Leevy *et al. Journal of Big Data* (2022) 9:6
https://doi.org/10.1186/s40537-021-00558-z

Journal of Big Data

RESEARCH Open Access

IoT information theft prediction using ensemble feature selection

Joffrey L. Leevy*, John Hancock, Taghi M. Khoshgoftaar and Jared M. Peterson

*Correspondence:
jleevy2017@fau.edu
Florida Atlantic University,
777 Glades Road, Boca
Raton 33431, FL, USA

Abstract

The recent years have seen a proliferation of Internet of Things (IoT) devices and an associated security risk from an increasing volume of malicious traffic worldwide. For this reason, datasets such as Bot-IoT were created to train machine learning classifiers to identify attack traffic in IoT networks. In this study, we build predictive models with Bot-IoT to detect attacks represented by dataset instances from the Information Theft category, as well as dataset instances from the data exfiltration and keylogging subcategories. Our contribution is centered on the evaluation of ensemble feature selection techniques (FSTs) on classification performance for these specific attack instances. A group or ensemble of FSTs will often perform better than the best individual technique. The classifiers that we use are a diverse set of four ensemble learners (Light GBM, CatBoost, XGBoost, and random forest (RF)) and four non-ensemble learners (logistic regression (LR), decision tree (DT), Naive Bayes (NB), and a multi-layer perceptron (MLP)). The metrics used for evaluating classification performance are area under the receiver operating characteristic curve (AUC) and Area Under the precision-recall curve (AUPRC). For the most part, we determined that our ensemble FSTs do not affect classification performance but are beneficial because feature reduction eases computational burden and provides insight through improved data visualization.

Keywords: Bot-IoT, IoT, Ensemble feature selection, Information theft, Machine learning

Introduction

The IoT is a network of physical objects with limited computing capability [1]. In recent years, there has been rapid growth in the use of these smart devices, as well as an increasing security risk from malicious network traffic. Several datasets have been created for the purpose of training machine learning classifiers to identify attack traffic. One of the more recent datasets for network intrusion detection is Bot-IoT [2].

The Bot-IoT dataset contains instances of various attack categories: denial-of-service (DoS), distributed denial-of-service (DDoS), reconnaissance, and information theft. The processed full dataset was generated by the Argus network security tool [3] and is available from an online repository of several comma-separated values (CSV) files. Bot-IoT has 29 features and 73,370,443 instances. Table 1 shows the categories and subcategories of IoT network traffic for the full dataset.

Table 1 Bot-IoT: full set

Category	Subcategory	No. of instances
Normal	Normal	9543
DoS	TCP	12,315,997
	UDP	20,659,491
	HTTP	29,706
DDoS	TCP	19,547,603
	UDP	18,965,106
	HTTP	19,771
Reconnaissance	OS fingerprinting	358,275
	Service scanning	1,463,364
Information theft	Keylogging	1469
	Data exfiltration	118

In this work, we identify normal and attack type classes of Bot-IoT. The attack types we identify are from the information theft category and from its data exfiltration and keylogging subcategories. Hence, we evaluate three datasets, one composed of normal and information theft attack instances, another composed of normal and data exfiltration attack instances, and a third composed of normal and keylogging attack instances. The keylogging [4] subcategory refers to the interception of information sent from input devices, while the data exfiltration [5] subcategory broadly refers to the unauthorized transfer of data from a computer.

Each of the three datasets has 9543 instances labeled as Normal traffic. In addition, the Information Theft dataset has 1587 instances labeled as the information theft attack type, the data exfiltration dataset has 118 instances labeled as the data exfiltration attack type, and the keylogging dataset has 1469 instances labeled as the keylogging attack type. Based on the minority-to-majority class ratios (i.e., ratios of attack-to-normal instances), all three datasets have a class imbalance. To address the class imbalance, we use the random undersampling (RUS) [6] technique.

In all experiments, we employ the following eight learners: CatBoost [7], Light GBM [8], XGBoost [9], RF [10], DT [11], LR [12], NB [13], and a MLP [14]. To gauge the performance of these classifiers, the AUC and AUPRC metrics are used. Also, during training for some experiments, we apply hyperparameter tuning.

The crux of this study involves the comparison of results from experimentation with and without ensemble feature selection for the information theft, data exfiltration, and keylogging attack types. Feature selection reduces computational burden (i.e., speeds up training time) and provides data clarity (i.e., facilitates visual detection of patterns). In some cases, feature selection can improve classification performance and mitigate the effects of class imbalance [6]. Ensemble FSTs incorporate multiple feature ranking techniques to produce a combined method that is more efficient than its individual components [15]. Our ensemble FSTs are derived from both supervised and filter-based feature ranking techniques [16].

As far as we are aware, this is the first paper that exclusively evaluates the effect of ensemble feature selection on the Bot-IoT information theft category and subcategories. Furthermore, our use of eight different classifiers boosts the validity of our results.

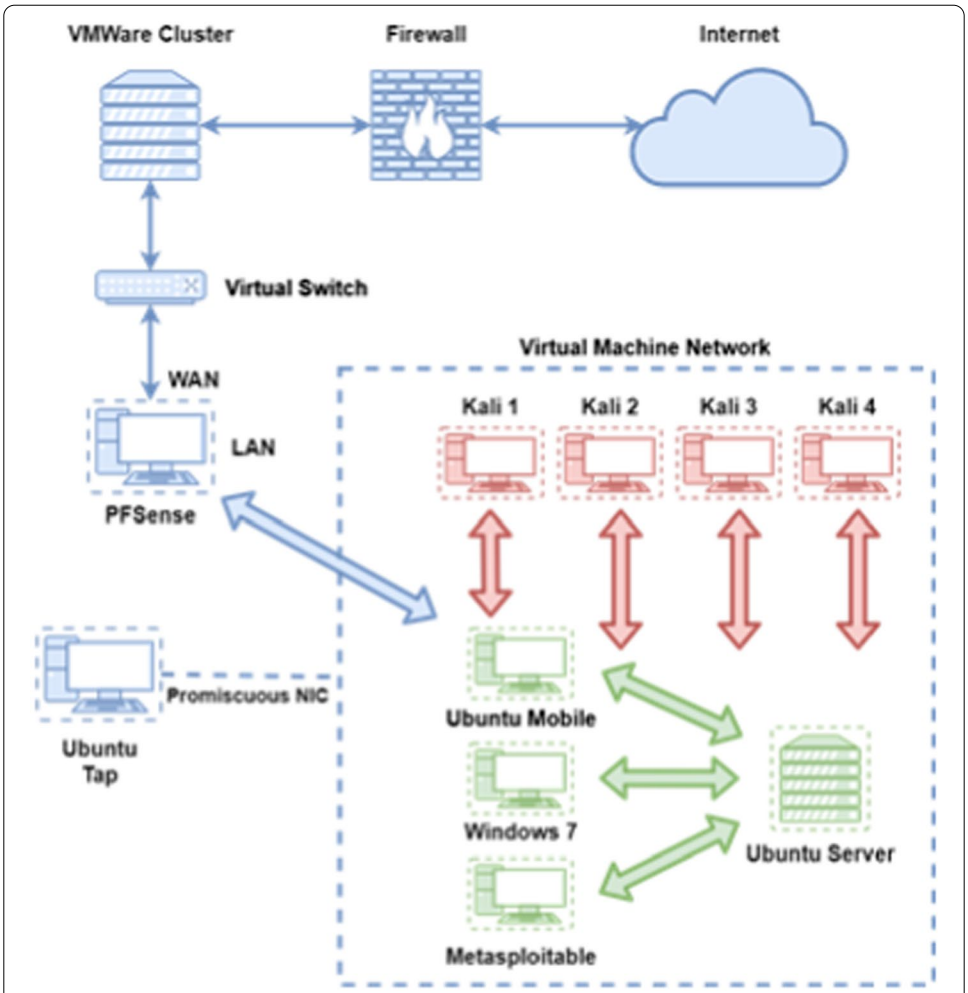

Fig. 1 Network diagram

The remainder of this paper is organized as follows: "Bot-Iot developmental environment" section describes the developmental environment and tools used to create Bot-IoT; "Related works" section discusses related Bot-IoT literature; "Methodology" section discusses preprocessing and classification tasks; "Results and discussion" section presents and analyzes our results; and "Conclusion" section concludes with the main points of this paper. In addition, Appendix A provides a list of features and their definitions, Appendix B contains tables of selected features, and Appendix C contains tables of tuned parameter values.

Bot-Iot developmental environment

Created in 2018 by the University of New South Wales (UNSW), Bot-IoT was designed to be a realistic representation of botnet attacks on IoT devices. The dataset was developed in a virtualized environment, with ten virtual machines hosted by an ESXi [17] hypervisor that accessed the Internet through a packet-filtering firewall. Fig 1 shows a schematic of the network within which Bot-IoT was created.

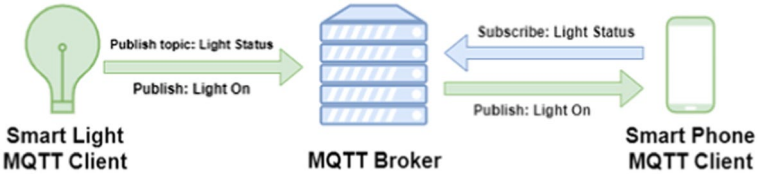

Fig. 2 MQTT example

The virtual machines were divided into three groups: IoT, Bots, and Supporting. Normal network traffic was generated by the Ostinato [18] tool, and IoT traffic was generated by the Node-Red [19] tool, along with several scripts run on a group of the virtual machines. Node-Red was used to simulate five IoT scenarios: a weather station, a smart refrigerator, motion activated lights, a smart garage door, and a smart thermostat. Attack traffic was generated using a variety of tools from the Kali Linux [20] suite, an open-source Linux distribution popular with penetration testers and others in the information security industry.

The IoT group of virtual machines comprised four machines running different operating systems. An Ubuntu server [21], which hosted some basic network services as well as the Message Queuing Telemetry Transport (MQTT) [22] protocol broker, was one of the most essential IoT virtual machines. In IoT networks, the MQTT protocol is used to share data between IoT devices and other clients using a publish/subscribe approach. It is utilized in a range of industries and provides a lightweight communications solution that uses less bandwidth. Mosquito MQTT [23] is the MQTT broker used in this testbed. Fig 2 provides an illustrative example of the MQTT broker involved in a data sharing task.

The other three IoT virtual machines ran Ubuntu Mobile [24], Windows 7, and Metasploitable [25], among other operating systems. Ubuntu Mobile is a Linux-based mobile phone operating system that is based on the Ubuntu desktop operating system. Windows 7 is a deprecated version of the Microsoft Windows operating system. Rapid7 created and published Metasploitable, a Linux-based virtual machine with many built-in vulnerabilities. The flaws were purposely built into Metasploitable to allow it to be utilized in penetration testing laboratories. These three virtual machines used Node-Red to act as IoT clients and devices, communicating with the Ubuntu server using the MQTT protocol. These four machines combined served as the attack surface for the Bots group of virtual machines.

The Bots group was made up of four virtual machines intended to simulate a botnet, which is a collection of machines controlled and used by a criminal actor. The Kali Linux operating system was installed on each of the four machines. They were equipped with a large toolkit that was used to conduct various assaults against the IoT group. The Theft attacks were carried out with the help of exploits and tools from the Metasploit penetration testing framework [26].

The Supporting group assisted in the generation and collection of data within the testbed. These machines consisted of an Ubuntu tap and a pfSense [27] firewall. The Ubuntu tap virtual machine sat on the same virtual network as the IoT and Bots groups. It utilized a conventional Ubuntu operating system, but was configured with a promiscuous

network interface controller (NIC) to operate in promiscuous mode. The automatic filtering of network traffic not intended for the host is disabled when the NIC is set to promiscuous mode. This allows packet analyzers like tcpdump [28] to record all traffic passing through the NIC. All communication to and from any of the IoT or Bots group of virtual machines was captured by the Ubuntu tap virtual machine. The pfSense firewall, which is the other virtual machine in the Supporting group, had both a local area network (LAN) interface and a wide area network (WAN) interface. The pfSense device served as the primary gateway out of the virtual network.

