Turkish Journal of Engineering
https://dergipark.org.tr/en/pub/tuje
e-ISSN 2587-1366

Understanding Machine Learning Model Behavior for Intrusion Detection Across Attacks

Mehmet Erdi Özbek[1], Ece Gelal Soyak[*2]

[1] *Cyber Security Graduate Program, Bahcesehir University, Istanbul, Turkey, erdiozbek4@gmail.com*
[2] *Computer Engineering, Bahcesehir University, Istanbul, Turkey, ece.gelalsoyak@bau.edu.tr*

Cite this study: Özbek, M.E., & Soyak, E.G. (2025). Understanding Machine Learning Model Behavior for Intrusion Detection Across Attacks. Turkish Journal of Engineering, 2025, 9(3), 768-778.

https://doi.org/10.31127/tuje.1613468

Keywords
Intrusion Detection Systems
Machine Learning
Malware classification
Attack behavior
Cybersecurity

Research/Review Article

Received:05.01.2025
Revised:22.04.2025
Accepted:07.05.2025
Published:30.10.2025

Abstract

With the growing volume and variety of network traffic driven by various applications such as real-time communications and cloud services, combined with the increasing sophistication and frequency of malicious attempts, network administrators are facing greater challenges in securing their networks against malware. Over the past two decades, advances in machine learning and deep learning have led to a growing number of proposals for intelligent Network Intrusion Detection Systems (NIDS) that leverage these models to detect the unauthorized entry of security threats into the network. Existing studies focus on improving model accuracies, without a closer analysis of the underlying characteristics of the data. In this work, we analyze the effectiveness of NIDS mechanisms in different scenarios using different machine learning models. By examining classification performance across various data distributions -including scenarios with and without normal traffic and cases addressing class imbalance- we identify patterns in model behaviors and their correlation with attack characteristics. In our experiments, we have observed, (i) the kNN algorithm achieved the fastest training and testing times while maintaining adequate accuracy, (ii) XGBoost performed best in detecting the most commonly occurring attacks, (iii) MLP provided the highest improvement in minority class labels when resampling was applied in the dataset, and (iv) notably, while Reconnaissance attacks were consistently detected even with limited samples, detection of DoS attacks remained challenging with all models. We believe NIDS systems could benefit from the insights raised in this work based on the interplay between attack behaviors, data distributions, and model characteristics.

1. Introduction

Over the past two decades, the Internet has become indispensable for both individuals and organizations, in many applications including real-time communications, cloud-based services, the Internet of Things and autonomous vehicles [1]. This reliance on digital connectivity has also led to a parallel surge in malicious attempts aimed at disrupting the functioning of networks. Cyberattacks, ranging from sophisticated phishing schemes to large-scale distributed denial-of-service (DDoS) attacks, pose significant threats to the integrity and security of network infrastructures [2]. In response to these challenges, network administrators have widely adopted signature-based mechanisms such as firewalls, to defend the network against unauthorized access and malicious traffic. However, the integration of edge computing and network virtualization solutions, a multiplicity of new flows are introduced with diverse characteristics and thus, despite the effectiveness of traditional firewalls, the dynamic and evolving nature of cyber threats necessitates the development of more intelligent and adaptive solutions.

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions [3]. Machine learning (ML) models can be used for designing self-learning intrusion detection mechanisms. These schemes analyze the network's normal behavior as well as the characteristics of the low-footprint malware, and raise a flag when anomalies are detected. The past decade has seen the proposal of many ML -based intrusion detection schemes (IDS), competing to achieve higher success rate and lower false positives. However, many of the designed mechanisms lack a deeper understanding about the working principle of the malware types or the machine learning algorithms.

Our work evaluates a variety of machine learning models, including k-Nearest Neighbor (kNN), Naive Bayes (NB), Random Forest (RF), XGBoost, and Multilayer Perceptron (MLP), representing distance-based, statistics-based, tree-based, and neural network approaches, respectively. The models are trained on one of the widely accepted datasets, UNSW-NB15 [4], which contains a variety of contemporary attack and normal network behaviors that are representative of modern network attacks and traffic. By examining classification performance across various data distributions, including scenarios with and without normal traffic and cases addressing class imbalance, we identify patterns in model behaviors and their correlation with attack characteristics. The main contributions of this paper can be summarized as follows:

- We experimentally analyze malware detection performance using different ML approaches,
- We study the binary and multiclass malware detection separately, on datasets comprising different portions of malicious and normal traffic,
- We systematically analyze how different machine learning models behave across various malware types.

In the rest of the paper, we first summarize the prior research work on malware/intrusion detection in Section 2, then provide an overview of the malware attack types studied in this work in Section 3. The description of the selected dataset and our data preprocessing on the dataset is explained in Section 4. Our experiments on the dataset and our observations therein are explained in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

The past decade witnessed the extensive study of Network Intrusion Detection Systems (NIDS) [5]. Prior to the adoption of artificial intelligence (AI) for automating the detection of network anomalies, NIDS predominantly utilized signature-based methods [6]. Despite being accurate in detecting anomalies whose signatures were specified, these tools failed to detect zero-day attacks or identify malware that slightly altered the normal behavior [7] as well as malware that can mutate and diversify its structure (e.g. worms). As machine learning (ML) and deep learning (DL) methods became more popular and easy to implement with ready libraries, the studies on NIDS utilizing these tools significantly increased [8],[9]. To facilitate these studies, a number of artificially generated datasets have been produced.

KDD99 dataset was created in 1998 via simulations. While facilitating a majority of the initial research efforts on ML-based NIDS [10] [11], this dataset had limitations on capturing the complexity and diversity of real-world network traffic. NSL-KDD dataset was created in 2009 to address some of these limitations [12], *e.g.,* new attack categories have been included, and a more balanced distribution of attack types have been provided. More recently generated datasets include CICIDS2017 dataset, containing botnets, DoS, brute force, infiltration attack traces obtained from network traffic and system logs of university servers [13], Bot-IoT dataset including DDoS, DoS, service scan, keylogging and data exfiltration attacks [14], and HIKARI-2021 containing encrypted synthetic attack and benign traffic [15]. There have been AI-based analysis on these datasets, focusing on attacks in specific networks such as Internet of Things (IoT) [11], performing multi-class classification of intrusions [16], focusing on intrusion detection over encrypted data [17], or proposing architectures that work across multiple of the aforementioned datasets [18], [19], [20].

