Signal, Image and Video Processing (2025) 19:254
https://doi.org/10.1007/s11760-025-03813-5

ORIGINAL PAPER

Machine learning pipelines for IoT botnet detection and behavior characterization in heavily imbalanced settings

Djordje D. Jovanović[1,2] · **Pavle V. Vuletić**[2]

Received: 4 October 2024 / Revised: 13 December 2024 / Accepted: 3 January 2025

Abstract
This paper, presents a new methodology for IoT botnet detection based on network intra-flow parameter time series analysis and supervised machine learning classification. The study focuses on time series feature extraction and machine learning pipeline improvements and methods to solve the problem of heavily imbalanced datasets, characteristics of many information security use cases. Another side result is the inference of key distinguishing malware behavior features that make them detectable with large precision. The research is based on real-world IoT malware dynamic behavior analysis, The samples were collected over 4 years (2019–2023), presenting one of the most recent IoT malware datasets and a unique long-term malware behavior analysis. The analysis suggests the type and rate of changes in IoT botnet malware behavior and some invariant features that can be used to reliably detect even previously unseen malware samples (so-called zero-day cases). Presented experimental results prove that the synthetic sample generation methodologies used in this study do not overfit the classifiers, but can detect zero-day malware samples with 0.9706 accuracy and 0.9041 f1 score.

Keywords IoT botnet · Imbalanced datasets · Zero-day detection

1 Introduction

Botnets represent a collection of devices infected with malware, controlled by a malicious administrator (botmaster), aiming to execute various attacks on computer infrastructure, such as performing Distributed Denial of Service (DDoS) attacks, spreading ransomware, stealing personal information, unwanted digital currency mining, and other attacks [1]. Recently, botnets have become an increasing threat, especially since Internet of Things (IoT) devices, often lacking security measures of their own, and whose numbers have significantly grown in recent years, have become targets. Probably the best known is the Mirai botnet which at one moment consisted of more than 600.000 devices and performed devastating attacks on ISP infrastructures [2]. Most existing approaches to botnet attack detection focus on identifying attacks (usually DDoS) postmortem, i.e. after they occur using typically vast datasets obtained through full packet captures on a given link. Given the destructive power of newer botnets and that devices can be part of a botnet for days or months before being used in an attack, detecting the botnet as early as possible to prevent the attack and neutralize the botnet is of great importance, and this is the aim of our work.

Our previous research [3] offered a successful alternative approach to botnet detection in comparison to the methods described in the research literature: early detection of infected devices by observing botnet Command and Control (CnC) network flows as time series and extracting intra-flow statistical features from them, leveraging Software Defined Networks. The goal of this feature extraction method is to save computational resources required for detection, while at the same time maintaining high detection accuracy. The extraction of features and the application of machine learning classifiers achieved the goal of detecting botnet communication before an attack occurs while maintaining the detection accuracy as in similar proposed systems that use full traffic analysis, requiring up to two orders of magnitude less processing power and storage space, leading to the design of

✉ Djordje D. Jovanović
dj.jovanovic@turing.mi.sanu.ac.rs

Pavle V. Vuletić
pavle.vuletic@etf.bg.ac.rs

[1] Mathematical Institute of the Serbian Academy of Sciences and Arts, Kneza Mihaila 36, Belgrade 11000, Serbia

[2] University of Belgrade, School of Electrical Engineering, Bulevar Kralja Aleksandra 73, Belgrade 11000, Serbia

Published online: 28 January 2025

a machine learning-based system for botnet communication detection.

In this study, we move one step forward, toward more accurate results in botnet detection. We present a new IoT botnet behavior analysis and detection methodology that reduces the problem to one of time series analysis. Also, the focus is primarily on the machine learning pipeline improvements and methods to solve the problem of heavily imbalanced datasets, characteristics of many information security use cases, and the inference of the distinguishing malware behavior features. The research is based on real-world static IoT malware code and dynamic behavior analyses collected over four years (2019–2023), presenting the most recent malware samples and such long-term analysis for the first time. This extensive IoT malware behavior analysis through time series analysis suggested the type and rate of changes in IoT botnet malware behavior and some invariant features that can be used to reliably detect even previously unseen malware samples (so-called zero-day cases).

Key contributions of this paper include the extended analysis of the IoT malware behavior over four years, the analysis of techniques for feature selection from time series, hyperparameter optimization, model selection, artificial sample generation, and the extraction of the set of features that are invariant for IoT malware for long periods. The suggested machine learning pipeline approaches were proven not to be overfitted to the specific datasets, being able even to detect zero-day samples with high scores. This analysis yielded far better botnet detection results, with an F1 score of 0.9041 for pipelines that did not use artificial sample generation and an F1 score of 0.9984 for pipelines that did use artificial sample generation, significantly surpassing the detection accuracy of other studies and methodologies in the field.

The paper is organized as follows: Sect. 2 overviews the related work in the field of botnet detection and machine-learning-based detection methods. Section 3 briefly overviews the PI-BODE system for time series and statistical parameter extraction from live network flows. Section 4 describes used machine learning pipelines. Section 5 describes the experimental setup, results and discussion while Sect. 6 concludes the paper.

2 Literature overview

Botnets have garnered industry and scientific communities' attention because of their potential to execute large-scale attacks. The two most researched topics in the field of botnets are: (1) detection of attacks and attack patterns (with DDoS attacks being the most commonly studied), which encompasses the majority of botnet research, and (2) detection of botnet infrastructure and bot behavior. The first group of studies can detect botnets after the attack and is typically based on coarse-grained, but economic network flow analysis because the attacks are often seen as sharp increases in overall flow statistics (e.g. total number of bytes or packets). The second group of studies aims to detect more subtle features of botnet behavior, through full packet captures. This overview focuses more on this second group of research efforts as it is much closer to our research goals.

Botmasters must maintain a communication channel with infected devices to send the appropriate commands and be aware of the controlled devices. This communication channel is called command and control (CnC) communication. One of the earliest studies on CnC focused on IRC communication [4], used in the first botnets, extracting 16 features from packet captures. These features represent summary flow statistics, such as the total number of packets and bytes, packet size histograms, byte variance, and inter-arrival times for each flow. On the other hand, a more recent study offered a richer data model for network traffic [5], providing 29 recorded and 14 generated network traffic features related to flow group specifics, along with three categorical fields.