Related works

In this section, we highlight works associated with detecting malicious traffic in Bot-IoT. To the best of our knowledge, none of the related works have exclusively focused on detecting instances of information theft with ensemble FSTs.

Koroniotis et al. [29] proposed a network forensics model that authenticates network data flows and uses a deep neural network (DNN) [30] based on particle swarm optimization (PSO) [31] to identify traffic anomalies. The authors used the 5% Bot-IoT dataset, which they split in an 80:20 ratio for training and testing. The logistic cost function [32] was utilized, as it has been shown to be efficient at separating normal from attack traffic. The cost function is defined by the following equation [29]:

$$-\frac{1}{m}\sum_{i=1}^{m}(y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)) \tag{1}$$

To treat class imbalance, weights for normal traffic w_0 and attack traffic w_1 were incorporated into this cost function. This modified equation is defined as follows: equation [29]:

$$-\frac{1}{m}\sum_{i=1}^{m}(w_1 y_i log(\hat{y}_i) + w_0(1-y_i) log(1-\hat{y}_i)) \tag{2}$$

The best scores obtained by the model for accuracy, precision, recall, and F-measure were 99.90%, 100%, 99.90%, and 99.90%, respectively. Model performance was evaluated against the reported performance of models from other studies. The authors state that their model outperformed these other models (NB, MLP, association rule mining (ARM) [33], DT, support vector machine (SVM) [34], recurrent neural network (RNN) [35], and long short-term memory (LSTM) [36]). Evaluating model performance from one study against reported model performance from a non-identical study is problematic, since there may be too many factors of variation in the external study to account for.

Using a convolutional neural network (CNN) [37] and CUDA deep neural network LSTM (cuDNNLSTM)[1] hybrid, Liaqat et al. [38] set out to prove that their proposed model could outperform a Deep Neural Network-Gated Recurrent Unit (DNN-GRU) [39] hybrid, as well as a long short-term memory-gated re-current unit (LSTM-GRU) [39] hybrid. The dataset sample contained 477 normal instances and 668,522 attack instances from Bot-IoT. During data preprocessing, the data was normalized, feature extraction was performed, and to address class imbalance, the number of normal

[1] https://developer.nvidia.com/cudnn.

instances was up-sampled to 2,400. The hybrid CNN-cuDNNLSTM model was shown to be the best performer with top scores of 99.99%, 99.83%, 99.33%, and 99.33% for accuracy, precision, recall and F-measure, respectively. We point out that the use of up-sampling techniques can sometimes result in overfitted models [40]. For this study, the authors have not provided adequate information on their up-sampling technique and the model-building process in order for us to rule out the occurrence of overfitting. Therefore, we believe that their proposed model should also be evaluated on out-of-sample data to reaffirm their results.

Mulyanto et al. [41] proposed a cost-sensitive neural network based on focal loss [42]. The cross-entropy loss function [43], which is widely used in neural network classification models, is integrated with focal loss to reduce the influence of the majority class(es) for binary and multi-class classification. A CNN and a DNN served as the neural network classifiers for this approach. The networks were trained on the NSL-KDD [44], UNSW-NB15, and Bot-IoT datasets. The Bot-IoT dataset sample contained about 3,000,000 instances. For binary classification, the cost-sensitive neural networks based on focal loss outperformed neural networks where the synthetic minority oversampling technique (SMOTE) [45] technique was applied and also outperformed plain neural networks. For Bot-IoT, top scores were obtained for the DNN cost-sensitive, focal-loss model: 99.83% (accuracy), 99.93% (precision), 96.89% (recall), and 98.30 (F-measure). We note that there is an inadequate amount of information provided on the preprocessing stage for Bot-IoT.

Ge et al. [46] trained a feedforward neural network [47] and an SVM on Bot-IoT to evaluate performance for binary and multi-class classification. About 11,175,000 Bot-IoT instances were selected. After feature extraction and data preprocessing, the dataset was split in a 64:16:20 ratio for training, validation, and testing, respectively. To address class imbalance, higher weights were assigned to underrepresented classes. Class weights for the training data were obtained by dividing the packet count for each class by the total packet count and then inverting the quotient. Classification results show that the neural network outperformed the SVM. For binary classification with the neural network, the best score for accuracy was 100%, while the best scores for precision, recall, and F-measure were all 99.90%. While multi-class classification scores were provided for the SVM classifier, binary classification scores were not. Hence, the binary classification scores for the neural network cannot be compared with classification scores for the SVM. This detracts from the authors' conclusion that the feedforward neural network is the better classifier.

Finally, to detect DDoS attacks, Soe et al. [48] trained a feedforward neural network on 477 normal instances and about 1,900,000 DDoS instances from Bot-IoT. The dataset was split in a ratio of 66:34 for training and testing, and the SMOTE algorithm was utilized to address class imbalance. After the application of SMOTE during data preprocessing, the training dataset contained about 1,300,000 instances for each class, while the test dataset contained 655,809 normal instances and 654,285 DDoS instances. The data was then normalized. Precision, Recall, and F-measure scores were all 100%. We point out that there is a lack of information on data cleaning in the study. In addition, it is unclear why the authors believe that balancing the classes

(positive to negative class ratio of 50:50) is the optimal solution. Also, in their paper, only the DT classifier is used, i.e., reliance on only one classifier.

We again note that after reviewing these five works, none were found to be solely based on Bot-IoT information theft detection. In addition, we use ensemble feature selection for building our predictive models.

Methodology

Data cleaning

As discussed in the next three paragraphs, there are six features in Bot-IoT that do not provide generalizable information [49].

The *pkSeqID* and *seq* features were removed because they are row or sequence identifiers and only provide information regarding order. Removing *pkSeqID* was obvious based on the definition provided by Koroniotis et al. [2]. However, removing *seq* was less so, as *seq* has been highly utilized in many studies [49]. Based on our consultation with the developers of the Argus network security tool and on our own investigation, we discovered that *seq* is a monotonically increasing sequence number pertaining to the records processed by the tool. With this clarification, we determined that it did not provide any additional relevant or generalizable information for our models.

The features *stime* and *ltime* are timestamps corresponding to the start packet time and last packet time for each instance. While they are useful for determining key features like duration or rate, this information is already provided by the features *dur*, *rate*, *srate*, and *drate*. With that information already present, we believe that both *stime* and *ltime* are unlikely to provide any additional information and may contribute to the overfitting of data by our models.

The *saddr* and *daddr* features pertain to the source and destination Internet Protocol (IP) addresses for each of the instances. While this information can provide highly relevant contextual information for security analysts, we chose to exclude it because private IP addresses can vary from network to network. Should a model learn to attribute a behavior based entirely or partially on the source or destination IP, it would be ineffective if tested against a dataset generated with different IP addresses.

We also discovered that many instances using Internet Control Message Protocol (c) have a hexadecimal value for the *sport* and *dport* features or are missing values for these features. Because of this, we changed all missing and hexadecimal ICMP values for *sport* and *dport* to -1 to indicate an invalid port value.

Data transformations

The *flgs_number*, *fproto_number*, *sport*, *dport*, and *state_number* categorical features were one-hot encoded. The one-hot encoding process, which was implemented by CatBoost's Ordered Target Statistics[2], transforms categorical features into dummy variables. We also performed feature scaling to provide a [0,1] normalized range for all numerical features.

[2] https://contrib.scikit-learn.org/category_encoders/catboost.html.

Data sampling

RUS is a technique to deal with class imbalance where we discard members of the majority class until the ratio of instances of majority and minority class members reaches a desired level. If the experiment involved RUS, we applied it to the training data only. The desired minority-to-majority class ratios we used RUS to obtain are those from [50], where the authors report yield strong results. These ratios are 1:1, 1:3, and 1:9. However, given the initial ratios of minority-to-majority class in the Information Theft dataset (1587:9543), it was only possible to do RUS for the 1:1 and 1:3 ratios.

Ensemble feature selection

A further preprocessing step we employed for some experiments is our ensemble FST. The FST is a two-step process. For the first step, we employed three filter-based feature ranking techniques. They are the information gain [51], information gain ratio [52], and Chi squared (Chi 2) [53] feature ranking techniques. We also ranked features according to four supervised learning-based feature ranking techniques. Feature importance lists from RF, CatBoost, XGBoost, and LightGBM served as the basis for the supervised feature ranking techniques.

We then took the 20 highest ranked attributes. We decided to use 20 features based on results of previous studies [54]. Therefore, we searched for features occurring in a set number of 7 rankings, where that number ranges from 4 to 7. Put another way, since we have 7 rankings, we require a majority of rankings to agree that a feature is among the 20 most important features in order to select it. This yielded 4 datasets sets of selected features, which we called the 4 Agree, 5 Agree, 6 Agree, and 7 Agree datasets. The tables in Appendix A show the features selected for the supervised-based feature ranking techniques, the filter-based feature ranking techniques, and the 4, 5, 6 and 7 Agree datasets.

Classifiers and performance metrics

This study involves four ensemble classifiers (CatBoost, Light GBM, XGBoost, and RF) and four non-ensemble classifiers (DT, LR, NB, and an MLP). These classifiers belong to various machine learning families of algorithms and are widely considered to be reliable [55]. CatBoost, Light GBM, and XGBoost were implemented with their respective self-named Python libraries, while the other classifiers were implemented with Scikit-learn.[3]

CatBoost, Light GBM, and XGBoost are gradient-boosted decision trees (GBDTs) [56], which are ensembles of sequentially trained trees. An ensemble classifier combines weak algorithms, or instances of an algorithm, into a strong learner. CatBoost relies on ordered boosting to order the instances used for fitting DTs. Light GBM is characterized by gradient-based one-side sampling and exclusive feature bundling. One-side sampling disregards a significant fraction of instances with small gradients, and exclusive feature bundling categorizes mutually exclusive features to reduce variable count. XGBoost utilizes a sparsity-aware algorithm and a weighted quantile sketch. Sparsity is defined by zero or missing values, and a weighted quantile sketch benefits from approximate tree

[3] https://scikit-learn.org/stable/.

learning [57] to support pruning and merging tasks. RF is also an ensemble of DTs, but unlike the GBDTs, it uses a bagging technique [58].

DT is a non-parametric approach that offers a simplistic tree-like representation of observed data. Each internal node represents a test on a specified feature, and each branch represents the outcome of a particular test. Each leaf node represents a class label. LR uses a sigmoidal function to generate probability values. Predictions for class membership are centered on a specified probability threshold. NB uses conditional probability to determine class membership. This classifier is considered "naive" because it operates on the assumption that features are independent of each other. An MLP is a type of artificial neural network with fully connected nodes. It utilizes a non-linear activation function and contains an input layer, one or more hidden layers and an output layer.

In our work, we used more than one performance metric (AUC and AUPRC) to better assess the challenge of evaluating classifiers. AUC is the area under the receiver operating characteristic (ROC) curve, which is a plot of true positive rate (TPR) versus false positive rate (FPR). This metric incorporates all classification thresholds represented by the curve [59] and is essentially a summary of overall model performance. AUPRC is the area under the Precision-Recall curve. This metric shows the trade-off between precision and recall for specific classification thresholds [60].

Parameters and cross validation

Before we trained our classifiers, we used default hyperparameters and/or tuned hyperparameters to help control the learning process. Hyperparameter tuning was effected by RandomizedSearchCV[4], a module of Scikit-learn. We list these tuned parameters in Appendix B.