A high-quality NIDS dataset must include a comprehensive reflection of contemporary threats and a range of benign behaviour spanning multiple protocols. In this regard, the UNSW-NB15 dataset [4] has been adopted as a benchmark for evaluating NIDSs. The dataset contains nine different attack types explained under Section 3. Many studies utilized this dataset for designing intelligent NIDS models that identify (the presence of) attacks [21], identify attack types [22], [23], combine multiple models [24], [25], combat class imbalance [26], [27], or study the problem as a time series analysis [28].

Beyond the accuracy obtained using different ML/DL models, it is valuable to explore which features impact the model's decision. Extreme gradient boosting (XGBoost) model was used with bivariate analysis in [29], in order to compute the percentage of records in a particular value range corresponding to an attack type, and a subset of 23 impactful features were identified. [30] performed binary classification, *i.e.,* detecting whether traffic is normal or attack, and compared model performance using the top 5-100% of the features where the important features for the 10 runs were identified using the Gedeon method. In another study, the attributes were statistically analyzed and their correlations were examined [10] comparatively on the KDD99 and UNSW-NB15 datasets, where it was shown that the features in the latter dataset are more efficient. [31] analyzed feature importances with Random Forest (RF) and Support Vector Machine (SVM) on the UNSW-NB15 dataset, and investigated the impact of subsets of MQTT and TCP features on overall accuracy. Another study applied preprocessing through different sets of features, namely basic connection features, content characteristics, statistical characteristics, traffic-based and connection direction-based characteristics, to obtain the most accurate model [32].

Novelty of This Work: Although many prior efforts studied ML-assisted intrusion detection utilizing a variety of models on well-known datasets, this work differentiates from existing work by *(i)* analyzing malware classification accuracy using different portions of malicious and normal traffic samples, *(ii)* evaluating the impact of different features on the obtained performance, *(iii)* discussing the impact of different model behavior on each attack category.

3. Studied Prevalent Network Attacks

In this section, we briefly discuss the behavior of the different malware types within the UNSW-NB15 dataset. Knowing the distinctive characteristics of these attack categories is helpful in interpreting the output of the ML models in Section 5. In the documentation of the UNSW-

NB15 dataset [4], it is indicated that the IXIA Perfect Storm tool was used in generating these attack types. While the specific details of how these attacks were generated were not provided, we provided insights into these attack scenarios, to help the reader assess the corresponding entries in the dataset.

Generic attack category refers to attacks that exploit cryptographic weaknesses in protocols or mechanisms such as encryption, hashing algorithms, key exchanges. These attacks are called "generic" as they do not focus on a specific application or service, but can affect a wide range of systems by exploiting flaws in cryptographic implementations. Using the IXIA PerfectStorm tool, the generic attacks in the UNSW-NB15 dataset may be simulated via conducting man-in-the-middle attacks during cryptographic key exchanges, injecting malformed cryptographic packets to bypass security mechanisms, forcing systems to use weaker cryptographic protocols or ciphers, or simulating replay attacks. Using IDS, the attack can be identified due to the higher probability of collisions discovered between random attack attempts [33].

Exploits allow attackers to leverage existing security flaws in the network, application or operating system. In the context of the UNSW-NB15 dataset, exploits may have been generated via protocol exploits (e.g., HTTP header injections, ICMP floods). Exploit traffic generally exhibits characteristics such as anomalies in network packets (*e.g.,* too small or too large packet sizes or using rarely used protocol fields), session behaviors (*e.g.,* login failures), or communication flows (*e.g.,* many packets in short time).

Fuzzers are meant to test software robustness by sending malformed or unexpected inputs. They work by automatically generating random input to see if they cause errors such as buffer overflows, segmentation fault, etc. The traffic generated by fuzzing may look like a high-frequency burst of requests with abnormal format, length, or content. For example, if an attacker is fuzzing a web server, the traffic may consist of a large number of HTTP requests with randomized headers or body content. Fuzzing traffic might be identified by high request rates, randomized payloads, and anomalous packet sizes.

Denial of Service (DoS) attacks are originated from one or multiple devices by generating an enormous number of network requests. Although there are many types of a DoS attack, general methodology is generating network packets in microsecond level time intervals and channeling them to a singular destination. The aim is to obstruct availability of services by exhausting the network sources. In the UNSW-NB15 dataset, DoS traffic may have been generated using the IXIA PerfectStorm, by simulating a SYN flood, UDP flood, or ICMP flood, targeting a specific IP address at specified packet rates and sizes.

Reconnaissance (also called "recon" or "scanning") attack probes the network to gather information (e.g., vulnerabilities, potential entry points) for a subsequent attack. The traffic generated by reconnaissance typically consists of probes to multiple IP addresses and ports to check for open services, *e.g.,* ICMP ping sweeps to identify active hosts on a network or TCP SYN scans to identify open ports. In the UNSWNB15 dataset, the scanning behavior of reconnaissance attacks would be represented by low-volume, high-frequency scanning of different ports and IP addresses.

Analysis attacks involve gathering detailed information following an initial compromise, including post-exploitation actions like privilege escalation, mapping the internal network, and data exfiltration. The UNSW-NB15 dataset contains three subcategories belonging to the Analysis attack type, namely port scanners, HTML attacks and spams. Unlike direct attacks, their goal is to uncover insights for future exploitation like Reconnaissance [14], but with a deeper focus on identifying specific vulnerabilities, while Reconnaissance type is optimized to go through a wide network and find general information.

Backdoor attacks aim to gain unauthorized remote access to a system by bypassing normal authentication through hidden or undocumented entry points. Backdoor malware typically establishes communication with a remote command-and-control (C2) server, which can issue commands, exfiltrate data, or trigger further malicious activity. In terms of network behavior, Backdoor traffic may vary; it may present as a surge in outgoing traffic when the backdoor is used to exfiltrate large amounts of data or execute large commands. Alternatively, it may also exhibit stealthy, low-volume, and periodic communication such as small packets or Beacon signals send to the C2 server.

Shellcode malware is a subset of the Exploit attack class. The malicious code is injected into a target application to exploit and get access to the target system. Shellcodes generally open a command shell which can be controlled by the attacker. A recent study showed that shellcodes can be identified using basic network features such as service type and number of connections to the same destination IP address [33].

