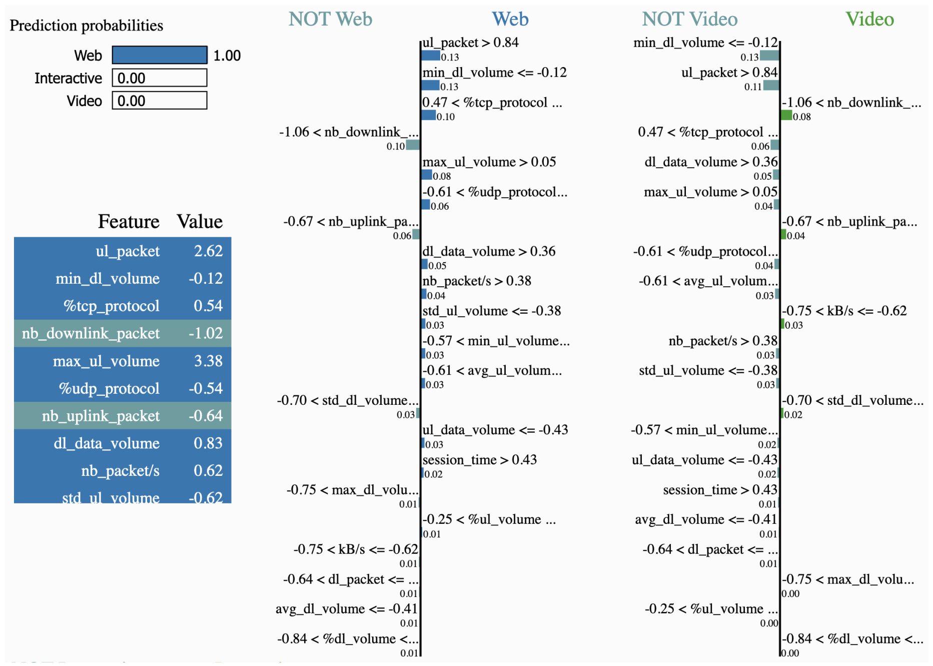


Fig. 10: SHAP dependence plots for Web activities classification using the Keras model.

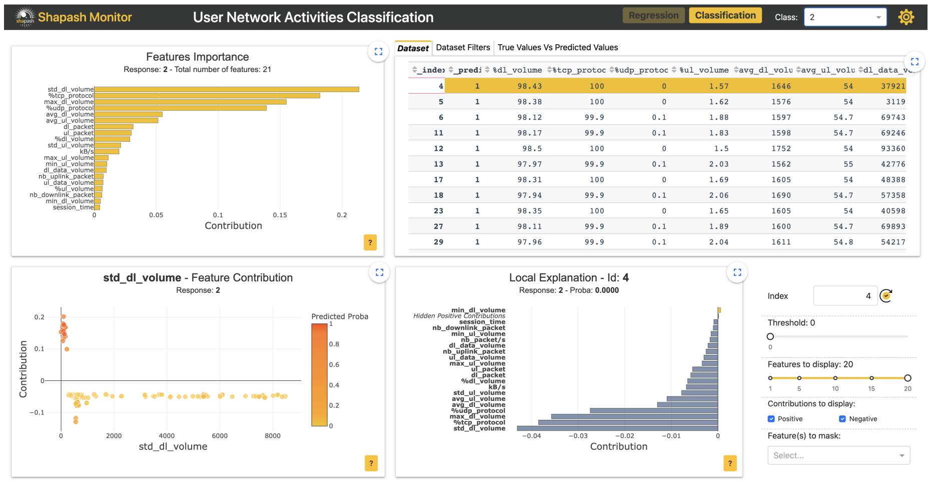
Note that the concrete features' values are in the table in the bottom left. Moreover, having $nb\_downlink\_packet > -1.06$ would decrease on average the prediction probability by 0.10. We have the similar observation by looking at the positive impact of the input feature $nb\_downlink\_packet$ with the value -1.02 on the column *Video*. This is because $nb\_downlink\_packet$ is the second most important feature in classifying an instance as a Video activity, as discussed earlier in Figure 9 (c) showing the SHAP summary plot for Video activities. Overall, by only looking at the two columns *NOT Web* and *Web*, we observe that there are more input features with a bigger positive contribution to the prediction probability of this instance being a Web activity. This is why the Keras model gives us a prediction of being a Web activity for this particular instance. We can also conclude that two XAI methods SHAP and LIME are complement to each other and provide us the similar explanations of our AI-based application for user activity classification.