During training, we implemented ten iterations of stratified five-fold cross-validation, which means there are 50 performance scores obtained per classifier for each metric. For k-fold cross-validation, every instance is placed in a validation set once and placed in a training set k-1 times. This means that for five-fold cross-validation, every instance gets to be in a validation set once and in a training set four times. The stratified component of cross-validation aims to ensure that each class is equally represented across each fold [61]. Because we randomly shuffled instances before cross-validation, certain algorithms such as LR may yield different results when the order of instances is changed [62]. One way of addressing the undesirable effect of this randomness is by performing several iterations, as we have done [63]. Note that each AUC or AUPRC value shown in the classification performance tables is an average of the 50 performance scores.

High-level methodology overview

For each of the attack types (information theft, data exfiltration, and keylogging), we divided our experiments into those that did not use ensemble FSTs and those that used them. For the tasks that did not involve ensemble FSTs, we first performed experimentation where we did not use hyperparameter tuning or data sampling.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

Table 2 Classification results for the information theft dataset no sampling default hyperparameter values experiments; mean and standard deviations of AUC and AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC	Mean AUPRC	SD AUPRC
CatBoost	0.99703	0.00199	0.99984	0.00027
DT	0.99575	0.00216	0.99393	0.00275
Light GBM	0.99670	0.00219	0.99936	0.00114
LR	0.99285	0.00287	0.99691	0.00238
MLP	0.99483	0.00261	0.99851	0.00155
NB	0.99291	0.00254	0.97743	0.00523
RF	0.99553	0.00235	0.99969	0.00056
XGBoost	0.99649	0.00223	0.99958	0.00066

Second, we performed experimentation where we applied data sampling. Third, we performed experimentation where we applied both data sampling and hyperparameter tuning. For the tasks that involved ensemble FSTs, feature selection was a necessary component for all three experimentation steps.

Results and discussion

In this section, we present results for classification performance, with and without the application of FSTs, and also present statistical analyses. Results are compartmentalized by attack type (information theft, data exfiltration, and keylogging).

Experiment names indicate the classifier used, whether we apply RUS (and in what ratio), and whether we apply hyperparameter tuning. For example, the experiment name "MLP Tuned RUS 1:3 6 Agree" indicates we use the MLP classifier, with hyperparameters tuned by RandomizedSearchCV, with RUS to a 1:3 ratio, and the 6 Agree FST. If there is no mention of RUS or a ratio in the experiment name this means we did not apply RUS.

Information theft

Information theft without ensemble feature selection

Table 2 contains results for experiments with default hyperparameters and no RUS. For AUC, the highest score of 0.99703 was obtained by CatBoost. For AUPRC, the highest score of 0.99984 was obtained by CatBoost.

Table 3 contains results for experiments with default hyperparameters after RUS is applied. For AUC, the highest score of 0.99842 was obtained by CatBoost with a balanced class ratio of 1:1. For AUPRC, the highest score of 0.99983 was obtained by CatBoost with a balanced class ratio of 1:1.

Table 4 contains results for experiments after RUS and hyperparameter tuning are applied. For AUC, CatBoost with a balanced class ratio of 1:1 obtained the highest score of 0.99803. For AUPRC, CatBoost with a minority-to-majority class ratio of 1:3 obtained the highest score of 0.99980.

Table 3 Classification results for the information theft dataset with sampling and default hyperparameters experiments; mean and standard deviations of AUC and AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC	Mean AUPRC	SD AUPRC
CatBoost RUS 1:3	0.99836	0.00124	0.99982	0.00030
CatBoost RUS 1:1	0.99842	0.00130	0.99983	0.00029
DT RUS 1:3	0.99560	0.00253	0.99115	0.00358
DT RUS 1:1	0.99567	0.00201	0.98735	0.00468
Light GBM RUS 1:3	0.99723	0.00190	0.99934	0.00111
Light GBM RUS 1:1	0.99759	0.00188	0.99954	0.00079
LR RUS 1:3	0.99304	0.00280	0.99704	0.00228
LR RUS 1:1	0.99394	0.00258	0.99641	0.00419
MLP RUS 1:3	0.99498	0.00247	0.99844	0.00170
MLP RUS 1:1	0.99476	0.00241	0.99649	0.01032
NB RUS 1:3	0.99266	0.00254	0.97693	0.00522
NB RUS 1:1	0.99247	0.00276	0.97723	0.00736
RF RUS 1:3	0.99682	0.00207	0.99972	0.00050
RF RUS 1:1	0.99727	0.00172	0.99968	0.00057
XGBoost RUS 1:3	0.99711	0.00182	0.99956	0.00068
XGBoost RUS 1:1	0.99723	0.00163	0.99958	0.00061

Table 4 Classification results for the information theft dataset with sampling and tuned hyperparameters experiments; mean and standard deviations of AUC and AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC	Mean AUPRC	SD AUPRC
CatBoost Tuned RUS 1:3	0.99727	0.00188	0.99980	0.00032
CatBoost Tuned RUS 1:1	0.99803	0.00149	0.99979	0.00035
DT Tuned RUS 1:3	0.99362	0.00330	0.99524	0.00281
DT Tuned RUS 1:1	0.99357	0.00395	0.99337	0.00424
Light GBM Tuned RUS 1:3	0.99539	0.00305	0.99915	0.00106
Light GBM Tuned RUS 1:1	0.99569	0.00240	0.99925	0.00073
LR Tuned RUS 1:3	0.99495	0.00241	0.99872	0.00082
LR Tuned RUS 1:1	0.99499	0.00231	0.99658	0.01024
MLP Tuned RUS 1:3	0.99398	0.00372	0.99782	0.00188
MLP Tuned RUS 1:1	0.99420	0.00363	0.99710	0.00555
NB Tuned RUS 1:3	0.99266	0.00254	0.97693	0.00522
NB Tuned RUS 1:1	0.99247	0.00276	0.97723	0.00736
RF Tuned RUS 1:3	0.99671	0.00213	0.99960	0.00060
RF Tuned RUS 1:1	0.99719	0.00181	0.99967	0.00051
XGBoost Tuned RUS 1:3	0.99688	0.00198	0.99957	0.00071
XGBoost Tuned RUS 1:1	0.99711	0.00183	0.99957	0.00063

Information theft with ensemble feature selection

Due to the number of experiments performed, Tables 5 and 6 only report the best performance by classifier, over all combinations of RUS, hyperparameter tuning, and ensemble FSTs. Table 5 shows AUC scores obtained. CatBoost, with a class ratio of

Table 5 Maximum AUC by classifier for the information theft attack type; mean and standard deviations of AUC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC
LR Tuned 7 Agree	0.99504	0.00218
CatBoost RUS 1:3 5 Agree	0.99838	0.00131
DT 4 Agree	0.99589	0.00199
Light GBM RUS 1:1 6 Agree	0.99757	0.00192
RF RUS 1:1 4 Agree	0.99732	0.00182
MLP Tuned 6 Agree	0.99537	0.00242
NB 7 Agree	0.99324	0.00203
XGBoost RUS 1:1 4 Agree	0.99731	0.00169

Table 6 Maximum AUPRC by classifier for the information theft attack type; mean and standard deviations of AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUPRC	SD AUPRC
LR Tuned RUS 1:3 5 Agree	0.99872	0.00082
CatBoost Tuned 7 Agree	0.99986	0.00019
DT Tuned 6 Agree	0.99733	0.00169
Light GBM Tuned 7 Agree	0.99956	0.00058
RF RUS 1:3 6 Agree	0.99976	0.00042
MLP Tuned 7 Agree	0.99908	0.00108
NB Tuned RUS 1:1 7 Agree	0.97858	0.00625
XGBoost RUS 1:3 7 Agree	0.99964	0.00050

Table 7 ANOVA for classifier, RUS, hyperparameters and FST as factors of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Classifier	7	2.66	0.38	5531.44	0.0000
RUS	2	0.04	0.02	326.04	0.0000
Hyperparameters	1	0.01	0.01	113.22	0.0000
FST	4	0.01	0.00	19.18	0.0000
Residuals	11985	0.82	0.00		

Table 8 ANOVA for classifier, RUS, hyperparameters and FST as factors of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Classifier	7	13.34	1.91	5767.49	0.0000
RUS	2	0.16	0.08	249.67	0.0000
Hyperparameters	1	0.00	0.00	8.15	0.0043
FST	4	0.17	0.04	131.17	0.0000
Residuals	11985	3.96	0.00		

Table 9 HSD test groupings after ANOVA of AUC for the Classifier factor

Group a consists of: CatBoost
Group ab consists of: LightGBM
Group bc consists of: XGBoost
Group c consists of: RF
Group d consists of: DT
Group e consists of: MLP
Group f consists of: LR
Group g consists of: NB

Table 10 HSD test groupings after ANOVA of AUC for the RUS factor

Group a consists of: 1:1, 1:3
Group b consists of: none

Table 11 HSD test groupings after ANOVA of AUC for the hyperparameters factor

Group a consists of: Tuned
Group b consists of: Default

1:3 and 5 Agree FST, yielded the highest score in this table of 0.99838. Table 6 shows AUPRC scores obtained. CatBoost with a 7 Agree FST and no RUS produced the highest score in this table of 0.99986.

Information theft statistical analysis

To understand the statistical significance of the classification performance scores, we run analysis of variance (ANOVA) tests. ANOVA establishes whether there is a significant difference between group means [64]. A 99% ($\alpha = 0.01$) confidence level is used for our ANOVA tests. The results are shown in Tables 7 and 8, where *Df* is the degrees of freedom, *Sum Sq* is the sum of squares, *Mean Sq* is the mean sum of squares, *F value* is the F-statistic, and *Pr(>F)* is the p-value.

The first ANOVA test we run for Information Theft evaluates the impact of factors on performance in terms of AUC. As shown in Table 7, the p-value associated with every factor is practically 0. Therefore, we conclude all factors have a significant impact on performance in terms of AUC.

The second ANOVA test we run for information theft evaluates the impact of factors on performance in terms of AUPRC. As shown in Table 8, the p-value associated with every factor is practically 0. Therefore, we conclude all factors have a significant impact on performance in terms of AUPRC

Since all factors significantly impact performance for both AUC and AUPRC, Tukey's Honestly Significant Difference (HSD) tests [65] are performed to determine which groups are significantly different from each other. Letter groups assigned via the Tukey method indicate similarity or significant differences in performance results within a factor.

Table 12 HSD test groupings after ANOVA of AUC for the FST factor

Group a consists of: All Features, 4 Agree, 5 Agree, 6 Agree
Group b consists of: 7 Agree

Table 13 HSD test groupings after ANOVA of AUPRC for the Classifier factor

Group a consists of: CatBoost, RF, XGBoost, LightGBM
Group b consists of: MLP
Group c consists of: DT
Group d consists of: LR
Group e consists of: NB

Table 14 HSD test groupings after ANOVA of AUPRC for the RUS factor

Group a consists of: none
Group b consists of: 1:3
Group c consists of: 1:1

Table 15 HSD test groupings after ANOVA of AUPRC for the hyperparameters factor

Group a consists of: Default
Group b consists of: Tuned

Table 16 HSD test groupings after ANOVA of AUPRC for the FST factor

Group a consists of: 5 Agree, 6 Agree, 4 Agree, All Features
Group b consists of: 7 Agree

With regard to AUC, we first apply the HSD test to the group classifier factor, with the results shown in Table 9. In this table, the highest ranked classifiers are the GBDT classifiers, followed by RF. Next, we cover the HSD test for the RUS factor. The results of this test, as shown in Table 10, reveal that undersampling, either to the 1:1 or 1:3 minority-to majority-class ratios yield similar performance, which is better than not using undersampling. For the hyperparameter tuning factor, the results of the HSD test provided in Table 11 indicate that hyperparameter tuning is generally better than the default hyperparameter values. Table 12 shows the HSD test results for the FST factor. Here, we find the only FST that has significantly worse performance than the others is the 7 Agree. Otherwise, using all features or any other FST yields similar results. Since the 6 Agree FST yields the smallest number of features in group 'a', we recommend using the features that the 6 Agree FST selects. Classifiers usually train faster on datasets with comparatively fewer features.