Worms are self-replicating malware that propagate across networks by scanning for vulnerable hosts through IP and port probing. In addition to replicating, they may deliver malicious payloads such as backdoors or data exfiltration tools. Because worms try to spread rapidly, they often generate high-volume, bursty traffic characterized by repetitive scanning and short interpacket intervals. Their network behavior, *e.g.,* packet size and interpacket time, can be used to identify them [34]. The dataset may also have a simulation of the attack behavior trying to deliver malicious payloads to other nodes.

4. Methodology

4.1. Description of the Dataset

The UNSW-NB15 intrusion detection dataset was released in 2015, to provide a comprehensive dataset containing traffic of contemporary benign and malicious activities. The dataset addresses the shortcomings of existing benchmark datasets such as lack of modern, low-footprint attack types. In the UNSW-NB15 dataset, the raw network traffic was created using a traffic generator tool; traffic was captured using tcpdump and was processed for extracting network features.

Table 1. Overview of Features in the Dataset

Category	Definition
flow info (5)	information about the endpoints of directional packet flow – addresses (*srcip, dstip*), port numbers (*sport, dport*), protocol (*proto*)
basic features (13)	statistics from packet headers or flows – *state, dur*, transmitted/received/dropped packets/bytes (*sbytes, dbytes, sloss, dloss, spkts, dpkts*), time-to-live (*sttl, dttl*), service type (*service*), data rates (*sload, dload*)
content features (8)	flow information – TCP window size (*swin, dwin*), TCP sequence number (*stcpb, dtcpb*), mean transmitted packet flow size (*smeansz, dmeansz*), HTTP transaction depth (*trans_depth*), size of HTTP data transferred (*res_bdy_len*)
time features (9)	time statistics – recorded start/end (*stime, ltime*), jitter in msec (*sjit, djit*), packet inter-arrival time in msec (*sintpkt, dintpkt*), time between TCP handshake packets (*synack, ackdat, tcprtt*)
additional features (12)	features to reveal attackers trying to collect insight on attack surface, calculated respective to previous 100 connections – src-dest IP/ports same (*is sm ips ports*), TTL for each state (*ct state ttl*), num. of HTTP flows with GET/POST (*ct_flow_http_mthd*), is FTP accessed via username+pwd (*is_ftp_login*), num. of FTP commands (*ct_ftp_cmd*), num. of flows with same src/dest (*ct_src_ltm, ct_dst_ltm*), flows with same src/dest and service type (*ct srv src,ct srv dst*), num. flows with same src and dport (*ct_src_dport_ltm*), num. flows with same dest and sport (*ct_dst_sport_ltm*), same src-dest (*ct_dst_src_ltm*)
label features (2)	Binary (Normal or malicious) (*label*) and attack type (*attak cat*)

The dataset contains 2,540,044 samples each having 49 features. The features are categorized into six main groups based on their characteristics and origin; namely flow (5), basic (13), content (8), time (9), additional (12) and label (2) features, as summarized in Table 1. 2,218,761 samples represent normal traffic, and a total of 321,283 samples represent attack behavior pertaining to nine categories, namely analysis, backdoor, DoS, exploit, fuzzer, generic, reconnaissance, shellcode and worm. The distribution of samples from each traffic type in the original dataset is shown in Table 2.

Table 2. Distribution of Labels in the Dataset

Attack Type	# Records (Orig.)	# Records (Post-Process)
Normal	2,218,761	1,959,745
Intrusion - Generic	215,481	193,933
Intrusion - Exploits	44,525	40,073
Intrusion - Fuzzers	24,246	21,821
Intrusion - DoS	16,353	14,718
Intrusion - Reconnaissance	13,987	12,588
Intrusion - Analysis	2,677	2,409
Intrusion - Backdoors	2,329	2,096
Intrusion - Shellcode	1,511	1,360
Intrusion - Worms	174	157

4.2. Preprocessing and Feature Engineering

First we handle missing values in the data (*e.g.,* 'NA', ' ', '-' , 'no'). For the features having few missing values, the corresponding rows have been dropped. Other features are treated based on context. As an example, the *service* feature had almost half of its data missing and it was not possible to predict the missing values, hence, the feature is dropped. In contrast, the missing values in the *attack category* attribute indicate the absence of attacks, *i.e., normal* traffic; thus, the missing rows have been labeled as 'normal'. The missing values in the features *ct_flw_http_mthd* or *ct_ftp_cmd*, representing a count, have been filled with zero ('0'). Hence, all missing values are handled with the respective proper approach, in an effort to keep as much data as possible.

Next, we perform exploratory data analysis (EDA) for a deeper understanding of the features. Many numerical features (*e.g., dur*, *djit*, *synack* and *ackdat*) exhibit wide value range and irregular distribution. Additionally, outliers may cause confusion on feature weighting algorithms. To address these, we monitor the skewness of numerical features using the *skew*() method, and mitigate skewed distributions through logarithmic scaling followed by normalization using the *StandardScaler*() method.

Certain categorical features (*e.g., proto* and *state*) have over 150 unique values that need to be simplified. Specifically, the *proto* feature contains 134 different protocol names, where the top 6 protocols with the highest frequencies collectively constitute more than 99% of the dataset, where TCP and UDP have 58.9% and 39% of the samples, followed by UNAS (0.6%), ARP (0.4%), OSPF (0.3%) and SCTP (0.1%). The remaining 128 categories are grouped together ('OTHER' label). The same principle is applied to the *state* feature (its categories are grouped under CON, ECO, FIN, INT, REG, RST, OTHER). Using One Hot Encoding, 14 new boolean features are generated in total, for *proto* and *state*.

Timestamp features (*e.g., Ltime* and *Stime*) were initially in integer format or converted to standard datetime (YYYY-MM-DD HH:MM:SS). To preserve their cyclical nature, such as hours ranging from 0 to 23, these values are transformed using sine and cosine functions (that vary within [−1,1]), allowing models to recognize patterns across time boundaries (e.g., between 23:00 and 00:00).

Highly correlated features increase the complexity of the ML models and withhold the model from consistently making accurate predictions. Figure 1 depicts the cross-correlation among the features. From among each pair of features with > 95% correlation, one feature is removed (adding up to a total of 9 features). Additionally, the features related to network address (*srcip*, *dstip*, *sport*, *dsport*) have been removed from the dataset.