Some authors have investigated botnet communication traffic features for CnC channels in HTTP-based botnets [6, 7]. Wang et al. [8] designed a system called BotMark, which combines statistical and graph-based network traffic features with a hybrid analysis using similarity, stability, and anomaly scores. These scores are used as input for an ensemble method aimed at classifying network traffic. One particularly interesting feature of this system is its detection method, which relies on observed flow similarity (bots exhibit similar communication patterns with the botmaster) as the basis for identifying botnet communication.

Kusak et al. [9] developed a system for detecting ransomware-type viruses based on machine learning and feature extraction using software-defined networks (SDNs). The network features in this study include inter-arrival times, the ratio of incoming to outgoing packets, and packet burst lengths, which exhibit specific values in the case of virus communication. Another study proposing SDNs for detecting threats in the Internet of Medical Things (IoMT) is the work by Liakat et al. [10]. The authors used SDNs for packet data collection and decision-making on packet forwarding, which can be employed to halt the spread of botnets. Their detection mechanism uses convolutional neural networks (CNNs) and the Cuda Deep Neural Network Long Short-Term Memory (cuDNNLSTM) model. CNN was used for feature selection, while the cuDNNLSTM network was used for classification.

Koryonitis et al. [5] worked on detecting various attacks on IoT infrastructure, including scanning attacks, DoS, and data theft. The work by De La Torre Parra et al. [11] provides an overview of IoT botnets and detection methods, using LSTM neural networks to detect attacks. The attack set in this study includes various types of network flooding, and scanning as a preparatory phase for attacks and spam distribution. Kurni-

abudi et al. [12] analyzed feature selection using information gain as a criterion and developed several machine-learning classification models. Their findings showed that model prediction depends on the number of selected features and that the optimal number of features varies from model to model. Mhmood et al. [13] used an innovative approach, combining game theory, population-based optimization algorithms, and deep learning attacks on smart grid infrastructure detection. Geetha et al. [14] demonstrated botnet detection in IoT environments using adaptive weighted SVM as a classifier, and multi-objective optimization.

A characteristic problem of data analysis and anomaly detection in information security is a class imbalance of datasets. This is a consequence of the fact that, in most cases, the attack phases to the computer infrastructure (except volumetric DDoS attacks), represent a minority compared to the total benign traffic or other data collected on devices [15]. Such is the case of botnet CnC flows, where the infected device emits several small packets per minute, making this class a significantly smaller part of the total number of network flows of a network, whose volume is often measured in the order of magnitude of several Gbit/s. A series of papers have dealt with solving the essential problem of datasets containing malware traffic, which is a class imbalance, using artificial sampling techniques [16–21]. This is because malware comprises only a small fraction of network traffic. One of the characteristic techniques is the various versions of the SMOTE (Synthetic Minority Oversampling Technique) algorithm, which are presented in the remainder of the paper. This technique is applicable in the case of supervised learning, as labels are required to detect which minority class needs to be oversampled. In papers [22, 23], modifications of the SMOTE algorithm were presented and applied to the malware detection problem. By creating synthetic samples, an improvement in model performance and overall generalization capability is expected, as demonstrated in works [24, 25], which focus on creating synthetic samples in the field of malware detection. Bojarajulu and Tanwar [21] used a customizable convolutional neural network (CCNN) and an Enhanced SMOTE approach for botnet detection in IoT.

Our paper differs from the previous research in several key elements: (1) it is not based on full packet captures, but on a unique method of converting network flows into time series of packet and byte count parameters, providing the same or better detection accuracy while keeping the required computing resources up to two orders of magnitude lower compared to the full packet capture approaches, (2) it provides a more detailed analysis and application of class imbalance mitigation methodology than the previous research (3) it shows invariant features of IoT botnet that can be leveraged for zero-day botnet detection (4) the results are verified on the data gathered from the long-term analysis of real IoT botnet samples which is presented in this paper.

3 IoT botnet command and control communication detection

As said in the introduction, we have developed and previously reported on PI-BODE [3], a system for IoT botnet detection that uses the detection of bot CnC communication through the analyses of per network flow time series. Network flows are defined as a tuple (Source IP address, Destination IP address, Source Port, Destination Port, Transport protocol) that presents a single communication transaction. PI-BODE system periodically samples packet and byte counters for each flow, and calculates the difference between the results of the two consecutive samples forming per-flow time series for packet and byte counts as shown in Fig. 1. A sampling rate of 1 s which was used in these experiments proved to provide accurate and reliable results for IoT botnet detection, although with changing behavior patterns this parameter can be changed.

These time series are then used to calculate various per-flow statistical parameters that are fed into the machine-learning pipeline to detect the botnet command and control flows and this way infected computers. In the previous research, we used 14 parameters extracted from the time series and were able to obtain the same accuracy in bot detection as the other reported research which used full packet captures. On the other side, using and storing only per-flow data series instead of full packet captures reduced the processing and storage overhead between 28 and 280 times, depending on the sampling rate.

PI-BODE design and later evaluation were based on the dataset formed from our original research of the Command and Control communication patterns [26]. That dataset consists of sixteen Mirai, eighteen Gafgyt and one NanoCore sample and is publicly available [27]. In addition, we have used the IoT23 dataset provided by the Stratosphere laboratory of Czech Technical University in Prague, also known for their CTU-13 dataset.

However, we have continued gathering IoT malware until 2023 and added 14 new Mirai-based samples. As expected, malware designers tend to change malware behavior to avoid detection that is based on signatures or some specific parameter. We have observed some new communication patterns like using variable TCP ports for malware upload to the infected device, varied number of packets, hiding the communication by Transport Layer Security (TLS), one-way heartbeat connections, and new CnC heartbeat periods (15 s, 20 s and 48 s, unlike typical 60-s interval observed in the first samples). Figure 2 shows the command and control communication patterns of some of the newer malware samples. It can be seen that some samples (e.g. assailant_arm6) use Transport Layer Security (TLS) to hide the message content and to look similar to the regular HTTPS traffic. Other samples vary other parameters like the set of TCP flags used in the packets (e.g.