Shapash Shapash [22] is an open source Python library to visualize AI models to make them reliable, transparent and understandable for everyone. It is compatible with many models, including Scikit-learn, XGBoost and LightGBM models for both classification and regression tasks. Moreover, Shapash allows



Feature	Value
ul_packet	2.62
min_dl_volume	-0.12
%tcp_protocol	0.54
nb_downlink_packet	-1.02
max_ul_volume	3.38
%udp_protocol	-0.54
nb_uplink_packet	-0.64
dl_data_volume	0.83
nb_packet/s	0.62
std_ul_volume	-0.62

Fig. 11: Illustration of the LIME method results for the prediction using the Keras model (details of *NOT Interactive* and *Interactive* are omitted).


Shapash Monitor
User Network Activities Classification
Regression
Classification
Class: 2
Features Importance
Response: 2 - Total number of features: 21
Contribution
std_dl_volume - Feature Contribution
Response: 2
Predicted Proba
std_dl_volume
Local Explanation - Id: 4
Response: 2 - Proba: 0.0000
Hidden Positive Contributions
Dataset
Dataset Filters
True Values Vs Predicted Values
Index 4
Threshold: 0
Features to display: 20
Contributions to display:
Positive
Negative
Feature(s) to mask:
Select...

_index	_predi	%dl_volume	%tcp_protoc	%udp_protoc	%ul_volume	avg_dl_volu	avg_ul_volu	dl_data_vo
4	1	98.43	100	0	1.57	1646	54	37921
5	1	98.38	100	0	1.62	1576	54	3119
6	1	98.12	99.9	0.1	1.88	1597	54.7	69743
11	1	98.17	99.9	0.1	1.83	1598	54.7	69246
12	1	98.5	100	0	1.5	1752	54	93360
13	1	97.97	99.9	0.1	2.03	1562	55	42776
17	1	98.31	100	0	1.69	1605	54	48388
18	1	97.94	99.9	0.1	2.06	1690	54.7	57358
23	1	98.35	100	0	1.65	1605	54	40598
27	1	98.11	99.9	0.1	1.89	1600	54.7	69893
29	1	97.96	99.9	0.1	2.04	1611	54.8	54217

Fig. 12: Shapash dashboard for visualizing the Random Forest Classifier model.

users to easily understand their AI models via a nice and user-friendly Web dashboard to navigate between the feature importance, global and local explainability with popular XAI methods like SHAP, Active Coalition of Variables[6] (ACV) and LIME as backend. As depicted in Figure 12, on the Shapash's dashboard of the Random Forest Classifier model, we can easily visualize and interact with the dataset, observe how each feature contributes to model predictions and local explanation of an individual instance.

Different XAI methods may give us different results or different explanations. To increase the degree of confidence of applying different XAI methods, we need to define some metrics to assess the quality of their explanations. Shapash can measure some interesting metrics [23] to assess the degree of confidence on different XAI methods as follows.

- The ***consistency metric*** compares different XAI methods and evaluates them to see how close the explanations are to each other, for example by calculating an average distance between the explainability methods. If different XAI methods lead to similar results, this would mean a higher degree of confidence can be placed in using them. If not, we would need to carefully interpret the explanations of each method to identify which one is the best.
- The ***stability metric*** evaluates the similarity between different instances under two criteria: those instances must be close in the feature space and have similar outputs. If instances are similar, we would expect the respective model output for these instances to be similar as well. Therefore, this metric allows for building trust in a specific explanation.
- The ***compacity metric*** seeks to reduce complexity and overexplaining by measuring the explainability of a decision in relation to only the most impor-