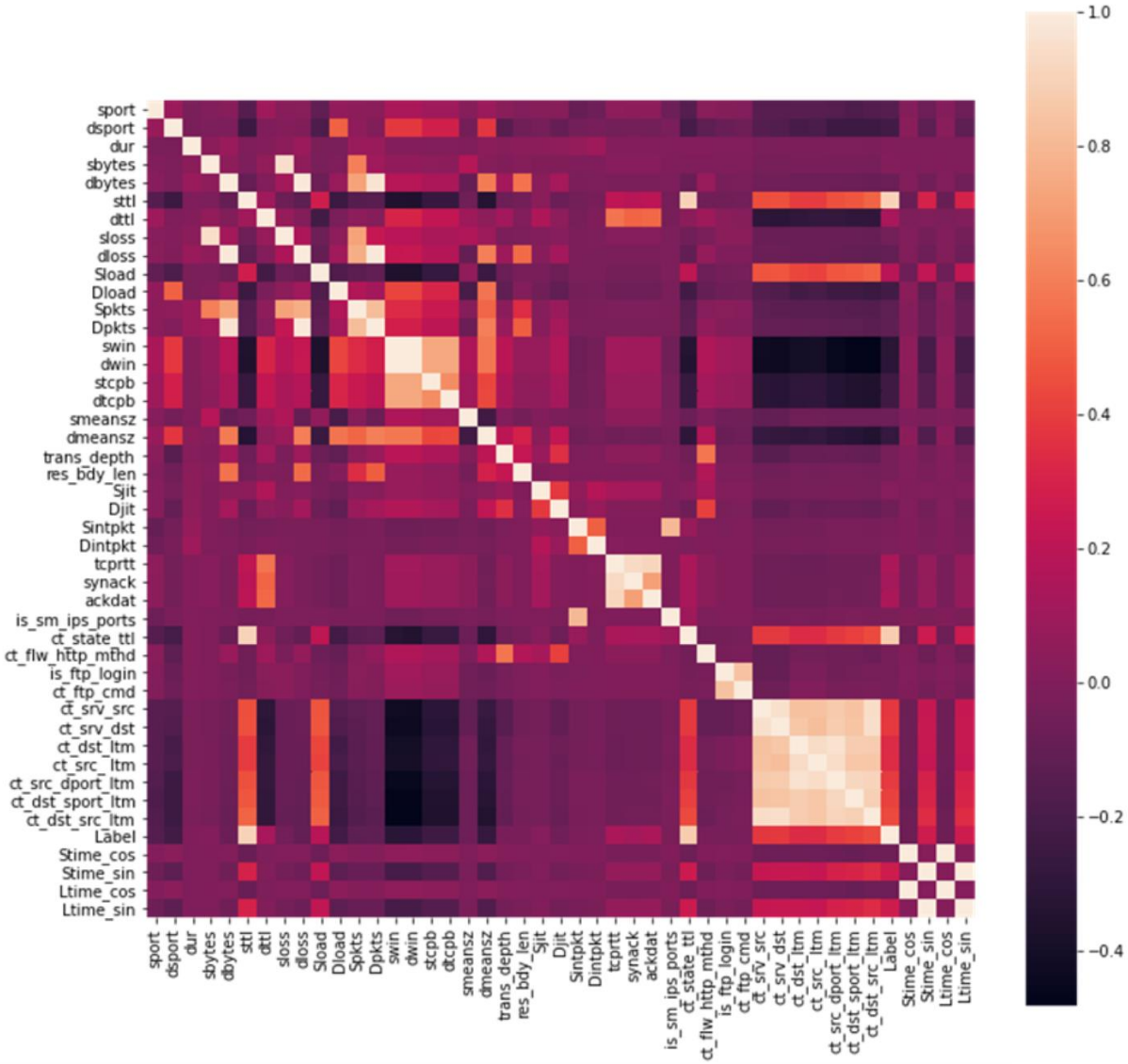


Figure 1. The Correlation Matrix of the Features in UNSW-NB15 Dataset

After preprocessing, our dataset has 51 features (including the two labels label and attack cat) and contains 2,248,900 samples.

5. Experiments

Our experiments analyze the success in detecting intrusion patterns in three setups, namely Case A, which is studied on the original dataset with 10 target classes including the normal traffic class, Case B, containing only malware data (excluding *Normal* traffic and having 9 target classes), and Case C, which contains only malware data with underrepresented categories oversampled. These setups are summarized in Table 3.

Table 3. Test Cases

	Malware Traffic	Normal Traffic	SMOTE	Size of Dataset
Case A	✓	✓	×	2,248,900
Case B	✓	×	×	289,155
Case C	✓	×	✓	333,133

We trained five machine learning models, *i.e.,* Random Forest, XGBoost, Multilayer Perceptron (MLP), Naive Bayes (NB) and k-Nearest Neighbor (kNN) on each setup. The implementation utilized the Python scikit-learn library and the tests were run on a machine with Intel Xeon Platinum 8171M CPU and 64 GB RAM. The model performances are evaluated in terms of accuracy, precision, recall, F1 score and training time.

In each experiment we performed 10-fold cross-validation, hence, the test set contains 10% of the full dataset. Due to the imbalanced distribution of data samples across classes, we used *Stratified KFold* method for splitting the dataset into training and testing partitions, which ensures train and test sets maintain the same distribution as the entire dataset and prevents disproportionately low representation of classes with fewer instances.

Hyperparameter tuning for each model has been performed by randomized search, where we provided a wide search space with granular increments for each feature. The optimized parameter values are summarized in Table 4. Naive Bayes has only a few hyperparameters and the model is not sensitive to their values, hence, it is not listed in the table.

Table 4. Optimized Model Parameters

Model	Optimized Parameter Values
Random Forest	n estimators=200, max depth=None, min samples split=2, min samples leaf=1 max features= 'sqrt'
XGBoost	max depth=12, learning rate=0.6, subsample=1.0, colsample bytree=0.8, colsample bylevel=1.0
MLP	α =0.0001, hidden layer sizes=100 activation: 'relu', optimizer: 'sgd', learning rate: 'constant'
kNN	n neighbors=7, weights= 'uniform' leaf size=5 , p=1

5.1. Case A: Malware Detection Performance

We first compared the model performances in the full dataset comprising both normal and malware traffic.