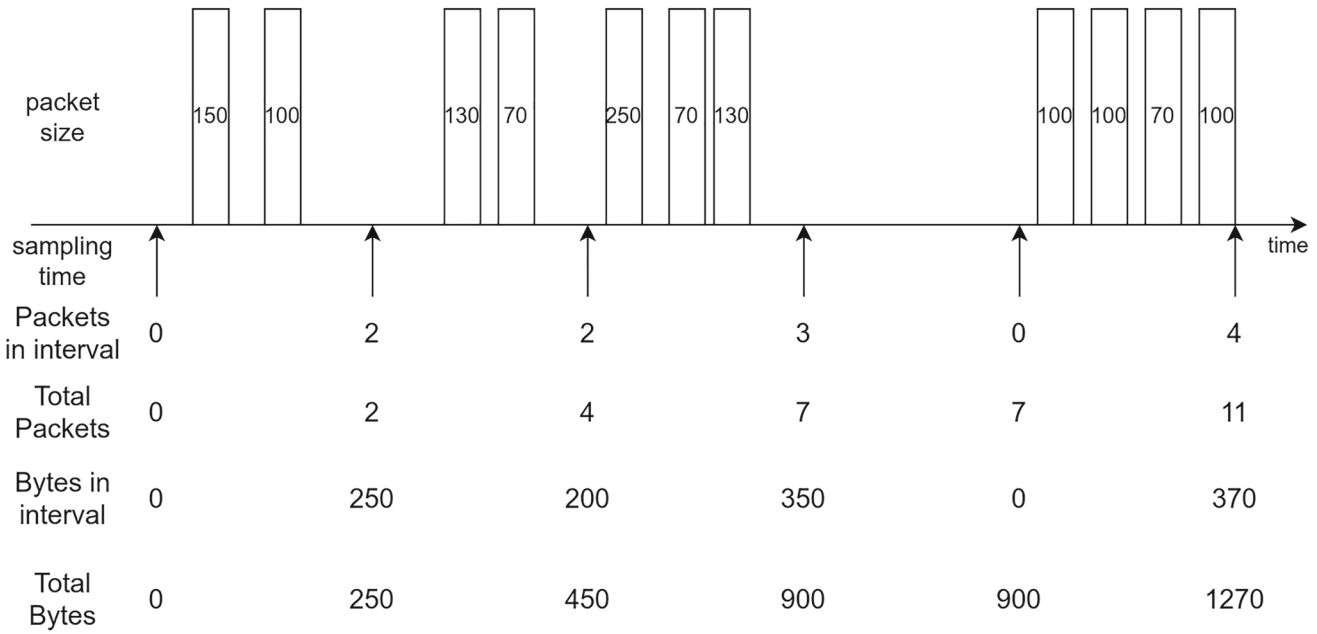

Fig. 1 Sampling flow statistics: a visualization on how packet size in bytes and number of packets is collected from a time series

m7_nn and assailant_arm6 use duplicate acknowledgments to look like regular retransmissions, while armv6l_2_1 does not) or message values to avoid signature-based detection present in common intrusion detection systems. One of the research hypotheses in this phase is that it will be possible to detect newer malware samples using the classifiers trained on the old ones meaning the potential to detect the zero-day botnet malware. We further tested the impact of machine learning pipeline decisions and artificial sample generation methods on overfitting the classifier and report these results in the subsequent sections.

4 Machine learning pipeline

The machine learning pipeline used in this study consists of the following steps: feature extraction, artificial sample generation, feature selection, hyperparameter search, and classification. The artificial sample generation is an optional step and is used in only a selection of experiments, as explained further, to analyze its effect on this type of imbalanced dataset.

4.1 Feature extraction

In the feature extraction phase, we used an enriched set of features compared to the previous studies, obtained using the tsfresh library [28]. The library extracts 76 statistical parameters from packet and byte count time series defined in 3. Using these parameters in combination with the statistical features yields more features that can be extracted and used in the dataset. In addition to the features offered by this library, we integrated extracting autocorrelation values for each delay possible for a particular time series. The total number of extracted features used in this dataset is 170. Features that contained missing values were removed. No normalization was performed on the data, because normalization removes information on features whose maximum value is not bounded. This is significant in cyber security use cases as new malware samples can have feature values that are outside of the training dataset value bounds.

For some features, the tsfresh library defines extraction parameters. For each value or combination of parameter values, a new feature is created in the dataset. Some of the extracted features have extraction parameters, which define either input parameters for statistical tests or algorithms (e.g. number of bins for the Lempel-Ziv complexity) or which output should be extracted as a result of a statistical test or algorithm (e.g. aggregation type for the FFT algorithm). Table 1 lists such features, their parameters, as well as the parameter values used in the experiments.

4.2 Synthetic sample generation

As already remarked, a characteristic problem in data analysis and anomaly detection in information security is dataset imbalance. A common methodology to mitigate this issue is synthetic sample generation. This study examines the impact of three synthetic sample generation methods on classification results: Borderline SMOTE [29], SMOTE ENN