[6] https://github.com/salimamoukou/acv00

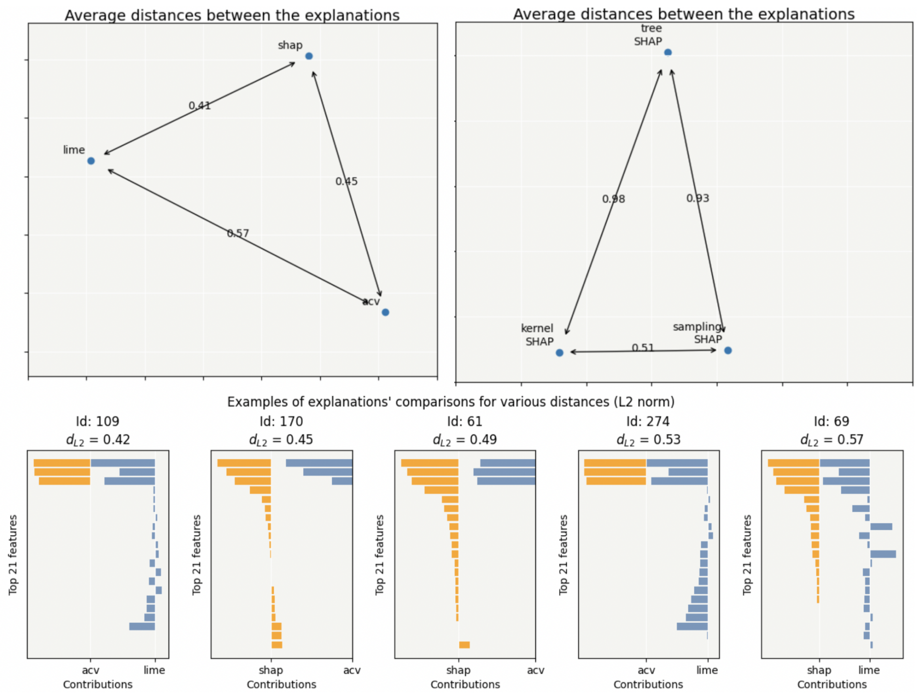


Fig. 13: Consistency of explanations provided by different XAI methods.

tant features. For each instance, after identifying feature importance using XAI methods, we select a subset of features with the highest contributions and observe how well they approximate the model.

However, Shapash has still some limitations. First, users may need to develop new or less popular XAI methods that have not been supported by Shapash. Second, multi-class classification, like our user activities classification problem is not supported yet to compute some metrics discussed above. Therefore, we *simplify* our problem for *classifying an instance as a Web activity or not*. Herein, we can employ the Shapash library to calculate some confidence metrics of the Random Forest Classifier model.

The consistency plots in Figure 13 show the average distances between different XAI methods and different types of the SHAP method. Clearly, SHAP and LIME are more similar than ACV as the average distance between SHAP and LIME is smallest. In addition, Kernel SHAP and Sampling SHAP produce more similar explanations across all the features. Also, this metric extracts 5 real comparisons from the dataset with distances similar to those in the average distance plot.

The compacity plots in Figure 14 show the link between the level of approximation, the number of required features to reach it and the proportion of the dataset on which it works. In the left graph, top 9 most importance features

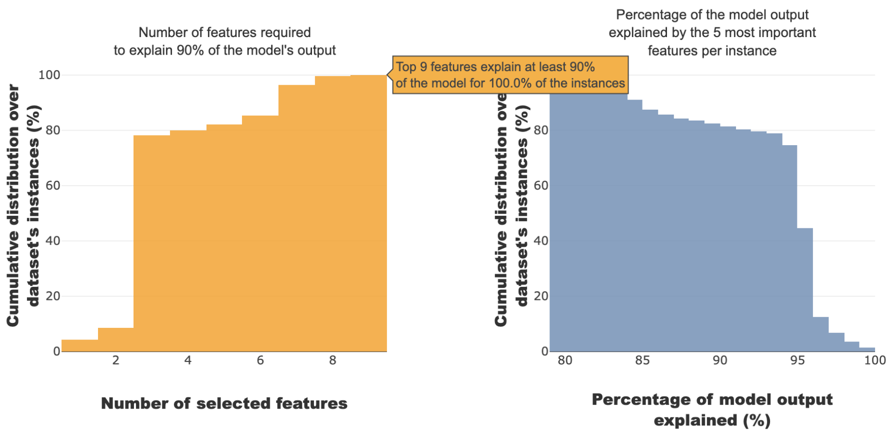


Fig. 14: Compacity of explanations.