Table 5 compares the accuracy, precision, recall, and F1-score for these models. For all models except Naive Bayes (NB), these metrics' values are nearly identical. NB trains extremely fast; however, it has high precision but low recall, *i.e.,* most of its predicted labels are correct, however it detects very few of the relevant labels. Training time is significantly high with XGBoost and MLP compared to other models. Random Forest offers the best tradeoff of accuracy and performance.

Table 5. Case A: Model Performance Results

Model	Accuracy	Precision	Recall	F1-score	Training
NB	0.483	0.964	0.483	0.632	8.1 s
kNN	0.978	0.977	0.978	0.977	406.6 s
RF	**0.982**	**0.982**	**0.982**	**0.981**	**546.0 s**
XGBoost	**0.984**	**0.984**	**0.984**	**0.983**	**2816.4 s**
MLP	0.979	0.978	0.979	0.978	1304.4 s

The performances for each class (label) in the dataset have been analyzed and summarized in Table 6. In the table, the prefix "I-" indicates these labels belong to intrusion types. The labels are displayed in descending order of the total number of samples in the dataset.

Table 6. Case A: Accuracy of Each Class Prediction

Traffic Type	NB	kNN	RF	XGBoost	MLP
Normal	0.456	**0.997**	**0.998**	**0.998**	**0.996**
I-Generic	0.970	0.980	0.985	**0.986**	0.983
I-Exploits	0.045	0.696	0.806	0.835	**0.840**
I-Fuzzers	0.057	0.544	0.656	**0.663**	0.552
I-DoS	0.001	0.391	0.242	0.317	0.173
I-Reconnaissance	0.000	**0.711**	**0.769**	**0.770**	**0.771**
I-Analysis	**0.674**	0.038	0.116	0.120	0.011
I-Backdoor	**0.197**	0.026	0.073	0.082	0.013
I-Shellcode	**0.993**	0.219	0.477	0.583	0.364
I-Worms	**0.889**	0.000	0.111	0.444	0.056

Except for Naive Bayes, all models predicted the Normal class with 99% success rate; this is primarily due to this class label covering 87.3% of the samples in the dataset. We observe a wide distribution of classification performance across different models for some attack labels. Generally, MLP performed poorly in classifying undersampled labels; and XGBoost and Random Forest outperformed Naive Bayes and kNN on all labels (except DoS). As all intrusion types except Generic and Exploits have less than 1% of the dataset, all models yielded significantly low performance in terms of classifying traffic samples into these labels. An exception is the case of Reconnaissance traffic traces, which has been relatively precisely classified despite the low sample count. We discuss these observations in Section 6.

5.2. Case B: Attack Identification Performance

The results in Section 5.1 demonstrate that *Normal* traffic patterns have been successfully recognized by most algorithms. To focus on the detection of the intrusion types, we prepared a smaller dataset comprising only intrusion traffic patterns (*i.e.,* we have excluded the Normal traffic from the dataset). These experiments tackle a more complex problem, the correct classification of intrusion types from the traffic patterns.

The model summary is presented in Table 7. Though we observe an approximately 9-10% decrease in all performance metrics compared to Table 5, XGBoost, RF and MLP continue to have good performance in terms of not only accuracy, which may be insufficient in imbalanced sets as this one, but also precision and recall.

Table 7. Case B: Model Performance Results

Model	Accur.	Prec.	Recall	F1-score	Training
NB	0.672	0.878	0.672	0.682	1 s.
kNN	0.868	0.874	0.868	0.870	7.82 s.
RF	**0.885**	0.889	0.885	0.880	47.8 s.
XGBoost	**0.902**	**0.906**	**0.902**	**0.897**	781.6 s.
MLP	**0.889**	0.892	0.889	0.885	698 s.

Table 8 presents the performance of each label. When compared to Table 6, the categories Generic and Exploits, respectively comprising 67% and 14% of the data in Case B, showed slight performance improvements. Notably, performance for Fuzzers (7.5% of the dataset) increased by approximately 31.4% across all models. Other class labels with smaller sample sizes (except Reconnaissance) also improved relative to Case A; however, their detection performance remains unsatisfactory. Among the models, XGBoost classification performance improved (*e.g.,* for Shellcode). The MLP model, which performed poorly in Case A, improved slightly. kNN outperformed others in detecting DoS attacks, though its performance was still relatively low. NB yielded high accuracies for the lowest-sampled two classes. These behaviors are also discussed under Section 6.

All models except NB continue to perform poorly in detecting malware types with few samples in the dataset. To investigate how increasing the sample size for these categories would affect performance, we proceed to the third set of experiments, Case C.

Table 8. Case B: Accuracy of Each Malware Prediction

Traffic Type	NB	kNN	RF	XGBst	MLP
I-Generic	0.968	0.979	0.982	**0.985**	0.981
I-Exploits	0.041	0.694	0.814	**0.866**	0.832
I-Fuzzers	0.066	0.795	**0.864**	**0.871**	**0.836**
I-DoS	0.004	0.401	0.204	0.300	0.303
I-Reconnaissance	0.000	**0.739**	**0.761**	**0.771**	**0.771**
I-Analysis	0.444	0.157	0.183	0.213	0.131
I-Backdoor	0.451	0.030	0.077	0.094	0.086
I-Shellcode	**0.987**	0.318	0.470	**0.729**	0.477
I-Worms	**0.765**	0.000	0.177	0.471	0.118

5.3. Case C: Attack Identification on Slightly Balanced Dataset

The dataset used in Case B was imbalanced, *i.e.,* had disproportionate numbers of samples from different malware types. In datasets with imbalanced class distributions, the minority class samples may be misclassified, as the training phase aims to minimize the loss function by correctly classifying the instances of the majority class. To alleviate the class imbalance issue, Synthetic Minority Over-Sampling Technique (SMOTE) was applied to synthetically generate data points belonging to the minority classes. The sample sizes after oversampling for all attack labels are shown in Table 9.