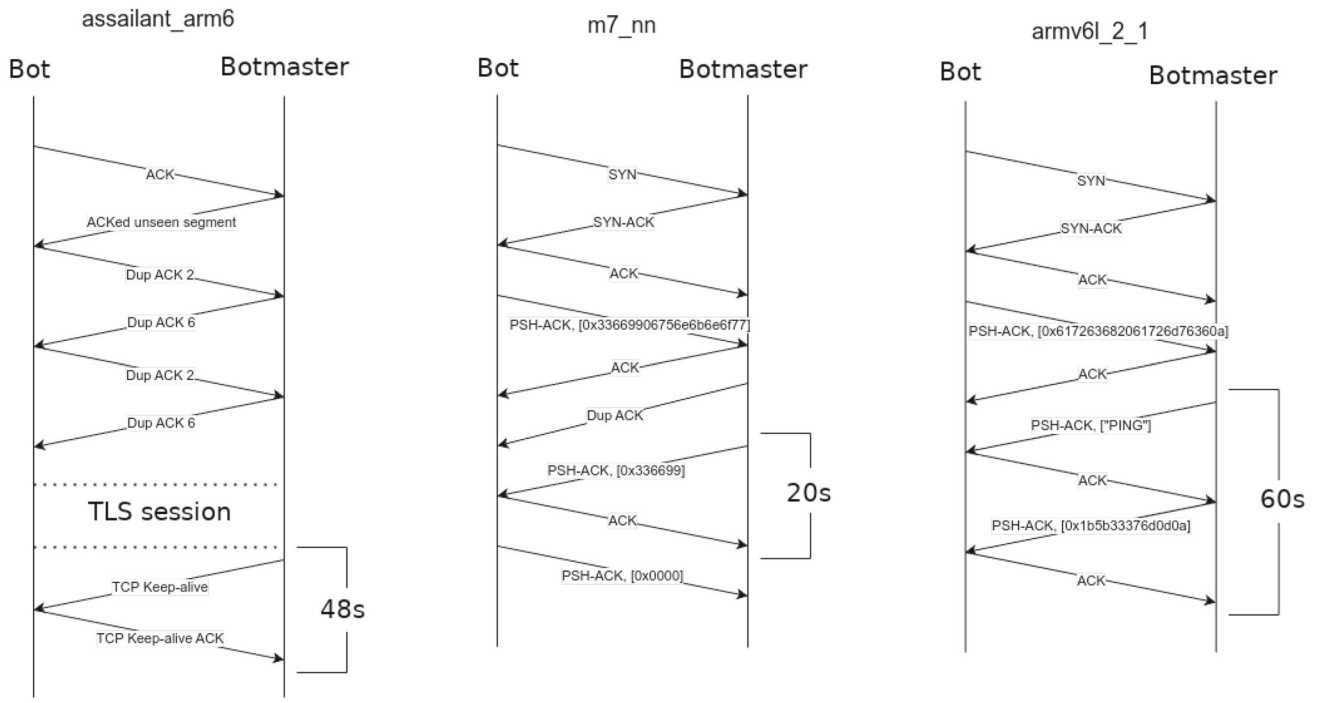

Fig. 2 Botnet communication diagrams for collected botnet samples

Table 1 List of parameters used for extracting parametrized features

Feature	Parameter	Parameter value
augmented_dickey_fuller	attr	teststat, pvalue, usedlag
	autolag	AIC, BIC, t-stat, None
cid	normalize	True, False
fft_aggregated	aggtype	centroid, variance, skew, kurtosis
fourier_entropy	bins	2–10
index_mass_quantile	q	0.05−0.95, step 0.05
large_standard_deviation	r	0.05−0.95, step 0.05
lempel_ziv_complexity	bins	2–10
linear_trend	attr	pvalue, rvalue, intercept, slope
mean_n_absolute_max	number_of_maxima	2–5
quantile	q	0.05−0.95, step 0.05
ratio_beyond_r_sigma	r	0.25–2, step 0.25
symmetry_looking	r	0.05–1, step 0.05

[30, 31], and ADASYN [32], which are implemented in the imbalanced-learn library [33]. This step in the workflow is carried out immediately after feature extraction. The main advantage of this step is increasing detection scores, yet it does not have good performance in presence of noisy data.

4.2.1 SMOTE ENN

SMOTE [30] is a popular algorithm used to address class imbalance in datasets to improve the accuracy of machine learning models. Its main principle is adding minority class samples by generating synthetic examples, rather than simply repeating existing ones, which is the case when an oversampling technique is performed on a dataset. By creating synthetic examples strategically positioned near the decision boundary, SMOTE enhances the model's ability to accurately classify minority class instances. However, despite its benefits, there are some drawbacks to the SMOTE algorithm. One significant limitation is its sensitivity to noise in the data, which can potentially lead to the creation of synthetic samples that misrepresent the target distribution of the minority class. Since SMOTE generates synthetic samples based

on existing minority class instances and their nearest neighbors, samples with statistical errors in the minority class can greatly influence the generation of synthetic examples. This can result in synthetic samples that poorly represent the desired distribution of the minority class. In order to improve SMOTE results, an edited nearest neighbor step was introduced (ENN for short), where artificially generated samples near the decision boundary are removed. This modification of SMOTE algorithm is called SMOTE ENN [30, 31].

4.2.2 Borderline SMOTE

In response to the limitations of the traditional SMOTE method, the Borderline SMOTE algorithm [29] offers an extension by focusing on borderline instances of the minority class. These are instances near the decision boundary, which makes them crucial for the learning process. The first step in the Borderline SMOTE algorithm involves identifying minority class instances located close to the decision boundary. These samples have statistically small distances relative to the boundary that separates the minority from the majority class in the feature space. The borderline instances are typically more challenging to classify correctly, and the algorithm places special attention on them. Similar to the traditional SMOTE method, the Borderline SMOTE algorithm proceeds by generating synthetic instances based on the selected borderline instances. The goal of this approach is to reduce the potential impact of noisy data and ensure that the synthetic samples accurately capture the key characteristics of the minority class. One of the key advantages of Borderline SMOTE is its ability to adapt to complex data distributions within the minority class. By focusing on borderline instances, the algorithm aims to capture the complexities of the data distribution more precisely, thus addressing the limitations of traditional SMOTE when working with complex data structures.

4.2.3 ADASYN

Adaptive Synthetic Sampling (ADASYN) [32] is an extension of the SMOTE algorithm, designed to address some of its limitations. The key principle behind ADASYN is the adaptive generation of synthetic samples for the minority class based on the distribution of existing samples, with a focus on areas where the minority class instances are densely distributed. This approach allows ADASYN to better handle complex data distributions by concentrating on the most critical areas where the class imbalance is most pronounced. By doing so, ADASYN improves the model's ability to learn and accurately classify the minority class, particularly in more challenging regions of the feature space.

4.3 Feature selection

Feature selection involves the choice of the most relevant features for building a predictive model. This process improves model performance by reducing overfitting, simplifying the model, and speeding up the training process. By selecting the most important features, the model's interpretability can be enhanced, and better generalization to new data can be achieved. In the field of botned detection, papers [34, 35] proposed different methods for feature selection. In [34], a comparison of filter-based feature selection methods was performed on the KDD 99 CUP botnet dataset, while [35] proposed an adaptation of the Cuttlefish algorithm for feature selection problem on the same dataset. In our experiments, we decided to take a different approach to feature selection, by using wrapper-based methods. In this paper, the following feature selection methods will be examined: Recursive Feature Elimination (RFE) [36], Recursive Feature Addition (RFA) [37], and Boruta [38, 39].

4.3.1 Recursive feature selection algorithms

Recursive Feature Elimination and Recursive Feature Addition are feature selection techniques designed to identify the most important features of a given machine learning model. This iterative approach works by recursively removing or adding the least relevant feature(s) until a predefined number of features is reached. Although these two methods are powerful, they have some limitations. For example, it may not perform well in the presence of highly correlated features or when the signal-to-noise ratio is low. In such cases, alternative feature selection methods or data preprocessing and transformation techniques may be needed.

4.3.2 Boruta algorithm

The Boruta algorithm is a feature selection method designed to identify all relevant features in a dataset. It is useful when working with high-dimensional datasets with many potential features. Boruta is based on the random forest algorithm and can handle various data types, including continuous, categorical, and mixed variables. The core idea behind Boruta is the creation of "shadow features," which are random permutations of the original columns in the dataset. By comparing the importance of the original features to their corresponding "shadows," Boruta determines the significance of each feature in predicting the target variable. If a feature's importance is statistically significantly higher than its shadow counterparts, it is considered relevant and retained for the model. Otherwise, it is deemed irrelevant and removed. This process is repeated iteratively until all relevant features are identified. Boruta aims to capture all relevant features, including complex interactions and nonlinear relationships, enhancing the

overall model's performance. Additionally, it can be used with any model and adapts to a mixture of continuous and categorical variables, eliminating the need for data preprocessing to convert types.