explain at least 90% of the model for 100% of the instances. In this case, considering a small subset of features could provide a reliable explanation for almost all instances. In the right graph, top 5 features reach at least 80% of the reference model for 100% of the instances. Therefore, if we want more precise explanations, we would need to consider more than top 5 features in the explanations.

Figure 15 is the stability plot showing the neighborhood in terms of features and model's output around each particular instance. The x-axis and y-axis show the average variability of the feature across the instances' neighborhood and the average importance of the feature across the dataset, respectively. Consequently, left features are much more stable in the neighborhood than right ones and top features are more important than bottom ones. As shown in Figure 15, *%tcp_protocol*, *std_dl_volume*, *max_dl_volume* and *%udp_protocol* are important features and have strong and relatively stable contributions to the model's output. On the other hand, some features belong to the *Uplink* category, such as *%ul_volume*, *nb_uplink_packet* and *ul_data_volume*, are unstable, thus we should be careful to interpret explanations around these features.

4 Discussion

XAI, or explainable artificial intelligence, has recently gained a lot of attention as it provides insights into the black-box nature of many machine learning models. In the context of network classification, XAI can be used to provide explanations for the predictions made by the network. This can help users understand why a particular prediction was made, which can be useful for debugging the network or improving its accuracy.

XAI-based anomaly detection helps address concerns around false positives and false negatives, since the algorithm can provide insights into why certain activity is classified as anomalous, and can be refined over time to improve its

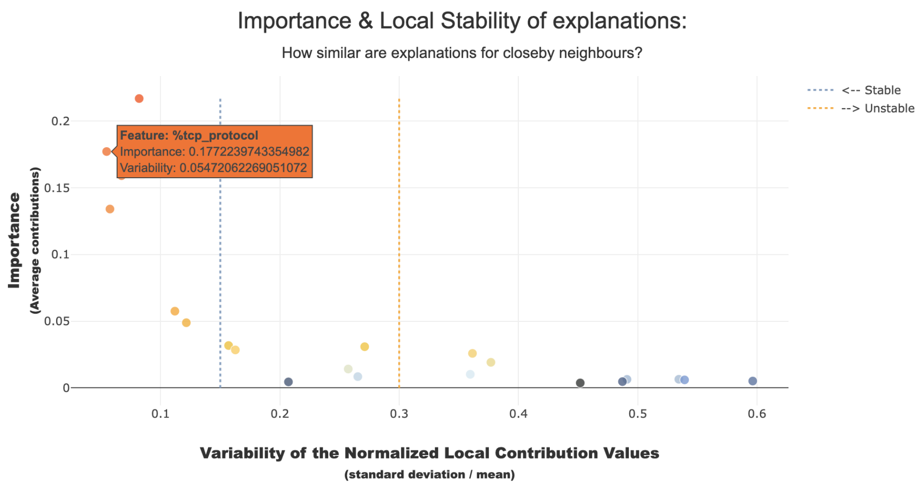


Fig. 15: Importance & Local Stability of explanations.

accuracy. However, XAI-based anomaly detection also requires significant expertise and resources to develop and maintain. It involves working with large datasets and complex statistical models, and requires a deep understanding of both network security and machine learning. Therefore, it is important to carefully evaluate the costs and benefits of XAI-based anomaly detection before implementing it in a network environment. More work on this cost assessment is planned by the authors of this chapter.