Table 9. Number of Records Before/After Oversampling

Attack Type	Before	After
Generic	193,933	193,933
Exploits	40,073	40,073
Fuzzers	21,821	21,821
DoS	14,718	14,718
Reconnaissance	12,588	12,588
Analysis	2,409	12,500
Backdoor	2,096	12,500
Shellcode	1,360	12,500
Worms	157	12,500

We observe that oversampling the minority classes did not cause significant performance improvement with any of the models compared to Case B and caused an increase in training time (Table 10).

Table 10. Case C: Model Performance Results

Model	Accur.	Prec.	Recall	F1-score	Training
NB	0.675	0.685	0.872	0.675	1.2
kNN	0.863	0.875	0.863	0.868	8.81
RF	**0.888**	0.890	0.888	0.885	67.1
XGBoost	**0.897**	**0.900**	**0.897**	**0.897**	1454.3
MLP	**0.874**	0.884	0.874	0.872	904.2

We then take a closer look at individual attack performances (Table 11). On particularly the lowest-sample label (Worms), the performance increased from 0% to ~65% with kNN and from ~12% to ~71% with MLP. DoS, Analysis and Backdoor attacks are not properly identified by any model. The performance with this case most clearly demonstrates the performance of detecting different malware types, in isolation from the impact of the number of samples. We discuss the challenges in distinguishing the attack types in Section 6.

Table 11. Case C: Accuracy of Each Malware Prediction

Traffic Type	NB	kNN	RF	XGBst	MLP
I-Generic	0.969	0.978	0.983	**0.985**	0.979
I-Exploits	0.045	0.677	0.832	**0.849**	0.808
I-Fuzzers	0.076	0.760	0.844	**0.855**	0.792
I-DoS	**0.014**	**0.378**	**0.195**	**0.300**	**0.118**
I-Reconnaissance	0.000	0.718	0.759	0.762	0.723
I-Analysis	0.485	0.224	0.220	0.209	0.224
I-Backdoor	0.464	0.060	0.129	0.155	0.305
I-Shellcode	0.987	0.636	**0.815**	**0.808**	**0.828**
I-Worms	0.765	0.647	0.647	0.471	0.706

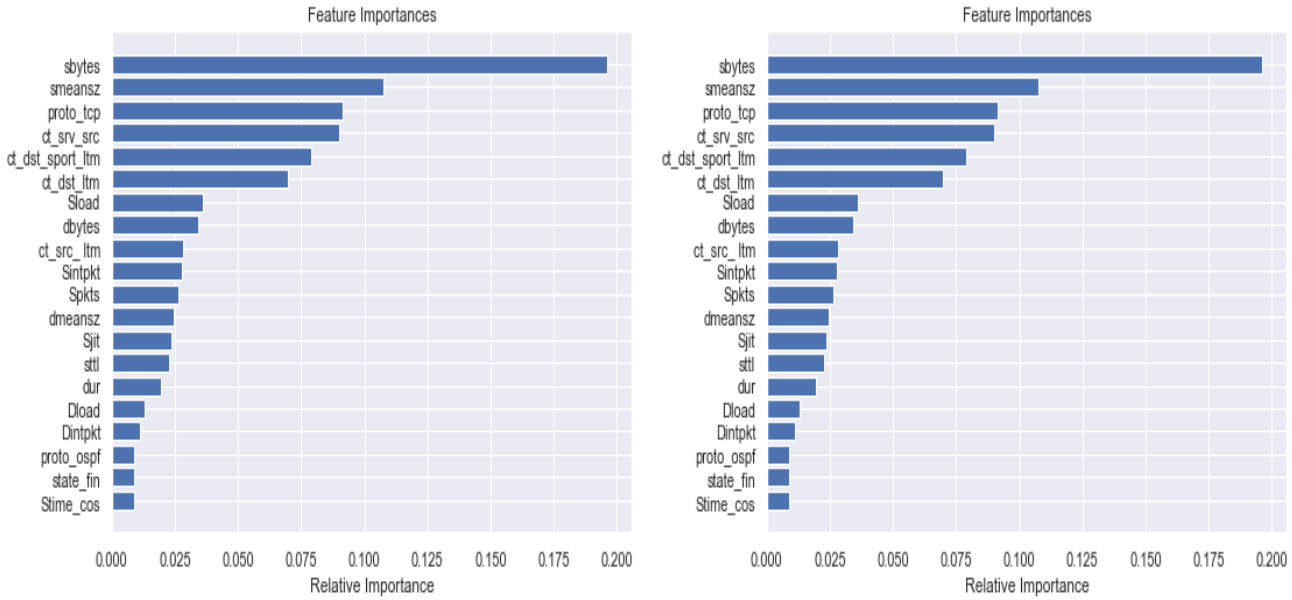
6. Discussion

We have several observations from our experiments that are unexpected and worthy of discussion. Specifically, we discuss: *(i)* Naive Bayes performance, *(ii)* DoS attack detection challenges, *(iii)* Reconnaissance attack being detected despite underrepresented, (iv) Analysis and Backdoor detection challenges, and (v) The impact of oversampling. In this section we discuss the potential causes of the observed behavior.

Naive Bayes (NB) has high precision and low recall in Case A and Case B, which means NB can make correct predictions, but not all true labels are detected. Intriguingly, NB has poor performance in classifying all Normal samples (which constitute ~87% of the dataset) in Case A, while it classifies the two lowest-sample classes, Shellcode and Worms, having respectively 0.06% and 0.007% of the dataset. The poor generalization of the Normal class may be due to its features not being well separated from the malicious traffic, as NB model's decision boundaries are sensitive to the feature distributions. Observing the confusion matrix for Case A (excluded here due to lack of space), we observe that Normal samples are confused with Analysis (5.6% of samples) and Worms traffic (45% of samples).

On the other hand, ***Naive Bayes outperformed all models in detecting underrepresented classes***. NB estimates the probability of each class given the input features. As NB focuses on the likelihood of individual features, it does not require a large number of examples to learn about each class and performs well for rare attack types. The two lowest-numbered labels, Shellcode and Worms, are best detected by NB-based classifier. This is a sharp contrast to other classifiers (particularly MLP and kNN) that require more data to perform well.