In terms of AUPRC, the HSD test results for the group classifier factor, as shown in Table 13, indicate that the GBDT classifiers yield the best performance. However, unlike

Table 17 Classification results for the data exfiltration dataset no sampling default hyperparameter values experiments; mean and standard deviations of AUC and AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC	Mean AUPRC	SD AUPRC
CatBoost	0.98126	0.02518	0.99446	0.00787
DT	0.97395	0.02657	0.96053	0.02803
Light GBM	0.98043	0.02721	0.98507	0.02160
LR	0.94149	0.03630	0.92768	0.07234
MLP	0.95924	0.03355	0.95868	0.04305
NB	0.97410	0.02317	0.82526	0.04383
RF	0.96438	0.03388	0.99208	0.01231
XGBoost	0.97747	0.02833	0.98798	0.01983

the case for performance in terms of AUC, the performance of RF, which is not a GBDT classifier, is not significantly less than that of the GBDT classifiers. The HSD test results for the RUS factor are provided in Table 14. It turns out that not doing RUS is the best choice. As shown in Table 15, the HSD test results for the hyperparameter tuning factor indicate that default hyperparameter values are the better choice. Finally, Table 16 provides HSD test results for the FST factor. These results are similar to those we see for performance in terms of AUC. Only the 7 agree FST yields a lower performance than other levels of FST. Therefore, the 6 Agree FST is preferred, since it has fewer features than the 4 Agree and 5 Agree FSTs.

Information theft conclusion

The GBDT classifiers, and in some cases RF, yield similar performance. HSD test results indicate their performance is better than that of the other classifiers. While the ANOVA tests reveal that hyperparameter tuning is a significant factor, we see that hyperparameter tuning yields better performance in terms of AUC, but worse performance in terms of AUPRC. Therefore, we conclude that hyperparameter tuning is only necessary for optimizing performance in terms of AUC. For the case of performance in terms of AUC, undersampling has a positive impact on performance. However, for performance in terms of AUPRC, we see that the best strategy for RUS is to not use it. We find there is no impact on performance in terms of AUC or AUPRC when we apply the 4, 5, or 6 Agree FSTs, since the HSD test ranks these in the best performing group, along with the technique where we use all features. Furthermore, these techniques all outperform the 7 Agree FST. Since using fewer features yields performance equivalent to using all features, we conclude that we should use the 6 Agree FST in future work. The 6 Agree FST yields the smallest number of features, and model training is usually faster with fewer features.

Data exfiltration

Data exfiltration without ensemble feature selection

Table 17 contains results for experiments with default hyperparameters and no RUS. For AUC, the highest score of 0.98126 was obtained by CatBoost. For AUPRC, the highest score of 0.99446 was obtained by CatBoost.

Table 18 Classification results for the data exfiltration dataset with sampling and default hyperparameters experiments; mean and standard deviations of AUC and AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC	Mean AUPRC	SD AUPRC
CatBoost RUS 1:9	0.98588	0.02077	0.99148	0.01393
CatBoost RUS 1:3	0.98990	0.01378	0.99000	0.01384
CatBoost RUS 1:1	0.98918	0.01093	0.98100	0.03626
DT RUS 1:9	0.97883	0.02418	0.87681	0.04482
DT RUS 1:3	0.98413	0.01617	0.78582	0.06857
DT RUS 1:1	0.98131	0.01206	0.70475	0.07812
Light GBM RUS 1:9	0.98499	0.01852	0.98983	0.01417
Light GBM RUS 1:3	0.98947	0.01300	0.98032	0.03246
Light GBM RUS 1:1	0.98744	0.01019	0.94426	0.11722
LR RUS 1:9	0.95021	0.03785	0.90353	0.10087
LR RUS 1:3	0.95646	0.03519	0.80981	0.14864
LR RUS 1:1	0.96741	0.02800	0.71585	0.13645
MLP RUS 1:9	0.96662	0.02902	0.92406	0.10731
MLP RUS 1:3	0.96939	0.02808	0.81506	0.16246
MLP RUS 1:1	0.96725	0.02679	0.66960	0.13862
NB RUS 1:9	0.97480	0.02199	0.81701	0.05655
NB RUS 1:3	0.97185	0.02087	0.70617	0.06592
NB RUS 1:1	0.96574	0.02068	0.64355	0.06437
RF RUS 1:9	0.97567	0.02542	0.98981	0.01413
RF RUS 1:3	0.98787	0.01476	0.98985	0.01436
RF RUS 1:1	0.99011	0.01116	0.98924	0.01605
XGBoost RUS 1:9	0.98642	0.01975	0.98677	0.02014
XGBoost RUS 1:3	0.98707	0.01486	0.97802	0.04477
XGBoost RUS 1:1	0.98684	0.01235	0.94733	0.08212

Table 18 contains results for experiments with default hyperparameters after RUS is applied. For AUC, the highest score of 0.99011 was obtained by RF with a balanced class ratio of 1:1. For AUPRC, the highest score of 0.99148 was obtained by CatBoost with a minority-to-majority class ratio of 1:9.

Table 19 contains results for experiments after RUS and hyperparameter tuning are applied. For AUC, CatBoost with a minority-to-majority class ratio of 1:9 obtained the highest score of 0.99281. For AUPRC, CatBoost with a minority-to-majority class ratio of 1:9 obtained the highest score of 0.99292.

Data exfiltration with ensemble feature selection

Due to the number of experiments performed, Tables 20 and 21 only report the best performance by classifier, over all combinations of RUS, hyperparameter tuning, and ensemble FSTs. Table 20 shows AUC scores obtained. CatBoost, with a minority-to-majority class ratio of 1:9 and 4 Agree FST, yielded the highest score in this table of 0.99076. Table 21 shows AUPRC scores obtained. CatBoost with a 5 Agree FST and no RUS produced the highest score in this table of 0.99448.

Table 19 Classification results for the data exfiltration dataset with sampling and tuned hyperparameters experiments; mean and standard deviations of AUC and AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC	Mean AUPRC	SD AUPRC
CatBoost Tuned RUS 1:9	0.99281	0.00986	0.99292	0.01192
CatBoost Tuned RUS 1:3	0.98816	0.01388	0.98791	0.01652
CatBoost Tuned RUS 1:1	0.98728	0.01193	0.98444	0.01864
DT Tuned RUS 1:9	0.96722	0.03625	0.91543	0.07238
DT Tuned RUS 1:3	0.96993	0.03208	0.84267	0.08491
DT Tuned RUS 1:1	0.97706	0.01870	0.74923	0.08163
Light GBM Tuned RUS 1:9	0.98769	0.01770	0.98951	0.01959
Light GBM Tuned RUS 1:3	0.98960	0.01336	0.95978	0.08703
Light GBM Tuned RUS 1:1	0.98446	0.01508	0.91785	0.09834
LR Tuned RUS 1:9	0.95866	0.03334	0.90807	0.10925
LR Tuned RUS 1:3	0.96372	0.03201	0.81231	0.15087
LR Tuned RUS 1:1	0.96302	0.02723	0.67686	0.15659
MLP Tuned RUS 1:9	0.87040	0.11157	0.93840	0.07158
MLP Tuned RUS 1:3	0.95885	0.03652	0.85402	0.14352
MLP Tuned RUS 1:1	0.96377	0.02815	0.72101	0.15642
NB Tuned RUS 1:9	0.97481	0.02199	0.81701	0.05655
NB Tuned RUS 1:3	0.97188	0.02087	0.70621	0.06589
NB Tuned RUS 1:1	0.96574	0.02068	0.64355	0.06437
RF Tuned RUS 1:9	0.98592	0.01871	0.98808	0.01699
RF Tuned RUS 1:3	0.98663	0.01519	0.98461	0.01831
RF Tuned RUS 1:1	0.98708	0.01384	0.97930	0.02606
XGBoost Tuned RUS 1:9	0.98292	0.02143	0.98610	0.02217
XGBoost Tuned RUS 1:3	0.98713	0.01487	0.97596	0.03204
XGBoost Tuned RUS 1:1	0.98634	0.01371	0.93328	0.08744

Table 20 Maximum AUC by classifier for the data exfiltration attack type; mean and standard deviations of AUC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC
LR Tuned RUS 1:1 7 Agree	0.96968	0.02346
CatBoost Tuned RUS 1:9 4 Agree	0.99076	0.01461
DT RUS 1:3 4 Agree	0.98398	0.01531
Light GBM Tuned RUS 1:3 5 Agree	0.98947	0.01328
RF RUS 1:1 6 Agree	0.99024	0.01120
MLP RUS 1:3 4 Agree	0.96914	0.02814
NB Tuned RUS 1:3 7 Agree	0.97958	0.01523
XGBoost RUS 1:3 6 Agree	0.98737	0.01497

Data exfiltration statistical analysis

As done previously for the Information Theft statistical analysis, we perform ANOVA tests for the Data Exfiltration dataset. The results are shown in Tables 22 and 23,

Table 21 Maximum AUPRC by classifier for the data exfiltration attack type; mean and standard deviations of AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUPRC	SD AUPRC
LR Tuned RUS 1:9 6 Agree	0.93654	0.06464
CatBoost 5 Agree	0.99448	0.00784
DT Tuned 6 Agree	0.96776	0.02877
Light GBM RUS 1:9 6 Agree	0.99079	0.01350
RF 4 Agree	0.99199	0.01341
MLP 4 Agree	0.95897	0.04293
NB Tuned RUS 1:9 7 Agree	0.84859	0.04161
XGBoost 5 Agree	0.98823	0.01938

Table 22 ANOVA for classifier, RUS, hyperparameters and FST as factors of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Classifier	7	137.67	19.67	3564.07	0.0000
RUS	3	4.05	1.35	244.86	0.0000
Hyperparameters	1	0.95	0.95	171.29	0.0000
FST	4	0.48	0.12	21.65	0.0000
Residuals	15984	88.20	0.01		

Table 23 ANOVA for classifier, RUS, hyperparameters and FST as factors of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Classifier	7	891.99	127.43	7741.19	0.0000
RUS	3	67.63	22.54	1369.58	0.0000
Hyperparameters	1	0.99	0.99	60.31	0.0000
FST	4	4.94	1.24	75.07	0.0000
Residuals	15984	263.11	0.02		

Table 24 HSD test groupings after ANOVA of AUC for the classifier factor

Group a consists of: CatBoost
Group ab consists of: LightGBM, XGBoost, RF
Group b consists of: DT
Group c consists of: MLP
Group d consists of: LR
Group e consists of: NB

where *Df* is the degrees of freedom, *Sum Sq* is the sum of squares, *Mean Sq* is the mean sum of squares, *F value* is the F-statistic, and *Pr(>F)* is the *p*-value.

The first ANOVA test we run for Data Exfiltration evaluates the impact of factors on performance in terms of AUC. As shown in Table 22, the *p*-value associated with every

Table 25 HSD test groupings after ANOVA of AUC for the RUS factor

Group a consists of: 1:3, 1:1
Group b consists of: 1:9
Group c consists of: none

Table 26 HSD test groupings after ANOVA of AUC for the hyperparameters factor

Group a consists of: Tuned
Group b consists of: Default

Table 27 HSD test groupings after ANOVA of AUC for the FST factor

Group a consists of: 7 Agree
Group b consists of: 5 Agree, All Features, 4 Agree, 6 Agree

Table 28 HSD test groupings after ANOVA of AUPRC for the classifier factor

Group a consists of: CatBoost, RF, LightGBM, XGBoost
Group b consists of: DT
Group c consists of: MLP
Group d consists of: NB, LR

Table 29 HSD test groupings after ANOVA of AUPRC for the RUS factor

Group a consists of: none
Group b consists of: 1:9
Group c consists of: 1:3
Group d consists of: 1:1

factor is practically 0. Therefore, we conclude all factors have a significant impact on performance in terms of AUC.