4.1 Conclusion and Future Work

In this paper, we present our AI-based application for anomaly detection and activities classification based on network traffic. We employ and evaluate different supervised learning classification models, such as Random Forest, Keras, XGBoost and LightGBM against our full dataset. The best model is LightGBM with up to 98% global accuracy. Furthermore, we provide both global and local explanations of our evaluated models using popular XAI methods, like SHAP and LIME, to have deeper insights into the dataset and the models' predictions in our scenario.

As an extension of this work, we will try to improve our classification system by considering more complex (hybrid) ML models, adding more input features and taking advantage of the complementarity of different XAI methods to extend the existing interpretability analysis. The extra explainability layer could be useful for different AI-based applications, such as Root Cause Analysis [33] or our advanced encrypted traffic analysis [35]. We also aim to produce larger datasets with more types of activities, such as data transfer, idle behavior or simultaneous activities. Some other future work can approach different security scenarios in which we may need to identify specific security applications rather than general network traffic, for example, intrusion detection, malware detection and different types of malware classification. In addition, we could use our 4G/5G testbeds [11] to generate real datasets for mobile user activities classification, similar to [26].

Acknowledgements We thank Viet Pham and Hadil Saadaoui for help with experimental setup. This work has received funding from the European Union's H2020 Programme under grant agreement no 957212 for the VeriDevOps project as well as grant agreement no 101021808 for the SPATIAL project.

References

1. Apruzzese, G., Laskov, P., de Oca, E.M., Mallouli, W., Rapa, L.B., Grammatopoulos, A.V., Franco, F.D. "*The role of machine learning in cybersecurity.*" CoRR abs/2206.09707 (2022). https:doi.org/10.48550/arXiv.2206.09707
2. Bahri, M., Salutari, F., Putina, A., Sozio, M.: Automl. "*state of the art with a focus on anomaly detection, challenges, and research directions.*" Int. J. Data Sci. Anal. 14(2), 113–126 (2022). https:doi.org/10.1007/s41060-022-00309-0
3. Dwivedi, R., Dave, D., Naik, H., Singhal, S., Rana, O.F., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G., Ranjan, R. "*Explainable AI (XAI): core ideas, techniques, and solutions.*" ACM Comput. Surv. 55(9), 194:1–194:33 (2023). https://doi.org/10.1145/3561048
4. Eltanbouly, S., Bashendy, M., AlNaimi, N., Chkirbene, Z., Erbad, A. "*Machine learning techniques for network anomaly detection: A survey.*" In the IEEE International Conference on Informatics, IoT, and Enabling Technolo gies, ICIoT 2020, Doha, Qatar, February 2-5, 2020. pp. 156–162. IEEE (2020). https://doi.org/10.1109/ICIoT48696.2020.9089465
5. Ghafir, I., Prenosil, V., Svoboda, J., Hammoudeh, M. "*A survey on network security monitoring systems.*" In: Younas, M., Awan, I., Haddad, J.E. (eds.) 4th IEEE International Conference on Future Internet of Things and Cloud Workshops, FiCloud Workshops 2016, Vienna, Austria, August 22-24, 2016. pp. 77–82. IEEE Computer Society (2016). https://doi.org/10.1109/W-FiCloud.2016.30
6. O'Shea, K., Nash, R. "*An introduction to convolutional neural networks.*" CoRR abs/1511.08458 (2015),
7. Salazar, Z., Cavalli, A.R., Mallouli, W., Sebek, F., Zaïdi, F., Rakoczy, M.E. "*Monitoring approaches for security and safety analysis: Application to a load position system.*" In the 15th IEEE International Conference on Software Testing, Verification and Validation Workshops ICST Workshops 2022, Valencia, Spain, April 4-13, 2022. pp. 40–48. IEEE (2022). https:doi.org/10.1109/ICSTW55395.2022.00021
8. Silhan, T., Oehmcke, S., Kramer, O. "*Evolution of stacked autoencoders.*" In the IEEE Congress on Evolutionary Computation, CEC 2019, Wellington, New Zealand, June 10-13, 2019. pp. 823–830. IEEE (2019). https://doi.org/10.1109/CEC.2019.8790182
9. Wang, S., Balarezo, J.F., Kandeepan, S., Al-Hourani, A., Chavez, K.G., Rubinstein, B.. "*Machine learning in network anomaly detection: A survey.*" IEEE Access 9, 152379–152396 (2021). https://doi.org/10.1109/ACCESS.2021.3126834
10. Arrieta, Alejandro Barredo, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García et al. "*Explainable Artificial*

Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." Information Fusion 58 (2020): 82-115.