4.4 Hyperparameter selection

Hyperparameter selection refers to the process of optimizing a model's hyperparameter values to improve its performance in machine learning. In this study, the shap-hypertune package [40] was used to implement this step. This process involves defining the hyperparameter space as a set of possible values for each hyperparameter, followed by selecting an algorithm that searches for the combination of hyperparameter values that maximizes the chosen classification metric. For the experiments in this research, the Tree-structured Parzen Estimator (TPE) algorithm [41, 42] was used, which estimates the density distribution of the data and provides a basis for selecting optimal hyperparameters. This algorithm is implemented in the hyperopt library [43], which enables efficient hyperparameter search within the defined space.

4.5 Sample classification

For prediction, we used a machine learning model called gradient boosting. Gradient boosting [44–46] is a machine learning technique that has gained popularity due to its ability to perform classification with high accuracy, especially for tabular numeric data. It works by building a series of decision trees, where each tree learns from the mistakes of the previous one. Optimizing a specific target function by adding the weak decision trees (where "weak" means the classification ability is suboptimal), the iterative approach involves training new models on errors produced by previous models, gradually minimizing the overall error and improving prediction accuracy. Gradient boosting showed greater precision than deep learning methods for malware and intrusion detection and tabular datasets in recent studies [47, 48]. Two implementations of extreme gradient boosting were tested: LightGBM [49] and XGBoost [50–52]. The shap-hypertune package [40] was utilized to implement these two machine-learning models.

5 Experimental setting and results

All experiments were conducted on the ETF-IoTB dataset [27], supplemented with new samples of the same virus classes from 2022 and 2023, described in Sect. 3. The network conversations of this dataset were labeled manually. The dataset consists of 111 malicious and 4474 normal network conversations.

5.1 Pipeline parameter settings

This section outlines the parameters used for the feature selection step and hyperparameter spaces for hyperparameter selection. The first step of all three selected feature selection algorithms involves model selection and performance measure selection. The model used for these experiments was LightGBM, more accurate as it will be shown in Sect. 4.2, while the performance measure used was accuracy. Detailed definitions of all parameters are described in the shap-hypertune library. Table 2 presents the parameter values used for the feature selection algorithms, while Table 3 shows the hyperparameter search spaces considered for the XGBoost and LightGBM models.

5.2 Sample generation strategy experiments

In the first set of experiments, we tested 15 machine learning pipelines using different sample generation and feature selection techniques and two classification models, as shown in Table 4. The experiments were repeated 100 times for each machine learning process, each time with different samples in training and testing datasets, and the average of all scores was taken. The train-test set ratio was 80/20. The first group of experiments (pipelines 1–6), which did not include artificial sample generation, compared the performance of the XGBoost and LightGBM models. The results presented in Table 4 showed that LightGBM achieved the best results in this group of experiments, and was thus used in the second group of experiments (pipelines 7–15) to compare the impact of various synthetic sample generation techniques.

In Table 4, the experimental results reveal that workflows incorporating artificial sample generation exhibit higher scores. This finding is consistent with studies using similar methodologies [24, 25]. However, unlike those studies, this work did not utilize feature transformation. These results suggest that collecting more botnet malware samples would improve the accuracy of machine learning-based detection systems. For further discussion and experimentation, workflows with and without the artificial sample generation step will be considered separately, given their structural differences.

The results show that Workflow 2 (RFA and LightGBM) achieved the highest scores among workflows without artificial sample generation, while Workflow 13 (SMOTEENN, RFA, and LightGBM) scored highest among those with artificial sample generation. These two workflows will be used for further experiments.

5.3 Malware behavior inference

Since each workflow in the previous set of experiments was repeated 100 times, with different training and test sets for

Table 2 Parameters used for feature selection

Algorithm	Parameter	Value
RFA, RFE	n_iter	100
	sampling_seed	0
	importance_type	shap_importances
	step	0.02
Boruta	n_iter	100
	sampling_seed	0
	importance_type	shap_importances
	early_stopping_boruta_rounds	1000
	perc	10

Table 3 Hyperparameter search spaces used for hyperparameter selection

Algorithm	Hyperparameter	Value
XGBoost	booster	gbdt, dart, rf
	n_estimators	Random uniform variable in range [100,200]
	learning_rate	Random logarithmic variable in range [0.01, 0.2]
LightGBM	booster	gbdt, dart, rf
	n_estimators	Random uniform variable in range [100,200]
	learning_rate	Random logarithmic variable in range [0.01, 0.2]
	bagging_fraction	Random uniform variable in range [0.05,0.95]
	num_iterations	Random uniform variable in range [100,200]
	bagging_freq	Random uniform variable in range [2,10]

each repetition, it was observed that the set of features that influenced the classification the most was not identical. For both workflows, with and without the artificial sample generation step, data were extracted on which features influenced the classification result and their importance across the experiments. The number of features that had an impact in at least one experiment was 153 for workflows with artificial sample generation, compared to 141 for workflows without it. All features were ranked by importance, i.e., by how many experiments they influenced the classification. The results of feature ranking showed that the following features had the most significant impact on classification:

linear_trend__attr___intercept—This feature indicates that linear regression can approximate the time series. The fact that this feature is one of the most important suggests that malicious and benign traffic flows frequently differ in this characteristic. Since this feature points to the existence of trends in the time series, it can be inferred that one class of flows exhibits this property while the other does not. Considering that botnet control and heartbeat flows have constant traffic, while standard TCP sessions, which account for most benign traffic, exhibit an increasing flow due to TCP slow-start and window scaling mechanisms, the importance of this feature becomes clear.

large_standard_deviation—Large standard deviation suggests that botnet traffic has a characteristic standard deviation in relation to the difference between the minimum and maximum values in the time series. For normal traffic, the percentage of flows where the standard deviation exceeds a certain percentage of the difference between the minimum and maximum values is over 90%, while for botnet traffic, it is at most 54%. This is expected, given that most of the observed and analyzed botnet traffic time series consist of repeating values, resulting in a low standard deviation.

autocorrelation_stats__mean—Indicates that the mean value of the autocorrelation for all lag values differs between botnet and benign traffic. This is also expected, given that botnet traffic consists of repetitive and periodic time series with a well-defined lag equal to the period of packet transmission.