DoS attacks have not been detected properly by any of the models in all three cases. The confusion matrices indicate that RF, XGBoost and MLP confuse DoS with Exploits in significant proportions. Figure 2 demonstrates that the top features Random Forest and XGBoost focus on include the transport protocol, mean



(a) RF (b) XGBoost

Figure 2. Feature importances for Case B.

flow packet size from source, transferred number of bytes, and the number of connections with the same source address and service type. These features can be used to detect a large number of packets or connections with high frequency, which is a characteristic for both DoS and Exploits (like buffer overflow attacks). In practice, an Exploit may involve more targeted actions (than flooding for DoS) to compromise a system, but it can have similar traffic characteristics during the exploitation phase. Among the models, kNN performed best in detecting DoS attacks in all three cases (albeit still with low accuracy). kNN focuses on local distribution of data samples and thus can classify correctly when DoS attack samples are clustered together in certain regions of the feature space, even if samples might overlap with other classes in the global distribution.

Reconnaissance traffic seems to occupy a unique region in the feature space, making it easier to distinguish from other attack types. Reconnaissance attack detection accuracy is high with all models except for NB, in all three cases. Intriguingly, even in Case A where these attacks constitute less than 1% (precisely 0.55%) of the dataset, all models have a detection accuracy of greater than 71% for this malware. This behavior is attributed to the distinctive patterns in reconnaissance attacks, such as generating high frequency of connection attempts to multiple ports ("port scanning") and sequential scanning of IP addresses or ports ("network-wide probing"). These footprints, captured by features in the dataset such as destination ports, source to destination packet statistics, connection attempts (*synack*, *ct_dst_src_ltm*, *ct_srv_dst*) may render it easier for models to learn and classify Reconnaissance attacks correctly, even with fewer samples.

Analysis and Backdoor attacks were not correctly detected by any of the models on any of the datasets. The best performance of detecting Analysis attacks is by Naive Bayes classifier, whose performance drops when the Normal traffic is removed. Other models attain a maximum of ~20% accuracy on Analysis and 30% on Backdoor samples.

First, we look into why Analysis attacks cannot get detected despite oversampling. These attacks aim to gather information regarding the architecture, configuration or vulnerabilities of a system, without directly disrupting its operation; hence, they are typically passive or stealthy in nature. Many analysis attacks (e.g., passive sniffing) generate little to no detectable network activity, which renders them difficult to be captured from packet payloads or traffic patterns. Confusion matrices demonstrate that Analysis is confused with Exploits commonly by all models. Both attacks may involve sending multiple small packets, to test vulnerabilities (Exploits) or gather information (Analysis), or both may involve many connection attempts. In the flow-based characterization of attack types, models may not be able to distinguish these samples.

Backdoor samples in the UNSW-NB15 dataset contain low-frequency, periodic traffic that simulates the infected system's attempts to communicate with the C2 server. Similar to Analysis, Backdoor traffic is also confused with Exploits by all models except NB. Backdoor's low and periodic communication patterns resemble Exploit patterns that involve repeated subtle attempts to exploit a vulnerability (e.g., low-frequency probing).

We infer that ***Exploits, Analysis, and Backdoor attacks have similar network behaviors.*** The functionality of these attacks are different; Exploits interact with services and generate unusual traffic patterns, Analysis involves probing and scanning for vulnerabilities, Backdoors establish unauthorized remote access and infiltrate data. Despite differences in functionality, these attacks may share behaviors such as multiple connection attempts or short-lived connections; consequently, the ports, protocols, connection activity, or

packet sizes may be similar, which may be visible through features such as *sttl* (source TTL), *dur* (duration), *ct_dst_sport_ltm* (count of connections with same destination address and source port in 100 connections). Furthermore, since Exploits have a larger number of samples than Analysis or Backdoor (even in Case C), models may be biased toward this class to minimize overall error.

Oversampling only helps the lowest two classes but does not impact other labels positively. In perclass performances, we observe that all models but NB- have poor accuracy in detecting Shellcode and Worms attack types. Naive Bayes explicitly uses the probability of each label in the dataset and this allows the model to assign a meaningful base likelihood to the lowest-sample (*i.e.,* "rare") classes, which may be a reason why NB can perform well on under-represented classes. MLP and kNN perform worse on rare classes due to their reliance on data distribution; the former requires large amount of data to generalize well, and the latter depends on the distance between instances, which may be higher for the minority class, causing misclassifications. Upon oversampling, we observe the performances of these two models increase, MLP more significantly. The second lowest-sample class, Shellcodes, demonstrate more significant improvement in detection accuracy using all models.

7. Conclusion

In this study, we evaluated the performance of multiple machine learning (ML) models for classifying the nine attack types in the UNSW-NB15 dataset. We explored the behavior of these ML models across different proportions of class distributions, and discussed how model characteristics and attack behaviors interact in the context of intrusion detection.

Our results revealed that *(i)* certain attack types have specific feature distribution and are easily detected (e.g., Reconnaissance); *(ii)* some attack types demonstrate similar behavior and are confused with each other by multiple models (e.g., Exploits, Fuzzers, Analysis, Backdoor); and *(iii)* some attack types (e.g., DoS) exhibit low accuracy across all models due to their overlapping feature space and potentially less distinctive characteristics.

In terms of model performance, we observed that *(i)* Naive Bayes, despite its simplicity, was successful in detecting underrepresented classes due to its reliance on probabilistic priors; *(ii)* kNN outperformed even more complex models, in cases with local clusters and specific attack types; *(iii)* tree-based models like RF and XGBoost performed well overall but struggled with low-sample classes, demonstrating dependence on balanced data distributions; and *(iv)* Neural network-based approach (MLP) was able to capture complex patterns but required sufficient class representation to generalize effectively.

Our results highlight the importance of adapting the ML models based on the unique characteristics of network traffic and attack types, suggesting that robust intrusion detection systems may benefit from hybrid approaches that combine the strengths of different algorithms. As part of future work, we plan to incorporate model-agnostic interpretability methods such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations) to provide finer-grained insights into feature contributions across attack classes. This will complement our macro-level analysis by enabling a deeper understanding of how individual features influence model decisions. Such analysis could also help guide the design of more robust and targeted detection strategies.

Acknowledgement

This study was performed within the scope of graduate thesis (ID 765088) in Bahcesehir University [35].