The second ANOVA test we run for data exfiltration evaluates the impact of factors on performance in terms of AUPRC. As shown in Table 23, the p-value associated with every factor is practically 0. Therefore, we conclude all factors have a significant impact on performance in terms of AUPRC

With regard to AUC, we begin with the HSD test applied to the group classifier factor, with the results shown in Table 24. In this table, the GBDT classifiers and RF show the best performance. However, CatBoost is ranked above all others. Next, we address the HSD test for the RUS factor. The results of this test, as shown in Table 25, indicate that the 1:1 or 1:3 class ratios yield the best performance. For the hyperparameter tuning factor, the results of the HSD test provided in Table 26 indicate that hyperparameter tuning is a better alternative than the default hyperparameter values. Table 27 shows the HSD test results for the FST factor. Here, we find the

Table 30 HSD test groupings after ANOVA of AUPRC for the hyperparameters factor

Group a consists of: Tuned
Group b consists of: Default

Table 31 HSD test groupings after ANOVA of AUPRC for the FST factor

Group a consists of: 6 Agree, 5 Agree, All Features, 4 Agree
Group b consists of: 7 Agree

Table 32 Classification results for the keylogging dataset no sampling default hyperparameter values experiments; mean and standard deviations of AUC and AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC	Mean AUPRC	SD AUPRC
CatBoost	0.99624	0.00218	0.99987	0.00025
DT	0.99605	0.00225	0.99484	0.00229
Light GBM	0.99643	0.00218	0.99925	0.00131
LR	0.99415	0.00234	0.99718	0.00282
MLP	0.99574	0.00236	0.99875	0.00145
NB	0.99191	0.00201	0.97511	0.00533
RF	0.99573	0.00219	0.99978	0.00040
XGBoost	0.99608	0.00210	0.99957	0.00069

7 Agree FST yields better performance than any other FST. This is an ideal result, since the less complex model with fewer features outperforms the models that have more features.

In terms of AUPRC, the HSD test results for the group classifier factor, as shown in Table 28, indicate that the GBDT classifiers and RF yield the best performance. The HSD test results for the RUS factor are provided in Table 29. It turns out that not doing RUS is the best choice. As shown in Table 30, the HSD test results for the hyperparameter tuning factor indicate that hyperparameter tuning is a better alternative than the default hyperparameter values. Finally, Table 31 provides HSD test results for the FST factor. Since the 6 Agree technique yields fewer features than the 4 Agree and 5 Agree techniques, we prefer the 6 Agree.

Data exfiltration conclusion

Similar to results for classifying the Information Theft attack type data, the GBDT classifiers yield the best performance, along with RF. For classifying the data exfiltration attack type, we find results for AUC and AUPRC with hyperparameter tuning are higher than those where we use default parameter tuning. As in the case with classifying the Information Theft dataset, there are mixed results for the application of RUS. For performance in terms of AUC, applying RUS to the data before training improves performance. However, for performance in terms of AUPRC, not using RUS yields better results. For

Table 33 Classification results for the keylogging dataset with sampling and default hyperparameters experiments; mean and standard deviations of AUC and AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC	Mean AUPRC	SD AUPRC
CatBoost RUS 1:3	0.99779	0.00151	0.99980	0.00034
CatBoost RUS 1:1	0.99796	0.00165	0.99981	0.00036
DT RUS 1:3	0.99557	0.00230	0.99103	0.00343
DT RUS 1:1	0.99573	0.00206	0.98674	0.00620
Light GBM RUS 1:3	0.99674	0.00201	0.99944	0.00099
Light GBM RUS 1:1	0.99720	0.00199	0.99962	0.00053
LR RUS 1:3	0.99455	0.00236	0.99720	0.00283
LR RUS 1:1	0.99524	0.00233	0.99650	0.00576
MLP RUS 1:3	0.99591	0.00212	0.99877	0.00148
MLP RUS 1:1	0.99617	0.00192	0.99750	0.00985
NB RUS 1:3	0.99176	0.00211	0.97408	0.00570
NB RUS 1:1	0.99189	0.00212	0.97173	0.00698
RF RUS 1:3	0.99649	0.00209	0.99971	0.00049
RF RUS 1:1	0.99688	0.00193	0.99961	0.00062
XGBoost RUS 1:3	0.99662	0.00197	0.99955	0.00065
XGBoost RUS 1:1	0.99685	0.00188	0.99956	0.00063

Table 34 Classification results for the keylogging dataset with sampling and tuned hyperparameters experiments; mean and standard deviations of AUC and AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC	Mean AUPRC	SD AUPRC
CatBoost Tuned RUS 1:3	0.99709	0.00191	0.99968	0.00054
CatBoost Tuned RUS 1:1	0.99749	0.00170	0.99971	0.00051
DT Tuned RUS 1:3	0.99468	0.00347	0.99405	0.00362
DT Tuned RUS 1:1	0.99412	0.00270	0.99259	0.00661
Light GBM Tuned RUS 1:3	0.99675	0.00200	0.99932	0.00097
Light GBM Tuned RUS 1:1	0.99674	0.00198	0.99957	0.00066
LR Tuned RUS 1:3	0.99589	0.00240	0.99786	0.00223
LR Tuned RUS 1:1	0.99609	0.00214	0.99688	0.00992
MLP Tuned RUS 1:3	0.99489	0.00220	0.99785	0.00203
MLP Tuned RUS 1:1	0.99187	0.01955	0.99679	0.00440
NB Tuned RUS 1:3	0.99240	0.00182	0.97520	0.00536
NB Tuned RUS 1:1	0.99259	0.00208	0.97334	0.00680
RF Tuned RUS 1:3	0.99702	0.00194	0.99945	0.00077
RF Tuned RUS 1:1	0.99668	0.00210	0.99963	0.00047
XGBoost Tuned RUS 1:3	0.99640	0.00190	0.99956	0.00058
XGBoost Tuned RUS 1:1	0.99677	0.00184	0.99957	0.00057

classifying data exfiltration attack type data, if AUC is the more important metric, then the best FST for classification performance is the 7 Agree FST. However, if AUPRC is the more important metric, then the 6 Agree FST yields a dataset with the fewest features, but yields performance similar to using all features.

Table 35 Maximum AUC by classifier for the keylogging attack type; mean and standard deviations of AUC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUC	SD AUC
LR Tuned RUS 1:3 7 Agree	0.99639	0.00227
CatBoost RUS 1:1 6 Agree	0.99807	0.00165
DT 4 Agree	0.99608	0.00232
Light GBM RUS 1:1 6 Agree	0.99737	0.00176
RF Tuned RUS 1:3 4 Agree	0.99707	0.00190
MLP RUS 1:3 7 Agree	0.99630	0.00212
NB Tuned RUS 1:3 7 Agree	0.99516	0.00173
XGBoost RUS 1:1 5 Agree	0.99693	0.00192

Table 36 Maximum AUPRC by classifier for the keylogging attack type; mean and standard deviations of AUPRC, (10 iterations of 5-fold cross-validation)

Experiment name	Mean AUPRC	SD AUPRC
LR Tuned 6 Agree	0.99836	0.00166
CatBoost 6 Agree	0.99988	0.00022
DT Tuned 4 Agree	0.99773	0.00166
Light GBM RUS 1:1 6 Agree	0.99965	0.00048
RF RUS 1:3 4 Agree	0.99972	0.00051
MLP RUS 1:1 7 Agree	0.99891	0.00102
NB RUS 1:1 7 Agree	0.98416	0.00628
XGBoost 4 Agree	0.99959	0.00064

Keylogging

Keylogging without ensemble feature selection

Table 32 contains results for experiments with default hyperparameters and no RUS. For AUC, the highest score of 0.99643 was obtained by Light GBM. For AUPRC, the highest score of 0.99987 was obtained by CatBoost.

Table 33 contains results for experiments with default hyperparameters after RUS is applied. For AUC, the highest score of 0.99796 was obtained by CatBoost with a balanced class ratio of 1:1. For AUPRC, the highest score of 0.99981 was obtained by CatBoost with a balanced class ratio of 1:1.

Table 34 contains results for experiments after RUS and hyperparameter tuning are applied. For AUC, CatBoost with a balanced class ratio of 1:1 obtained the highest score of 0.99749. For AUPRC, CatBoost with a balanced class ratio of 1:1 also obtained the highest score of 0.99971.

Keylogging with ensemble feature selection

Due to the number of experiments performed, Tables 35 and 36 only report the best performance by classifier, over all combinations of RUS, hyperparameter tuning, and ensemble FSTs. Table 35 shows AUC scores obtained. CatBoost with a 6 Agree FST and

Table 37 ANOVA for classifier, RUS, hyperparameters and FST as factors of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Classifier	7	1.64	0.23	6477.60	0.0000
RUS	2	0.01	0.01	172.54	0.0000
Hyperparameters	1	0.01	0.01	164.64	0.0000
FST	4	0.00	0.00	33.30	0.0000
Residuals	11985	0.43	0.00		

Table 38 ANOVA for classifier, RUS, hperparameters and FST as factors of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Classifier	7	9.21	1.32	7851.99	0.0000
RUS	2	0.07	0.03	203.36	0.0000
Hyperparameters	1	0.00	0.00	0.06	0.8137
FST	4	0.08	0.02	126.09	0.0000
Residuals	11985	2.01	0.00		

Table 39 HSD test groupings after ANOVA of AUC for the classifier factor

Group a consists of: CatBoost, LightGBM
Group ab consists of: XGBoost
Group b consists of: RF
Group c consists of: DT
Group d consists of: MLP
Group e consists of: NB
Group f consists of: LR

a balanced class ratio of 1:1 yielded the highest score in this table of 0.99807. Table 36 shows AUPRC scores obtained. CatBoost with a 6 Agree FST and no RUS produced the highest score in this table of 0.99988.

Keylogging statistical analysis

As done previously for the Information Theft and data exfiltration datasets, we perform ANOVA tests for the Keylogging dataset. The results are shown in Tables 37 and 38, where *Df* is the degrees of freedom, *Sum Sq* is the sum of squares, *Mean Sq* is the mean sum of squares, *F value* is the F-statistic, and *Pr(>F)* is the p-value.

The first ANOVA test we run for keylogging evaluates the impact of factors on performance in terms of AUC. As shown in Table 37, the p-value associated with every factor is practically 0. Therefore, we conclude all factors have a significant impact on performance in terms of AUC.

The second ANOVA test we run for keylogging evaluates the impact of factors on performance in terms of AUPRC. As shown in Table 38, the p-value associated with all factors, except hyperparameter tuning (0.8137) is practically 0. Therefore, we conclude that

Table 40 HSD test groupings after ANOVA of AUC for the RUS factor

Group a consists of: 1:1
Group b consists of: 1:3
Group c consists of: none

Table 41 HSD test groupings after ANOVA of AUC for the hyperparameters factor

Group a consists of: Tuned
Group b consists of: Default

Table 42 HSD test groupings after ANOVA of AUC for the FST factor

Group a consists of: All Features
Group ab consists of: 4 Agree, 5 Agree
Group b consists of: 6 Agree
Group c consists of: 7 Agree

Table 43 HSD test groupings after ANOVA of AUPRC for the classifier factor

Group a consists of: CatBoost, RF, XGBoost, LightGBM
Group b consists of: MLP
Group c consists of: DT
Group d consists of: LR
Group e consists of: NB

Table 44 HSD test groupings after ANOVA of AUPRC for the RUS factor

Group a consists of: none, 1:3
Group b consists of: 1:1

Table 45 HSD test groupings after ANOVA of AUPRC for the FST factor

Group a consists of: 5 Agree, 4 Agree, 6 Agree, All Features
Group b consists of: 7 Agree

only hyperparameter tuning does not have a significant impact on performance in terms of AUPRC.