11. M. -D. Nguyen, V. H. La, R. Cavalli and E. M. de Oca, "Towards improving explainability, resilience and performance of cybersecurity analysis of 5G/IoT networks (work-in-progress paper)," 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2022, pp. 7-10, doi: 10.1109/ICSTW55395.2022.00016.
12. Lime: Explaining the predictions of any machine learning classifier, https://github.com/marcotcr/lime.
13. SHAP: A game theoretic approach to explain the output of any machine learning model, https://github.com/slundberg/shap.
14. Montimage, Full dataset and code of explaining ML models for user network activities classification, https://github.com/Montimage/activity-classification.
15. CSE-CIC-IDS2018 dataset on AWS. A collaborative project between the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC). https://www.unb.ca/cic/datasets/ids-2018.html
16. Scikit-learn metrics and scoring: quantifying the quality of predictions, https://scikit-learn.org/stable/modules/model_evaluation.html
17. Wireshark, https://www.wireshark.org/.
18. Montimage tools, https://github.com/Montimage.
19. MMT-Probe: A high-performance network monitoring tool, https://github.com/Montimage/mmt-probe
20. MMT-Operator, https://github.com/Montimage/mmt-operator
21. Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions.", Advances in neural information processing systems 30 (2017).
22. Shapash makes Machine Learning models transparent and understandable by everyone, https://github.com/MAIF/shapash.
23. Shapash tutorial: Building confidence on explainability methods, https://shapash.readthedocs.io/en/latest/tutorials/explainability_quality/tuto-quality01-Builing-confidence-explainability.html
24. Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. ""Why should i trust you?" Explaining the predictions of any classifier.", Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.
25. Wachter, Sandra, Brent Mittelstadt, and Chris Russell. "Counterfactual explanations without opening the black box: Automated decisions and the GDPR.", Harv. JL & Tech. 31 (2017): 841.
26. Nascita, Alfredo, et al. "XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures.", IEEE Transactions on Network and Service Management 18.4 (2021): 4225-4246.
27. Ding, Lei, et al. "A Classification Algorithm for Network Traffic based on Improved Support Vector Machine.", J. Comput. 8.4 (2013): 1090-1096.
28. Lotfollahi, Mohammad, et al. "Deep packet: A novel approach for encrypted traffic classification using deep learning.", Soft Computing 24.3 (2020): 1999-2012.
29. Liu, Yingqiu, Wei Li, and Yunchun Li. "Network traffic classification using k-means clustering.", Second international multi-symposiums on computer and computational sciences (IMSCCS 2007). IEEE, 2007.
30. Bar-Yanai, Roni, et al. "Realtime classification for encrypted traffic.", International Symposium on Experimental Algorithms. Springer, Berlin, Heidelberg, 2010.
31. Saltaformaggio, Brendan, et al. "Eavesdropping on Fine-Grained User Activities Within Smartphone Apps Over Encrypted Network Traffic." 10th USENIX Workshop on Offensive Technologies (WOOT 16). 2016.

32. Labayen, Víctor, et al. "Online classification of user activities using machine learning on network traffic.", Computer Networks 181 (2020): 107557.
33. Luong Nguyen, Vinh Hoa La, Wissam Mallouli, and Edgardo Montes de Oca. "Validation, Verification and Root-Cause Analysis.", DEVOPS FOR TRUSTWORTHY SMART IOT SYSTEMS (2021): 173.
34. Simple and efficient tools for predictive data analysis https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
35. Montimage advanced encrypted traffic analysis tool, https://github.com/Montimage/acas