Author contributions

Mehmet Erdi Özbek: Conceptualization, Methodology, Software, Tests, Visualization, Writing-Original draft.
Ece Gelal Soyak: Conceptualization, Methodology, Supervision, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

1. Alkashto, H., & Elewi, A. (2024). Integration of blockchain and machine learning for safe and efficient autonomous car systems: A survey. *Turkish Journal of Engineering*, *8*(2), 282-299
2. Ayas, M. Ş. (2021). A brief review on attack design and detection strategies for networked cyber-physical systems. *Turkish Journal of Engineering*, 5(1), 1-7.
3. Bace, R., & Mell, P. (2001). Intrusion detection systems, special publication, *National Institute of Standards and Technology (NIST)*, 16.
4. Moustafa, N., & Slay, J. (2015, November). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conf. (MilCIS)*(pp. 1-6). IEEE.
5. Basholli, F., Daberdini, A., & Basholli, A. (2023). Possibility of protection against unauthorized interference in telecommunication systems. *Engineering Applications*, 2(3), 265-278.
6. Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, *36*(1), 16-24.
7. Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1), 1-22.
8. Singh, A. P., Singh, M., Bhatia, K., Pathak, H. (2024). Encrypted malware detection methodology without decryption using deep

learning-based approaches. *Turkish Journal of Engineering*, 8(3), 498-509.

9. Singh, A. (2025). Real Time Intrusion Detection In Edge Computing Using Machine Learning Techniques. *Turkish Journal of Engineering*, *9*(2), 385-393.
10. Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, *25*(1-3), 18-31.
11. Shafiq, M., Tian, Z., Bashir, A. K., Du, X., & Guizani, M. (2020). CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine-learning techniques. *IEEE Internet of Things Journal*, 8(5), 3242-3254.
12. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (pp. 1-6). IEEE.
13. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1, 108-116.
14. Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, 100, 779-796.
15. Ferriyan, A., Thamrin, A. H., Takeda, K., & Murai, J. (2021). Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *Applied Sciences*, 11(17), 7868.
16. Guven, E. Y., Gulgun, S., Manav, C., Bakir, B., & Aydin, Z. G. (2022). Multiple classification of cyber attacks using machine learning. *Electrica*, vol. 22, no. 2, pp. 313–320, 2022.
17. Fernandes, R., & Lopes, N. (2022, June). Network intrusion detection packet classification with the HIKARI-2021 dataset: a study on ML algorithms. In *2022 10th International Symposium on Digital Forensics and Security (ISDFS)* (pp. 1-5). IEEE.
18. Vinayakumar, R., et al., (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7, 41525-41550.
19. Kilincer, I. F., Ertam, F., & Sengur, A. (2021). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks,* 188, 107840.
20. Zou, L., Wei, Y., Ma, L., & Leng, S. (2022, May). Feature-attended multi-flow LSTM for anomaly detection in internet of things. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 1-6). IEEE.
21. Kiflay, A. Z., Tsokanos, A., & Kirner, R. (2021, October). A network intrusion detection system using ensemble machine learning. In *2021 International Carnahan Conference on Security Technology (ICCST)* (pp. 1-6). IEEE.
22. Wester, P., Heiding, F., & Lagerström, R. (2021, October). Anomaly-based intrusion detection using tree augmented naive bayes. In *2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW)* (pp. 112-121).
23. Huma, Z. E., Latif, S., Ahmad, J., Idrees, Z., Ibrar, A., Zou, Z., ... & Baothman, F. (2021). A hybrid deep random neural network for cyberattack detection in the Industrial Internet of Things. *IEEE Access*, 9, 55595-55605.
24. Yang, S., Guo, H., & Moustafa, N. (2021, December). Hunter in the dark: Discover anomalous network activity using deep ensemble network. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)* (pp. 829-840). IEEE.
25. Abou El Houda, Z., Hafid, A. S., & Khoukhi, L. (2021, Dec.). A novel machine learning framework for advanced attack detection using SDN. In *2021 IEEE Global Comm. Conference (GLOBECOM)* (pp. 1-6).
26. Azad, S., Naqvi, S. S., Sabrina, F., Sohail, S., & Thakur, S. (2021, December). Iot cybersecurity: On the use of machine learning approaches for unbalanced datasets. In *IEEE Asia-Pacific Conf. on Computer Science and Data Engineering (CSDE)* (pp. 1-6). IEEE.
27. Arregoces, P., Vergara, J., Gutiérrez, S. A., & Botero, J. F. (2022, April). Network-based intrusion detection: A one-class classification approach. In *NOMS 2022-IEEE/IFIP Network Operations and Management Symposium* (pp. 1-6). IEEE.
28. Guizani, N., & Ghafoor, A. (2020). A network function virtualization system for detecting malware in large IoT based networks. *IEEE Journal on Selected Areas in Communications*, 38(6), 1218-1228.
29. Husain, A., Salem, A., Jim, C., & Dimitoglou, G. (2019, December). Development of an efficient network intrusion detection model using extreme gradient boosting (XGBoost) on the UNSW-NB15 dataset. In *2019 IEEE Int. Symposium on Signal Processing and Information Technology (ISSPIT)* (pp. 1-7). IEEE.
30. Al-Zewairi, M., Almajali, S., & Awajan, A. (2017, October). Experimental evaluation of a multi-layer feed-forward artificial neural network classifier for network intrusion detection system. In *2017 International Conference on New Trends in Computing Sciences (ICTCS)* (pp. 167-172). IEEE.

31. Ahmad, M., Riaz, Q., Zeeshan, M., Tahir, H., Haider, S. A., & Khan, M. S. (2021). Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set. *EURASIP Journal on Wireless Comm. and Networking*, 1-23.
32. Larriva-Novo, X., Villagrá, V. A., Vega-Barbas, M., Rivera, D., & Sanz Rodrigo, M. (2021). An IoT-focused intrusion detection system approach based on preprocessing characterization for cybersecurity datasets. *Sensors*, 21(2), 656.
33. Bagui, S., Kalaimannan, E., Bagui, S., Nandi, D., & Pinto, A. (2019). Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset. *Security and Privacy*, 2(6), e91.
34. Dainotti, A., Pescapé, A., & Ventre, G. (2007, June). Worm traffic analysis and characterization. In *2007 IEEE International Conference on Communications (ICC)* (pp. 1435-1442). IEEE.
35. M. Erdi Özbek (2022), "Malware analysis and identification with machine learning techniques", MSc Thesis, Bahcesehir University, Turkey.