With regard to AUC, we begin with the HSD test applied to the group classifier factor, with the results shown in Table 39. In this table, the GBDT classifiers and RF show the best performance. Next, we cover the HSD test for the RUS factor. The results of this test, as shown in Table 40, indicate that the 1:1 or 1:3 class ratios yield the best performance. For the hyperparameter tuning factor, the results of the HSD test provided in Table 41 indicate that hyperparameter tuning is a better alternative than the default

hyperparameter values. Table 42 shows the HSD test results for the FST factor. Here, we see that using the dataset with a larger number of features yields the best result.

In terms of AUPRC, the HSD test results for the group classifier factor, as shown in Table 43, indicate that the GBDT classifiers and RF yield the best performance. The HSD test results for the RUS factor are provided in Table 44. It turns out that not doing RUS is the best choice. Finally, Table 45 provides HSD test results for the FST factor. We prefer the 6 Agree FST since it yields performance similar to using all features, and it has the fewest features.

Keylogging conclusion

As with other attack types, we see the GBDT classifiers yield the best performance. The HSD tests for the influence of hyperparameter tuning on results in terms of AUC show that hyperparameter tuning yields better performance. However, for performance in terms of AUPRC, hyperparameter tuning is not significant. Similar to what we observe with results for classifying data exfiltration attack types, we see that if performance in terms of AUC is most important, then data sampling yields better results. However, for results in terms of AUPRC, we find that not applying data sampling gives better results. For performance in terms of AUPRC, we obtain similar results when using the 4 Agree, 5 Agree, 6 Agree FSTs or using all features, since all are in the HSD group 'a'. However, for performance in terms of AUC, the 4 Agree and 5 Agree FSTs are in category 'ab', while All Features is in category 'a'. This indicates that for AUC, the ensemble FSTs slightly impact performance.

Conclusion

The Bot-IoT dataset is geared toward the training of classifiers for the identification of malicious traffic in IOT networks. In this study, we examine the effect of ensemble FSTs on classification performance for information theft attack types. An ensemble feature selection approach is usually more efficient than its individual FSTs.

To accomplish our research goal, we investigate three datasets, one composed of Normal and Information Theft attack instances, another composed of Normal and data exfiltration attack instances, and a third composed of normal and keylogging attack instances. In general, we observed that our ensemble FSTs do not affect classification performance scores. However, our technique is useful because feature reduction lessens computational burden and provides clarity.

Future work will assess other classifiers that are trained on the identical datasets used in our study. There is also an opportunity to evaluate classifier performance, with respect to information theft detection, on other IOT intrusion detection datasets.

Appendix A

In this section, we provide a list of features and their definitions for the full processed Bot-IoT dataset (Table 46).

Table 46 Features and descriptions

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

* Dependent feature

Appendix B

Here, we report classification results where we use our ensemble FSTs. First, we report the 20 highest ranked features from each ranking technique in Tables 17 and 18. Throughout this case study the reader may notice a supervised feature ranking technique such as CatBoost or XGBoost may fail to yield 20 features. This is due to the fact that in some instances, classifiers do not assign importance to all features in a dataset. These rankings are followed by a report of which features are in each of the 4 Agree, 5 Agree, 6 Agree, and 7 Agree datasets (Tables 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58).

Feature selection for information theft

Table 47 Features by supervised ranking feature importance

XGBoost	CatBoost	Light GBM	RF
dport	dport	flgs_number	state_number
state_number	state_number	proto_number	dport
proto_number	max	dport	dur
pkts	bytes	state_number	max
rate	dur	dur	flgs_number
stddev	drate	sport	bytes
drate	dbytes	mean	dpkts
flgs_number	flgs_number	drate	dbytes
mean	pkts	rate	rate
dbytes	sport	min	stddev
dur	dpkts	sbytes	proto_number
dpkts	sbytes	dpkts	sbytes
max	proto_number	srate	srate
sport		pkts	sport
sum		dbytes	mean
min		stddev	drate
spkts		max	min
srate		bytes	spkts
sbytes		sum	
		spkts	

Table 48 Features by filter-based feature importance

Information gain	Information gain ratio	Chi 2
proto_number	proto_number	sbytes
state_number	state_number	dbytes
flgs_number	flgs_number	bytes
dport	dport	rate
bytes	bytes	pkts
sbytes	sbytes	spkts
rate	rate	drate
dur	dur	srate
sum	sum	dpkts
min	min	dur
mean	mean	sum
max	max	dport
dbytes	dbytes	state_number
srate	srate	proto_number
dpkts	dpkts	max
spkts	spkts	mean
drate	drate	min
pkts	pkts	stddev
sport	sport	flgs_number
stddev	stddev	sport

Table 49 Results of the 4-Agree and 5-Agree FSTs

4-Agree	5-Agree
dbytes	dbytes
sum	sum
srate	srate
max	max
drate	drate
mean	mean
pkts	pkts
rate	rate
flgs_number	flgs_number
dpkts	dpkts
sport	sport
proto_number	proto_number
bytes	bytes
min	min
dport	dport
state_number	state_number
spkts	spkts
sbytes	sbytes
stddev	stddev
dur	dur

Table 50 Results of the 6-Agree and 7-Agree FSTs

6-Agree	7-Agree
dbytes	proto_number
sum	flgs_number
srate	min
max	dport
drate	state_number
mean	rate
rate	stddev
flgs_number	dur
dpkts	
sport	
proto_number	
bytes	
min	
dport	
state_number	
spkts	
sbytes	
stddev	
dur	

Feature selection for data exfiltration

Table 51 Features by supervised ranking feature importance

XGBoost	CatBoost	Light GBM	RF
dport	sbytes	flgs_number	dport
bytes	dport	dport	state_number
dbytes	bytes	dur	bytes
rate	max	bytes	drate
dpkts	drate	state_number	srate
drate	dbytes	proto_number	dpkts
stddev	flgs_number	drate	dbytes
proto_number	stddev	rate	flgs_number
sum	min	stddev	proto_number
flgs_number	dpkts	sport	sum
dur	dur	dpkts	spkts
state_number		pkts	max
mean		mean	dur
spkts		dbytes	mean
pkts		min	sbytes
sport		max	min
max		sum	pkts
		srate	sport
		spkts	rate
		sbytes	

Table 52 Features by filter-based feature importance

Information gain	Information gain ratio	Chi 2
drate	drate	dbytes
dur	dur	bytes
sum	sum	sbytes
srate	srate	drate
rate	rate	dpkts
dbytes	dbytes	rate
proto_number	proto_number	spkts
sbytes	sbytes	pkts
flgs_number	flgs_number	dport
state_number	state_number	srate
max	max	sum
min	min	dur
bytes	bytes	state_number
mean	mean	min
dport	dport	mean
pkts	pkts	max
spkts	spkts	proto_number
dpkts	dpkts	flgs_number
stddev	stddev	stddev
sport	sport	sport

Table 53 Results of the 4-Agree and 5-Agree FSTs

4-Agree	5-Agree
dbytes	dbytes
sum	sum
srate	srate
max	max
drate	drate
mean	mean
pkts	pkts
rate	rate
flgs_number	flgs_number
dpkts	dpkts
sport	sport
proto_number	proto_number
bytes	bytes
min	min
dport	dport
state_number	state_number
spkts	spkts
sbytes	sbytes
stddev	stddev
dur	dur

Table 54 Results of the 6-Agree and 7-Agree FSTs

6-Agree	7-Agree
sum	proto_number
max	flgs_number
flgs_number	drate
proto_number	sport
bytes	dport
dpkts	state_number
drate	dur
mean	
sport	
dbytes	
dport	
state_number	
pkts	
spkts	
sbytes	
rate	
stddev	
dur	

Feature selection for keylogging

Table 55 Features by supervised ranking feature importance

XGBoost	CatBoost	Light GBM	RF
dport	dport	dport	state_number
state_number	state_number	dur	dport
stddev	bytes	flgs_number	bytes
pkts	dur	proto_number	dur
proto_number	max	drate	flgs_number
sum	proto_number	state_number	rate
dur	dbytes	sport	sbytes
dbytes	mean	rate	dbytes
flgs_number	sport	srate	dpkts
mean	pkts	mean	sum
rate		sbytes	drate
dpkts		min	min
sport		sum	proto_number
max		dbytes	sport
spkts		pkts	srate
drate		stddev	mean
srate		bytes	spkts
min		spkts	max
		dpkts	
		max	

Table 56 Features by filter-based feature importance

Information gain	Information gain ratio	Chi 2
proto_number	proto_number	bytes
state_number	state_number	sbytes
flgs_number	flgs_number	dbytes
dport	dport	rate
bytes	bytes	pkts
sbytes	sbytes	spkts
rate	rate	drate
dur	dur	dpkts
sum	sum	srate
min	min	dur
mean	mean	sum
max	max	dport
dbytes	dbytes	state_number
srate	srate	proto_number
dpkts	dpkts	max
spkts	spkts	mean
drate	drate	min
pkts	pkts	stddev
sport	sport	flgs_number
stddev	stddev	sport

Table 57 Results of the 4-Agree and 5-Agree FSTs

4-Agree	5-Agree
dbytes	dbytes
sum	sum
srate	srate
max	max
drate	drate
mean	mean
pkts	pkts
rate	rate
flgs_number	flgs_number
dpkts	dpkts
sport	sport
proto_number	proto_number
bytes	bytes
min	min
dport	dport
state_number	state_number
spkts	spkts
sbytes	sbytes
stddev	stddev
dur	dur

Table 58 Results of the 6-Agree and 7-Agree FSTs

6-Agree	7-Agree
sum	proto_number
srate	max
max	sport
flgs_number	mean
proto_number	dbytes
bytes	dport
dpkts	state_number
drate	dur
min	
mean	
sport	
dbytes	
dport	
state_number	
pkts	
spkts	
rate	
dur	

Appendix C

In this section, we report tuned hyperparameter values. Due to the number of experiments, we do not report hyperparameter values for every experiment. We report hyperparameter values for classifiers that yields best performance in terms of AUC or AUPRC over the possible levels of data sampling we employ. Since RandomizedSearchCV

employs stochastic techniques for discovering hyperparameters, it may discover different settings for hyperparameter values over 10 iterations of 5-fold cross validation. Therefore, we report the mode (most frequently occurring) value of hyperparameters that RandomizedSearchCV discovers.

If RandomizedSearchCV does not discover hyperparameter values different from a classifier's default values, we do not report a table of hyperparameters for that classifier. For default values of MLP, NB, DT, LR, and RF classifiers' hyperparameters please see the Scikit-learn classifier documentation [5], for XGBoost's default hyperparameter values we refer to XGBoost documentation [6], for CatBoost default hyperparameter values we refer the reader to the CatBoost documentation [7], and for Light GBM default hyperparameter values, please consult their documentation [8].

Information theft hyperparameters

Information theft without ensemble feature selection

Here, we report hyperparameter values for classifiers that yield results we report in Tables 2, 3 and 4 (Tables 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70).

Table 59 Modes of DT tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
criterion	gini
max_depth	16
max_features	sqrt
min_samples_leaf	5

Parameter values for classifier yielding best results in terms of AUPRC

Table 60 Modes of DT tuned hyperparameter values for experiments with the theft dataset

Parameter name	Value
criterion	gini
max_depth	None
max_features	sqrt
min_samples_leaf	7

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUC

[5] https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html.

[6] https://xgboost.readthedocs.io/en/latest/parameter.html.

[7] https://catboost.ai/docs/concepts/python-reference_catboostclassifier.html.

[8] https://lightgbm.readthedocs.io/en/latest/Parameters.html.