Quantile—Reflects the variability and range of the observed variable, in this case, the number of bytes in time slots. Based on the parameters obtained from the experiments, it was observed that for normal traffic, the average quantile values within time slots range between 20,000 and 38,000 bytes, while for botnet traffic, this value ranges between 100 and 750 bytes. The difference is due to the relatively constant packet sizes in botnet control and heartbeat communication, while packet sizes in other network flows vary (within the technology's limits of 64–1500 bytes).

mean_n_absolute_max—Shows that the average of the first few maximum values in the time series significantly dif-

Table 4 List of workflows with listed steps and results

Number	Pipeline			Botnet			Benign		
	Artificial sample generation	Feature selection	Model	Accuracy	Recall	f1-score	Accuracy	Recall	F1-score
1		RFA	XGBoost	0.9068	0.7536	0.817	0.9934	0.9977	0.9956
2		RFA	LightGBM	0.9098	0.775	0.8274	0.994	0.9977	0.9959
3		RFE	XGBoost	0.9059	0.72	0.7933	0.9925	0.9978	0.9952
4		RFE	LightBGM	0.8909	0.7233	0.7902	0.9926	0.9974	0.995
5		Boruta	XGBoost	0.8606	0.695	0.7607	0.9919	0.9967	0.9943
6		Boruta	LightGBM	0.8777	0.67	0.7476	0.9912	0.9972	0.9942
7	ADASYN	RFA	LightGBM	0.9969	0.9971	0.997	0.9971	0.9969	0.997
8	ADASYN	RFE	LightGBM	0.9926	0.9953	0.994	0.9953	0.9926	0.9939
9	ADASYN	Boruta	LightGBM	0.9936	0.996	0.9948	0.996	0.9936	0.9948
10	Borderline SMOTE	RFA	LightGBM	0.997	0.9959	0.9964	0.9959	0.997	0.9964
11	Borderline SMOTE	RFE	LightGBM	0.9935	0.9944	0.994	0.9944	0.9935	0.994
12	Borderline SMOTE	Boruta	LightGBM	0.9945	0.9951	0.9948	0.9952	0.9945	0.9948
13	SMOTE ENN	RFA	LightGBM	0.9991	0.9992	0.9992	0.9992	0.9991	0.9991
14	SMOTE ENN	RFE	LightGBM	0.9978	0.9965	0.9971	0.9964	0.9977	0.9971
15	SMOTE ENN	Boruta	LightGBM	0.9986	0.9981	0.9984	0.9981	0.9986	0.9984

fers between botnet and benign flows. For normal traffic, the average value of this feature ranges between 25,000 and 36,000 bytes, while for botnet traffic, it ranges from 500 to 1000. Similar to the previous parameter, this result is a consequence of the fact that the observed botnet traffic time series mainly consisted of identical (and relatively small) packets, causing the average of the maximum values to fall within a specific range for botnet traffic.

fft_aggregated__aggtype___variance—Suggests that the variance of the Discrete Fourier Transform (DFT) coefficients can help detect botnet traffic. This is also expected, given that botnet traffic consists of periodic time series, whose Fourier transform will have characteristic peaks at packet transmission intervals.

Since botnet communication via command and control channels is periodic and repetitive, the time series of such communication is expected to exhibit low variance, lack linear trends, and consist of repeated values at specific intervals, resulting in a characteristic frequency spectrum. The fact that the above features assess these properties and are among the most influential features in botnet traffic detection confirms the hypothesis that periodicity, repetition, and lack of trends in botnet traffic time series can be used to detect botnet communication through command and control channels.

5.4 Experiments on possibilities of zero-day attack detection

Since Workflows 2 and 13 produced the best results for the botnet dataset in terms of F1-score, these two workflows were chosen for the next group of experiments, where training strategies with both old and new botnet samples were analyzed, as well as the ability to detect new botnet variants by training on old samples.

Experiment Groups:

- P1: In the first group of experiments, samples from the original ETF-IoTB dataset (2019–2021) [27] were used for training, while newer samples (2022–2023) were used for testing.
- P2: In the second group, both the original dataset and half of the newer samples were used for training, while the other half of the newer samples were used for testing.
- P3: In the third group, both training and testing datasets consisted of all available samples, both old and new.

The best-performing workflows from previous experiments (one from those with artificial sample generation and one from those without) were tested in these experiments. Table 5 presents the results of these experiments.

These results, as expected, show that the larger the proportion of newer malware samples in the training set, the lower the percentage of false negatives, meaning the system's overall detection accuracy is higher. However, the more intriguing observation is that tests in which the system was trained solely on older samples and then used to detect new botnet malware (P1 experiment group), with scores of 0.9706 and 0.9871, demonstrate that it is possible to detect new, previously unseen botnet malware with high precision which is a case of zero-day attack detection. This also indicates that, despite the significant time gap between the release of the new and old samples (four years), the observed characteristics of network behavior remained sufficiently similar to detect botnet communication with considerable success. The use of artificial sample generation methods has a positive impact on increasing botnet traffic detection accuracy when the model is trained with all samples (P2 and P3 groups). In the P1 group, artificial sample generation (workflow 13) produced worse prediction results in the majority class, compared to the case when not used (WF 2). It can be argued that artificial sample generation contributes to overfitting the model to the old sample dataset. This shows a trade-off between detecting zero-day attacks more accurately and the overall classification precision. A recent study [21] represents state-of-the-art research in botnet detection using neural networks, where a CNN was trained to classify different types of IoT botnet attacks. The paper [21] used three different datasets to demonstrate the efficiency of botnet attack classification, two of which used PCAP files that contained recorded traffic from live networks, and the remaining dataset used traffic data from IoT devices. The scores presented in our study cannot be directly compared to those presented in [21] due to a difference in datasets, classification problems, and extracted scores. However, the results presented in this study show higher detection accuracy than the results in [21], especially when trained with all samples, indicating the potential of the presented approach in detecting botnet CnC channels before the attack begins, and with a significantly smaller dataset. Finally, by showing classifier performance on different types of dataset partitions, the results demonstrate that zero-day attacks could be detected with significant precision.