Table 61 Modes of XGBoost tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
max_depth	19
min_child_weight	1
reg_lambda	0
subsample	0.70258

Parameter values for classifier yielding best results in terms of AUPRC

Table 62 Modes of XGBoost tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
max_depth	43
min_child_weight	0.01000
reg_lambda	0
subsample	0.58748

Parameter values for classifier yielding best results in terms of AUC

Table 63 Modes of LR tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
penalty	l2
class_weight	None
C	5.95900

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUC

Table 64 Modes of LR tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
penalty	l2
class_weight	None
C	2.06300

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUC

Table 65 Modes of MLP tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
activation	tanh
alpha	0.80469
hidden_layer_sizes	[441, 538]
learning_rate	constant
solver	lbfgs

Parameter values for classifier yielding best results in terms of AUPRC

Table 66 Modes of MLP tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
activation	tanh
alpha	0.12062
hidden_layer_sizes	[480]
learning_rate	constant
solver	lbfgs

Parameter values for classifier yielding best results in terms of AUC

Table 67 Modes of RF tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
bootstrap	False
class_weight	balanced_subsample
criterion	gini
max_depth	8
max_features	log2
min_impurity_decrease	0.00016
min_samples_leaf	2
min_samples_split	2
n_estimators	155

Parameter values for classifier yielding best results in terms of AUPRC

Table 68 Modes of RF tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
bootstrap	False
class_weight	balanced_subsample
criterion	gini
max_depth	8
max_features	log2
min_impurity_decrease	0.00016
min_samples_leaf	2
min_samples_split	2
n_estimators	155

Parameter values for classifier yielding best results in terms of AUC

Table 69 Modes of XGBoost tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
n_estimators	100
min_child_weight	1
max_depth	8
learning_rate	0.30000
gamma	None

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUPRC

Table 70 Modes of XGBoost tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
n_estimators	200
min_child_weight	1
max_depth	5
learning_rate	0.30000
gamma	None

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUC

Information theft with ensemble feature selection

Here, we report hyperparameter values for classifiers that yield results we report in Tables 5 and 6 , after the application of ensemble FSTs (Tables 71, 72, 73, 74, 75).

Table 71 Modes of DT tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
criterion	gini
max_depth	16
max_features	None
min_samples_leaf	5

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUPRC

Table 72 Modes of LR tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
penalty	None
class_weight	None
C	9.35500

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUPRC

Table 73 Modes of LR tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
penalty	l2
class_weight	None
C	5.74700

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUC

Table 74 Modes of MLP tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
activation	relu
alpha	0.19409
hidden_layer_sizes	[104, 555]
learning_rate	adaptive
solver	lbfgs

Parameter values for classifier yielding best results in terms of AUC

Table 75 Modes of RF tuned hyperparameter values for experiments with the information theft dataset

Parameter name	Value
bootstrap	True
class_weight	balanced
criterion	gini
max_depth	9
max_features	log2
min_impurity_decrease	0.00000
min_samples_leaf	2
min_samples_split	5
n_estimators	174

Parameter values for classifier yielding best results in terms of AUC

Data exfiltration hyperparameters

Data exfiltration without ensemble feature selection

Here, we report hyperparameter values for classifiers that yield results we report in Tables 17, 18 and 19 (Tables 76, 77, 78, 79, 80, 81, 82 83, 84, 85, 86, 87,).

Table 76 Modes of DT tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
criterion	entropy
max_depth	16
max_features	log2
min_samples_leaf	4

Parameter values for classifier yielding best results in terms of AUPRC

Table 77 Modes of DT tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
criterion	entropy
max_depth	16
max_features	None
min_samples_leaf	8

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUC

Table 78 Modes of XGBoost tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
max_depth	48
min_child_weight	0.01000
reg_lambda	0
subsample	0.64976

Parameter values for classifier yielding best results in terms of AUPRC

Table 79 Modes of XGBoost tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
max_depth	48
min_child_weight	0.01000
reg_lambda	10
subsample	0.64976

Parameter values for classifier yielding best results in terms of AUC

Table 80 Modes of LR tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
penalty	none
class_weight	balanced
C	6.17700

Parameter values for classifier yielding best results in terms of AUPRC

Table 81 Modes of LR tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
penalty	none
class_weight	balanced
C	6.17700

parameter values for classifier yielding best results in terms of AUC

Table 82 Modes of MLP tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
activation	relu
alpha	0.31846
hidden_layer_sizes	[587, 191]
learning_rate	constant
solver	lbfgs

Parameter values for classifier yielding best results in terms of AUPRC

Table 83 Modes of MLP tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
activation	relu
alpha	0.51106
hidden_layer_sizes	[333, 164]
learning_rate	constant
solver	lbfgs

Parameter values for classifier yielding best results in terms of AUC

Table 84 Modes of RF tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
bootstrap	True
class_weight	balanced
criterion	entropy
max_depth	9
max_features	sqrt
min_impurity_decrease	0.00017
min_samples_leaf	2
min_samples_split	10
n_estimators	127

Parameter values for classifier yielding best results in terms of AUPRC

Table 85 Modes of RF tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
bootstrap	True
class_weight	balanced
criterion	entropy
max_depth	9
max_features	sqrt
min_impurity_decrease	0.00017
min_samples_leaf	2
min_samples_split	10
n_estimators	127

Parameter values for classifier yielding best results in terms of AUC

Table 86 Modes of XGBoost tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
n_estimators	100
min_child_weight	3
max_depth	7
learning_rate	0.30000
gamma	1.90000

Parameter values for classifier yielding best results in terms of AUPRC

Table 87 Modes of XGBoost tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
n_estimators	50
min_child_weight	3
max_depth	None
learning_rate	0.30000
gamma	None

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUC

Data exfiltration with ensemble feature selection

Here, we report hyperparameter values for classifiers that yield results we report in Tables 20 and 21 , after the application of ensemble FSTs (Tables 88, 89, 90, 91, 92).

Table 88 Modes of DT tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
criterion	gini
max_depth	None
max_features	auto
min_samples_leaf	6

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUPRC

Table 89 Modes of XGBoost tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
max_depth	18
min_child_weight	1
reg_lambda	0.10000
subsample	0.77026

Parameter values for classifier yielding best results in terms of AUPRC

Table 90 Modes of LR tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
penalty	none
class_weight	balanced
C	6.20100

Parameter values for classifier yielding best results in terms of AUPRC

Table 91 Modes of MLP tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
activation	relu
alpha	0.23714
hidden_layer_sizes	[590, 270]
learning_rate	constant
solver	adam

Parameter values for classifier yielding best results in terms of AUC

Table 92 Modes of MLP tuned hyperparameter values for experiments with the data exfiltration dataset

Parameter name	Value
activation	relu
alpha	0.04502
hidden_layer_sizes	[534]
learning_rate	adaptive
solver	lbfgs

Parameter values for classifier yielding best results in terms of AUPRC

Keylogging hyperparameters

Keylogging without ensemble feature selection

Here, we report hyperparameter values for classifiers that yield results we report in Tables 32, 33 and 34 (Tables 93, 94, 95, 96, 97, 98, 99, 100, 101 102, 103, 104).

Table 93 Modes of DT tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
criterion	gini
max_depth	None
max_features	sqrt
min_samples_leaf	3

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUPRC

Table 94 Modes of DT tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
criterion	entropy
max_depth	16
max_features	auto
min_samples_leaf	3

Parameter values for classifier yielding best results in terms of AUC

Table 95 Modes of XGBoost tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
max_depth	31
min_child_weight	0.01000
reg_lambda	10
subsample	0.61210

Parameter values for classifier yielding best results in terms of AUPRC

Table 96 Modes of XGBoost tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
max_depth	19
min_child_weight	0.01000
reg_lambda	0.10000
subsample	0.58197

Parameter values for classifier yielding best results in terms of AUC

Table 97 Modes of LR tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
penalty	none
class_weight	None
C	4.06500

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUPRC

Table 98 Modes of LR tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
penalty	none
class_weight	balanced
C	2.99100

Parameter values for classifier yielding best results in terms of AUC

Table 99 Modes of MLP tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
activation	relu
alpha	0.04400
hidden_layer_sizes	[115, 460]
learning_rate	constant
solver	lbfgs

Parameter values for classifier yielding best results in terms of AUPRC

Table 100 Modes of MLP tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
activation	relu
alpha	0.04400
hidden_layer_sizes	[115, 460]
learning_rate	constant
solver	lbfgs

Parameter values for classifier yielding best results in terms of AUC

Table 101 Modes of RF tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
bootstrap	True
class_weight	balanced_subsample
criterion	entropy
max_depth	9
max_features	sqrt
min_impurity_decrease	0.00000
min_samples_leaf	2
min_samples_split	9
n_estimators	141

Parameter values for classifier yielding best results in terms of AUPRC

Table 102 Modes of RF tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
bootstrap	True
class_weight	balanced_subsample
criterion	entropy
max_depth	9
max_features	sqrt
min_impurity_decrease	0.00000
min_samples_leaf	2
min_samples_split	9
n_estimators	141

Parameter values for classifier yielding best results in terms of AUC

Table 103 Modes of XGBoost tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
n_estimators	200
min_child_weight	1
max_depth	7
learning_rate	0.10000
gamma	None

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUPRC

Table 104 Modes of XGBoost tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
n_estimators	200
min_child_weight	1
max_depth	8
learning_rate	0.20000
gamma	None

"None" value indicates default value of hyperparameter is optimal; parameter values for classifier yielding best results in terms of AUC

Keylogging with ensemble feature selection

Here, we report hyperparameter values for classifiers that yield results we report in Tables 35 and 36 , after the application of ensemble FSTs (Tables 105, 106, 107, 108, 109).

Table 105 Modes of DT tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
criterion	entropy
max_depth	16
max_features	auto
min_samples_leaf	7

Parameter values for classifier yielding best results in terms of AUPRC

Table 106 Modes of XGBoost tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
max_depth	37
min_child_weight	0.01000
reg_lambda	5
subsample	0.31986

Parameter values for classifier yielding best results in terms of AUPRC

Table 107 Modes of LR tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
penalty	none
class_weight	balanced
C	3.90500

Parameter values for classifier yielding best results in terms of AUC

Table 108 Modes of MLP tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
activation	tanh
alpha	0.17176
hidden_layer_sizes	[109]
learning_rate	constant
solver	lbfgs

Parameter values for classifier yielding best results in terms of AUPRC

Table 109 Modes of MLP tuned hyperparameter values for experiments with the keylogging dataset

Parameter name	Value
activation	relu
alpha	0.10331
hidden_layer_sizes	[269, 339]
learning_rate	constant
solver	adam

Parameter values for classifier yielding best results in terms of AUC

Abbreviations
ANOVA: Analysis of variance; ARM: Association rule mining; ARP: Address resolution protocol; AUC: Area under the receiver operating characteristic curve; AUPRC: Area under the precision-recall curve; CNN: Convolutional neural network; CSV: Comma-separated values; cuDNNLSTM: CUDA deep neural network LSTM; CV: Cross-validation; DDoS: Distributed denial-of-service; DNN: Deep neural network; DNN-GRU: Deep neural network-gated recurrent unit; DoS: Denial-of-service; DT: Decision tree; ENN: Edited nearest neighbor; FAU: Florida Atlantic University; FN: False negative; FNR: False negative rate; FP: False positive; FPR: False positive rate; FST: Feature selection technique; GBDT: Gradient-boosted decision tree; GM: Geometric mean; HSD: Honestly significant difference; HTTP: Hypertext transfer protocol; ICA: Independent component analysis; ICMP: Internet control message protocol; IP: Internet protocol; IoT: Internet of Things; *k*-NN: k-Nearest neighbor; LR: Logistic regression; LSTM: Long short-term memory; LSTM-GRU: Long short-term memory-gated recurrent unit; MLP: Multi-layer perceptron; MSE: Mean square error; NB: Naive Bayes; NL: Noise level; NSF: National Science Foundation; OS: Operating system; PC: Principal component; PCAP: Packet capture; PCA: Principal component analysis; PSO: Particle swarm optimization; RF: Random forest; RNN: Recurrent neural network; ROC: Receiver operating characteristic; RUS: Random undersampling; SMOTE: Synthetic minority oversampling technique; SVM: Support vector machine; TCP: Transmission control protocol; TN: True negative; TNR: True negative rate; TP: True positive; TPR: True positive rate; ULB: Université Libre de Bruxelles; UDP: User datagram protocol; UNSW: University of New South Wales; AUC: Area under the receiver operating characteristic curve; FNR: False negative rate; GM: Geometric mean; AUC: Area under the receiver operating characteristic curve; AUPRC: Area under the precision-recall curve; DT: Decision tree; FST: Feature selection technique; ICA: Independent component analysis; IoT: Internet of Things; LR: Logistic regression; MLP: Multi-layer perceptron; NB: Naive Bayes; PCA: Principal component analysis; RF: Random forest; DT: Decision tree; FST: Feature selection technique; LR: Logistic regression; MLP: Multi-layer perceptron; NB: Naive Bayes; RF: Random forest; RUS: Random undersampling; RF: Random forest.

Acknowledgements
We would like to thank the reviewers in the Data Mining and Machine Learning Laboratory at Florida Atlantic University.

Authors' contributions
JLL searched for relevant papers and drafted the manuscript. All authors provided feedback to JLL and helped shape the work. JLL, JH, and JMP prepared the manuscript. TMK introduced this topic to JLL and helped to complete and finalize the work. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 16 September 2021 Accepted: 22 December 2021
Published online: 06 January 2022

References

1. Leevy JL, Khoshgoftaar TM, Peterson JM. Mitigating class imbalance for iot network intrusion detection: a survey. In: 2021 IEEE seventh international conference on big data computing service and applications (BigDataService). IEEE; 2021. 143–148.
2. 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. 2019;100:779–96.
3. Argus: Argus. https://openargus.org/.
4. Fu Y, Husain B, Brooks RR. Analysis of botnet counter-counter-measures. In: Proceedings of the 10th annual cyber and information security research conference, 2015;1–4.
5. Ullah F, Edwards M, Ramdhany R, Chitchyan R, Babar MA, Rashid A. Data exfiltration: A review of external attack vectors and countermeasures. Journal of Network and Computer Applications. 2018;101:18–54.
6. Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N. A survey on addressing high-class imbalance in big data. J Big Data. 2018;5(1):42.
7. Hancock JT, Khoshgoftaar TM. Catboost for big data: an interdisciplinary review. J Big Data. 2020;7(1):1–45.
8. Leevy JL, Hancock J, Khoshgoftaar TM, Seliya N. Iot reconnaissance attack classification with random undersampling and ensemble feature selection. In: 2021 IEEE 7th international conference on collaboration and internet computing (CIC). IEEE; 2021.

9. Hancock J, Khoshgoftaar TM. Medicare fraud detection using catboost. In: 2020 IEEE 21st international conference on information reuse and integration for data science (IRI). IEEE; 2020. 97–103.
10. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
11. Zuech R, Hancock J, Khoshgoftaar TM. Investigating rarity in web attacks with ensemble learners. J Big Data. 2021;8(1):1–27.
12. Rymarczyk T, Kozłowski E, Kłosowski G, Niderla K. Logistic regression for machine learning in process tomography. Sensors. 2019;19(15):3400.
13. Saritas MM, Yasar A. Performance analysis of ann and naive bayes classification algorithm for data classification. Int J Intell Syst Appl Eng. 2019;7(2):88–91.
14. Rynkiewicz J. Asymptotic statistics for multilayer perceptron with relu hidden units. Neurocomputing. 2019;342:16–23.
15. Wang H, Khoshgoftaar TM, Napolitano A. A comparative study of ensemble feature selection techniques for software defect prediction. In: 2010 Ninth international conference on machine learning and applications. IEEE; 2010. 135–140.
16. Najafabadi MM, Khoshgoftaar TM, Seliya N. Evaluating feature selection methods for network intrusion detection with kyoto data. Int J Reliabil Qual Saf Eng. 2016;23(01):1650001.
17. VMware: What is ESXi?: Bare Metal Hypervisor: Esx. https://www.vmware.com/products/esxi-and-esx.html.
18. Ostinato: Ostinato Traffic Generator for Network Engineers. https://ostinato.org/.
19. Foundation TO. Node-RED: Low-code programming for event-driven applications. https://nodered.org/.
20. OffSec: Kali Docs: Kali Linux documentation. https://www.kali.org/.
21. Canonical: enterprise open source and Linux. https://ubuntu.com/.
22. MQTT.org: MQTT—the standard for IoT messaging. https://mqtt.org/.
23. Foundation E. Eclipse mosquitto. https://mosquitto.org/.
24. Canonical: Ubuntu Phone Documentation. https://phone.docs.ubuntu.com/en/devices/.
25. Rapid7: Download metasploitable—intentionally vulnerable machine. https://information.rapid7.com/download-metasploitable-2017.html.
26. Metasploit R. Penetration testing, software, pen testing security. https://www.metasploit.com/.
27. pfSense: learn about the pfSense Project. https://www.pfsense.org/.
28. Tcpdump: TCPDUMP/LIBPCAP public repository. https://www.tcpdump.org/.
29. Koroniotis N, Moustafa N, Sitnikova E. A new network forensic framework based on deep learning for internet of things networks: a particle deep framework. Future Gener Comput Syst. 2020;110:91–106.
30. Amaizu GC, Nwakanma CI, Lee J-M, Kim D-S. Investigating network intrusion detection datasets using machine learning. In: 2020 International conference on information and communication technology convergence (ICTC). IEEE; 2020.1325–1328.
31. Malik AJ, Khan FA. A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection. Cluster Comput. 2018;21(1):667–80.
32. De Cock M, Dowsley R, Nascimento AC, Railsback D, Shen J, Todoki A. High performance logistic regression for privacy-preserving genome analysis. BMC Med Genomics. 2021;14(1):1–18.
33. Ceddia G, Martino LN, Parodi A, Secchi P, Campaner S, Masseroli M. Association rule mining to identify transcription factor interactions in genomic regions. Bioinformatics. 2020;36(4):1007–13.
34. Ahmad I, Basheri M, Iqbal MJ, Rahim A. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. IEEE Access. 2018;6:33789–95.
35. Ferrag MA, Maglaras L, Moschoyiannis S, Janicke H. Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study. J Inf Secur Appl. 2020;50:102419.
36. Lin P, Ye K, Xu C-Z. Dynamic network anomaly detection system by using deep learning techniques. In: International conference on cloud computing. Springer; 2019. 161–176.
37. Kaur G, Lashkari AH, Rahali A. Intrusion traffic detection and characterization using deep image learning. In: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE; 2020. 55–62.
38. Liaqat S, Akhunzada A, Shaikh FS, Giannetsos A, Jan MA. Sdn orchestration to combat evolving cyber threats in internet of medical things (iomt). Comput Commun. 2020;160:697–705.
39. Nakayama S, Arai S. Dnn-lstm-crf model for automatic audio chord recognition. In: Proceedings of the international conference on pattern recognition and artificial intelligence; 2018. 82–88.
40. Santos MS, Soares JP, Abreu PH, Araujo H, Santos J. Cross-validation for imbalanced datasets: avoiding overoptimistic and overfitting approaches [research frontier]. IEEE Comput Intell Mag. 2018;13(4):59–76.
41. Mulyanto M, Faisal M, Prakosa SW, Leu J-S. Effectiveness of focal loss for minority classification in network intrusion detection systems. Symmetry. 2021;13(1):4.
42. Nemoto K, Hamaguchi R, Imaizumi T, Hikosaka S. Classification of rare building change using cnn with multi-class focal loss. In: IGARSS 2018-2018 IEEE international geoscience and remote sensing symposium. IEEE; 2018. 4663–4666.
43. Ho Y, Wookey S. The real-world-weight cross-entropy loss function: modeling the costs of mislabeling. IEEE Access. 2019;8:4806–13.
44. Dhanabal L, Shantharajah S. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. Int J Adv Res Comput Commun Eng. 2015;4(6):446–52.
45. Shamsudin H, Yusof UK, Jayalakshmi A, Khalid MNA. Combining oversampling and undersampling techniques for imbalanced classification: a comparative study using credit card fraudulent transaction dataset. In: 2020 IEEE 16th international conference on control & automation (ICCA). IEEE; 2020. 803–808.
46. Ge M, Fu X, Syed N, Baig Z, Teo G, Robles-Kelly A. Deep learning-based intrusion detection for iot networks. In: 2019 IEEE 24th Pacific rim international symposium on dependable computing (PRDC). IEEE; 2019. 256–25609.

47. Varsamopoulos S, Criger B, Bertels K. Decoding small surface codes with feedforward neural networks. Quant Sci Technol. 2017;3(1):015004.
48. Soe YN, Santosa PI, Hartanto R. Ddos attack detection based on simple ann with smote for iot environment. In: 2019 Fourth international conference on informatics and computing (ICIC). IEEE; 2019. 1–5.
49. Peterson JM, Leevy JL, Khoshgoftaar TM. A review and analysis of the bot-iot dataset. In: 2021 IEEE international conference on service-oriented system engineering. IEEE; 2021. 10–17.
50. Zuech R, Hancock J, Khoshgoftaar TM. Detecting web attacks using random undersampling and ensemble learners. J Big Data. 2021;8(1):1–20.
51. Naghiloo M, Alonso J, Romito A, Lutz E, Murch K. Information gain and loss for a quantum maxwell's demon. Phys Rev Lett. 2018;121(3):030604.
52. Dong R-H, Yan H-H, Zhang Q-Y. An intrusion detection model for wireless sensor network based on information gain ratio and bagging algorithm. Int J Netw Secur. 2020;22(2):218–30.
53. Leevy JL, Khoshgoftaar TM. A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. J Big Data. 2020;7(1):1–19.
54. Leevy JL, Hancock J, Zuech R, Khoshgoftaar TM. Detecting cybersecurity attacks across different network features and learners. J Big Data. 2021;8(1):1–29.
55. Seiffert C, Khoshgoftaar TM, Van Hulse J, Napolitano A. Mining data with rare events: a case study. In: 19th IEEE international conference on tools with artificial intelligence (ICTAI 2007). IEEE; 2007;2, 132–139.
56. Hancock JT, Khoshgoftaar TM. Gradient boosted decision tree algorithms for medicare fraud detection. SN Comput Sci. 2021;2(4):1–12.
57. Gupta A, Nagarajan V, Ravi R. Approximation algorithms for optimal decision trees and adaptive tsp problems. Math Oper Res. 2017;42(3):876–96.
58. González S, García S, Del Ser J, Rokach L, Herrera F. A practical tutorial on bagging and boosting based ensembles for machine learning: algorithms, software tools, performance study, practical perspectives and opportunities. Inf Fusion. 2020;64:205–37.
59. Lobo JM, Jiménez-Valverde A, Real R. Auc: a misleading measure of the performance of predictive distribution models. Glob Ecol Biogeogr. 2008;17(2):145–51.
60. Saito T, Rehmsmeier M. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PloS One. 2015;10(3):0118432.
61. Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th international joint conference on artificial intelligence-Volume 2. Morgan Kaufmann Publishers Inc.; 1995. 1137–1143.
62. Suzuki S, Yamashita T, Sakama T, Arita T, Yagi N, Otsuka T, Semba H, Kano H, Matsuno S, Kato Y, et al. Comparison of risk models for mortality and cardiovascular events between machine learning and conventional logistic regression analysis. PLoS One. 2019;14(9):0221911.
63. Van Hulse J, Khoshgoftaar TM, Napolitano A. An empirical comparison of repetitive undersampling techniques. In: 2009 IEEE international conference on information reuse and integration. IEEE; 2009. 29–34.
64. Iversen GR, Wildt AR, Norpoth H, Norpoth HP. Analysis of variance. Sage, 1987.
65. Tukey JW. Comparing individual means in the analysis of variance. Biometrics.1949; 99–114.

Publisher's Note