6 Conclusion

This paper presents the research on IoT botnet detection through intra-flow time series statistics analysis and machine learning pipeline customization. It gives the guidelines for malware behavior data processing that achieves the best detection results. Two main conclusions can be drawn from showcased experiments. The proposed synthetic sample generation method increases system accuracy, while not overfitting the model, and enabling even zero-day botnet detection. The results of experiments with different training set partitions show that it is possible to detect new malware on a system trained on older data, confirming the hypothesis

Table 5 Classification scores for different dataset partitioning

EG[1]	WF[2]	B-ACC[3]	B-REC[4]	B-F1[5]	N-ACC[6]	N-REC[7]	N-F1[8]
P1	2	0.9706	0.8462	0.9041	0.9869	0.9978	0.9923
P1	13	0.9871	0.8669	0.9231	0.8807	0.9886	0.9315
P2	2	0.8929	0.8971	0.8913	0.9948	0.9945	0.9946
P2	13	1.0	0.9302	0.9638	0.934	1.0	0.9659
P3	2	0.9098	0.775	0.8274	0.994	0.9977	0.9959
P3	13	0.9991	0.9992	0.9992	0.9992	0.9991	0.9991

[1]Experiment group
[2]Workflow
[3]Botnet Accuracy
[4]Botnet Recall
[5]Botnet F1-Score
[6]Benign Accuracy
[7]Benign Recall
[8]Benign F1-Score

of this research. This also revealed that during the four years of botnet sample collection, some communication patterns between the bot and botmaster via the CnC channel have not fundamentally changed their form and can be leveraged for botnet detection. However, the problem of detecting malicious behavior in information technology is highly complex. Once principles for detecting malicious behavior are found and studied, they may motivate the attackers to find ways to hide their software and infected devices. It is expected that if this research demonstrates that certain features, such as variance, standard deviation, periodicity measures, or the presence of linear trends, distinguish between command and control communications of malicious and benign traffic, attackers may attempt to hide CnC channels by modifying malware behavior to mimic benign traffic and further blend their communication into it. This introduces a new class of attacks, known as adversarial attacks, where attackers analyze the capabilities and behavior of the detector and modify the malware behavior accordingly. The next phases of research will focus on analyzing new trends in IoT malware behavior and investigating the possibility of modifying botnet behavior through variations in parameters used for detection, to find new, more robust detection mechanisms resistant to some types of adversarial attacks. Also, additional analyses, including the evaluation of feature extraction time depending on the length of the time series, the processing time of traffic flows in a real network environment, and the stabilization time of the most important features depending on the length of the time series, are needed to enable real-time botnet detection, not the analysis of network flow data after the attack is completed.

Acknowledgements This research was partially financially supported by the Ministry of Science, Technological Development, and Innovation of the Republic of Serbia (Contract No. 451-03-68/2024-03/200103).

References

1. Vormayr, G., Zseby, T., Fabini, J.: Botnet communication patterns. IEEE Commun. Surv. Tutor. **19**(4), 2768–2796 (2017)
2. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., et al.: Understanding the Mirai Botnet. In: 26th USENIX Security Symposium (USENIX Security 17), pp. 1093–1110. USENIX Association, Vancouver (2017). https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis
3. Jovanovic, D.D., Vuletic, P.V.: PI-BODE: programmable intraflow-based IoT botnet detection system. Comput. Sci. Inf. Syst. **21**(1), 37–56 (2024). https://doi.org/10.2298/CSIS211116064J
4. Livadas, C., Walsh, R., Lapsley, D., Strayer, W.T.: Usilng machine learning techniques to identify botnet traffic. In: Proceedings. 2006 31st IEEE Conference on local computer networks, pp. 967–974. IEEE (2006)
5. Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B.: Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Gener. Comput. Syst. **100**, 779–796 (2019). https://doi.org/10.1016/j.future.2019.05.041
6. Lee, J.S., Jeong, H., Park, J.H., Kim, M., Noh, B.N.: The activity analysis of malicious HTTP-based botnets using degree of periodic repeatability. In: 2008 International Conference on Security Technology, pp. 83–86 (2008)
7. Eslahi, M., Rohmad, M.S., Nilsaz, H., Naseri, M.V., Tahir, N.M., Hashim, H.: Periodicity classification of HTTP traffic to detect HTTP Botnets. In: 2015 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), pp. 119–123 (2015)
8. Wang, W., Shang, Y., He, Y., Li, Y., Liu, J.: BotMark: automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. Inf. Sci. **09**, 511 (2019). https://doi.org/10.1016/j.ins.2019.09.024
9. Cusack, G., Michel, O., Keller, E.: Machine learning-based detection of ransomware using SDN. In: Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization. SDN-NFV Sec'18, pp. 1-6. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3180465.3180467
10. Liaqat, S., Akhunzada, A., Shaikh, F.S., Giannetsos, A., Jan, M.A.: SDN orchestration to combat evolving cyber threats in Internet of Medical Things (IoMT). Comput. Commun. **160**, 697–705 (2020). https://doi.org/10.1016/j.comcom.2020.07.006

11. De La Torre, Parra G., Rad, P., Choo, K.K.R., Beebe, N.: Detecting Internet of Things attacks using distributed deep learning. J. Netw. Comput. Appl. **163**, 102662 (2020). https://doi.org/10.1016/j.jnca.2020.102662
12. Stiawan, D., Bin Idris, M.Y., Bamhdi, A.M., Budiarto, R.: CICIDS-2017 dataset feature analysis with information gain for anomaly detection. IEEE Access **8**, 132911–132921 (2020). https://doi.org/10.1109/ACCESS.2020.3009843
13. Mhmood, A., Ergul, O., Rahebi, J.: Detection of cyber attacks on smart grids using improved VGG19 deep neural network architecture and aquila optimizer algorithm. Signal Image Video Process. **18**, 1477–1491 (2024). https://doi.org/10.21203/rs.3.rs-3217829/v1
14. Geetha, C., Johnson, S., Oliver, A., Lekha, D.: Adaptive weighted kernel support vector machine-based circle search approach for intrusion detection in IoT environments. SIViP **04**(18), 1–12 (2024). https://doi.org/10.1007/s11760-024-03088-2
15. Milosevic, M.S., Ciric, V.M.: Extreme minority class detection in imbalanced data for network intrusion. Comput. Secur. **123**, 102940 (2022). https://doi.org/10.1016/j.cose.2022.102940
16. Al, S., Dener, M.: STL-HDL: a new hybrid network intrusion detection system for imbalanced dataset on big data environment. Comput. Secur. **110**, 102435 (2021). https://doi.org/10.1016/j.cose.2021.102435
17. Masoudi-Sobhanzadeh, Y., Emami-Moghaddam, S.: A real-time IoT-based botnet detection method using a novel two-step feature selection technique and the support vector machine classifier. Comput. Netw. **217**, 109365 (2022). https://doi.org/10.1016/j.comnet.2022.109365
18. Aborujilah, A., Nassr, R., Al-Othmani, A., Ali, N., Awang Long, Z., Husen, M.N., et al.: SMOTE-based framework for IoT Botnet attack detection. Adv. Cyber Secur. (2021). https://doi.org/10.1007/978-981-33-6835-4_19
19. Kumar, R., Malik, A., Ranga, V.: An intellectual intrusion detection system using Hybrid Hunger Games Search and Remora Optimization Algorithm for IoT wireless networks. Knowl. Based Syst. **256**, 109762 (2022). https://doi.org/10.1016/j.knosys.2022.109762
20. Rust-Nguyen, N., Sharma, S., Stamp, M.: Darknet traffic classification and adversarial attacks using machine learning. Comput. Secur. **127**, 103098 (2023)
21. Bojarajulu, B., Tanwar, S.: Customized convolutional neural network model for IoT botnet attack detection. SIViP **06**(18), 1–13 (2024). https://doi.org/10.1007/s11760-024-03248-4
22. Rustam, F., Jurcut, A.D.: Malicious traffic detection in multi-environment networks using novel S-DATE and PSO-D-SEM approaches. Comput. Secur. **136**, 103564 (2024)
23. Qing, Y., Liu, X., Du, Y.: Mitigating data imbalance to improve the generalizability in IoT DDoS detection tasks. J. Supercomput. **80**(7), 9935–9960 (2024)
24. Alfrhan, A.A., Alhusain, R.H., Khan, R.U.: SMOTE: class imbalance problem in intrusion detection system. In: 2020 International Conference on Computing and Information Technology (ICCIT-1441), pp. 1–5. IEEE (2020)
25. Gonzalez-Cuautle, D., Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, L.K., Portillo-Portillo, J., Olivares-Mercado, J., et al.: Synthetic minority oversampling technique for optimizing classification tasks in botnet and intrusion-detection-system datasets. Appl. Sci. **10**(3), 794 (2020)
26. Jovanovic, D.D., Vuletic, P.V.: Analysis and characterization of IoT malware command and control communication. In: 27th Telecommunications Forum TELFOR. IEEE (2019). https://ieeexplore.ieee.org/abstract/document/8971194
27. Jovanovic, G., Vuletic, P.: ETF IoT Botnet Dataset. https://doi.org/10.17632/nbs66kvx6n.1
28. Christ, M., Braun, N., Neuffer, J., Kempa-Liehr, A.W.: Time series feature extraction on basis of scalable hypothesis tests tsfresh: a python package. Neurocomputing **307**, 72–77 (2017). https://doi.org/10.1016/j.neucom.2018.03.067
29. Han, H., Wang, W.Y., Mao, B.H.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: International Conference on Intelligent Computing, pp. 878–887. Springer (2005)
30. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
31. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Syst. Man Cybern. **3**, 408–421 (1972)
32. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). IEEE 2008, pp. 1322–1328 (2008)
33. Lemaître, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J. Mach. Learn. Res. **18**(17), 1–5 (2017)
34. Taheri, R., Ahmadzadeh, M.: Studying the effect of discretization of data on accuracy of predicting Naïve Bayes algorithm, case study KDD99 CUP. J. Curr. Res. Sci. **S**, 457–462 (2016)
35. Taheri, R., Ahmadzadeh, M., Kharazmi, M.: A new approach for feature selection in intrusion detection system. Cumhuriyet Dent. J. **01**(36), 1344–1357 (2015)
36. Chen, X., Jeong, J.C.: Enhanced recursive feature elimination. In: Sixth International Conference on Machine Learning and Applications (ICMLA 2007), pp. 429–435. IEEE (2007)
37. Hamed, T. Recursive Feature Addition: A Novel Feature Selection Technique, Including a Proof of Concept in Network Security. Ph.D. Dissertation, The University of Guelph, Guelph, ON, Canada, 2017
38. Kursa, M.B., Jankowski, A., Rudnicki, W.R.: Boruta: a system for feature selection. Fund. Inform. **101**(4), 271–285 (2010)
39. Kursa, M.B., Rudnicki, W.R.: Feature selection with the Boruta package. J. Stat. Softw. **36**, 1–13 (2010)
40. cerlymarco.: Shap-hypertune: a python package for simultaneous Hyperparameters Tuning and Features Selection for Gradient Boosting Models. Figshare https://github.com/cerlymarco/shap-hypetune
41. Watanabe, S.: Tree-structured parzen estimator: understanding its algorithm components and their roles for better empirical performance. arXiv preprint arXiv:2304.11127 (2023)
42. Nguyen, D.A., Kong, J., Wang, H., Menzel, S., Sendhoff, B., Kononova, A.V.: Improved automated cash optimization with tree Parzen estimators for class imbalance problems. In: IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–9. IEEE 2021 (2021)
43. Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: International Conference on Machine Learning, pp. 115–123. PMLR (2013)
44. Natekin, A., Knoll, A.: Gradient boosting machines, a tutorial. Front. Neurorobot. **7**, 21 (2013)
45. Sheridan, R.P., Wang, W.M., Liaw, A., Ma, J., Gifford, E.M.: Extreme gradient boosting as a method for quantitative structure–activity relationships. J. Chem. Inf. Model. **56**(12), 2353–2360 (2016)
46. Friedman, J.H.: Stochastic gradient boosting. Comput. Stat. Data Anal. **38**(4), 367–378 (2002)
47. Lichy, A., Bader, O., Dubin, R., Dvir, A., Hajaj, C.: When a RF beats a CNN and GRU, together—a comparison of deep learning and classical machine learning approaches for encrypted malware traffic classification. Int. J. Inf. Secur. (2022). https://doi.org/10.48550/arXiv.2206.08004

48. Ziza, K., Tadic, P., Vuletic, P.: DNS exfiltration detection in the presence of adversarial attacks and modified exfiltrator behaviour. Int. J. Inf. Secur. **22**, 1865–1880 (2023). https://doi.org/10.1007/s10207-023-00723-w
49. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: LightGBM: a highly efficient gradient boosting decision tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), 3149–3157. Curran Associates Inc, Red Hook, NY, USA (2017)
50. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
51. Bentéjac, C., Csörgő, A., Martínez-Muñoz, G.: A comparative analysis of gradient boosting algorithms. Artif. Intell. Rev. **54**, 1937–1967 (2021)
52. Sagi, O., Rokach, L.: Approximating XGBoost with an interpretable decision tree. Inf. Sci. **572**, 522–542 (2021)