Detecting Information Theft Attacks in the Bot-IoT Dataset

Joffrey L. Leevy *, John Hancock*, Taghi M. Khoshgoftaar*, and Jared Peterson*
*Florida Atlantic University
Email: jleevy2017@fau.edu, jhancoc4@fau.edu, khoshgof@fau.edu, jpeterson2019@fau.edu

***Abstract*—There are growing security risks tied to the recent proliferation of *Internet of Things* (IoT) devices. Due to this fact, datasets such as Bot-IoT were designed to train machine learning classifiers on network intrusion detection in IoT networks. In this research, we use Bot-IoT to build a predictive model for detecting information theft attacks. Our contribution is defined by the unique approach of using eight classifiers and two performance metrics to detect information theft traffic. Also, to the best of our knowledge, the Bot-IoT Information Theft category of attacks has never been the focus of a research paper. Our group of classifiers is a diverse range of four ensembles (CatBoost, LightGBM, XGBoost, and Random Forest) and four non-ensembles (Decision Tree, Logistic Regression, Naive Bayes, and a *Multilayer Perceptron* (MLP)). The metrics used to evaluate the classifiers are *Area Under the Receiver Operating Characteristic Curve* (AUC) and *Area Under the Precision-Recall Curve* (AUPRC). Through cross-validation, we train and test Bot-IoT instances (only normal and information theft traffic) to evaluate the best classifier(s). According to our results, the ensemble classifiers, particularly CatBoost, LightGBM, and XGBoost, are the top-performing models.**

***Index Terms*—IoT, Bot-IoT, intrusion detection, information theft, ensemble, machine learning**

1. Introduction

The *Internet of Things* (IoT) is an interconnected network of devices with limited computing capability [1]. There has been an explosive growth in the use of these smart devices in recent years, along with an escalating security risk. Microsoft reported a 35% increase in IoT cyber-attacks between the second half of 2019 and the first half of 2020 [2]. Equally interesting is a prediction by Forbes Magazine that there will be 35 billion smart devices online in 2021 and 75 billion in 2025 [3].

Several datasets have been published for the purpose of training machine learning classifiers to detect malicious network traffic. Among the more modern datasets for network intrusion detection is Bot-IoT [4]. This dataset, created in 2018 by the *University of New South Wales* (UNSW), was designed to be a realistic representation of botnet attacks on IoT devices. Bot-IoT consists of various attack categories: *Denial-of-Service* (DoS), *Distributed Denial-of-Service* (DDoS), Reconnaissance, and Information Theft. It contains 29 features, or independent variables, and 73,370,443 instances that are generated from a repository of *comma-separated values* (CSV) files by the Argus network security tool [5].

UNSW provides a variant of Bot-IoT that contains about 5% of the original instances. This subset has 3,668,522 instances and 43 features. To facilitate the research in our paper, we chose the variant over the original dataset. Table 1 shows categories and subcategories of network traffic for the Bot-IoT subset. Hereinafter, we refer to this subset as the 5% dataset.

TABLE 1: Network traffic instances for the 5% dataset

Category	Subcategory	No. of Instances
DoS/DDoS	TCP	1,593,180
	UDP	1,981,230
	HTTP	2,474
Reconnaissance	OS Fingerprinting	17,914
	Service Scanning	73,168
Information Theft	Keylogging	73
	Data Exfiltration	6
Normal	Normal	477

Our research goal is to build a reliable predictive model capable of detecting instances of information theft attacks. Therefore, we focus on the 477 Normal instances and 79 Information Theft instances shown in Table 1. The Keylogging [6] subcategory of Information Theft refers to the interception of information sent from input devices, while the Data Exfiltration [7] subcategory of Information Theft broadly refers to the unauthorized transfer of data from a computer. In terms of Normal and Information Theft traffic, the subgroup of 556 instances has a slight class imbalance. Class imbalance occurs when there is an uneven distribution between the majority (477 instances) and minority (79 instances) classes.

In classification tasks, machine learning algorithms usually perform better than traditional statistical methods [8]–[10]. For our experimentation, we use four ensemble classifiers (CatBoost [11], LightGBM [12], XGBoost [13], and Random Forest [14]) and four non-ensemble classifiers (Decision Tree [15], Logistic Regression [16], Naive Bayes [17], and a *Multilayer Perceptron* (MLP) [18]). To gauge the performance of these classifiers, the *Area Under the Receiver Operating Characteristic Curve* (AUC) and *Area under the Precision-Recall Curve* (AUPRC) metrics are used. As far as we are aware, the uniqueness of our work is reflected through the use of eight classifiers and two performance metrics to detect instances of theft attack traffic. This classifier/performance metric combination strengthens the credibility of our results. In addition, none of the current Bot-IoT works focus exclusively on the detection of instances from the Information Theft

category.

The remainder of this paper is organized as follows: Section 2 provides an overview of related Bot-IoT literature; Section 3 details the preprocessing activities involved; Section 4 describes the classifiers, training and testing procedure, and performance metrics; Section 5 presents and discusses our results; and Section 6 concludes with a brief summary and suggestions for future work.

2. 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 in the 5% dataset.

Koroniotis et al. [19] proposed a network forensics model that authenticates network data flows and uses a *Deep Neural Network* (DNN) [20] based on *Particle Swarm Optimization* (PSO) [21] 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 [22] 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 [19]:

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

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 [19]:

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

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 (Naive Bayes, MLP, *Association Rule Mining* (ARM) [23], Decision Tree, *Support Vector Machine* (SVM) [24], *Recurrent Neural Network* (RNN) [25], and *Long Short-Term Memory* (LSTM) [26]). 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.

Ge et al. [27] trained a feedforward neural network [28] 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.

Using a *Convolutional Neural Network* (CNN) [29] and *CUDA Deep Neural Network LSTM* (cuDNNLSTM) [1] hybrid, Liaqat et al. [30] set out to prove that their proposed model could outperform a *Deep Neural Network-Gated Recurrent Unit* (DNN-GRU) [31] hybrid, as well as a *Long Short-Term Memory-Gated Recurrent Unit* (LSTM-GRU) [31] 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 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 [32], [33]. 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. [34] proposed a cost-sensitive neural network based on focal loss [35]. The cross-entropy loss function [36], 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 [37], 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) [38] 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.

Finally, to detect DDoS attacks, Soe et al. [39] 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

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

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 Decision Tree classifier is used, i.e., reliance on only one classifier.

3. Data Preprocessing

Data cleaning was the first step of our preprocessing operation. As discussed in the next three paragraphs, there are six features in Bot-IoT that do not provide generalizable information [40].

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. [4] but removing *seq* was less so, as *seq* has been highly utilized in many studies. 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. Due to space limitations, we are unable to provide more details about the Argus network security tool.

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 for two reasons. The first is the nature of IP addresses, specifically private IP addresses, which 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. The second reason was due to the limitations of the environment in which the dataset was created. When analyzing the data, we found that 100% of all attack instances have a private IP for both *saddr* and *daddr*, but only 35% of all non-attack instances have a private IP for both *saddr* and *daddr*. This leads to a problem where 65% of non-attack instances can be identified because their *saddr* and *daddr* features display a public IP.

After the six features were dropped, the dataset was reduced to 37 features. Following this step, the *flgs_number*, *fproto_number*, *sport*, *dport*, and *state_number* categorical features were one-hot encoded. Finally, feature scaling was performed to provide a [0,1] normalized range for all numerical features.

4. Classification Modeling

A. Classifiers

This study involves four ensemble classifiers (CatBoost, LightGBM, XGBoost, and Random Forest) and four non-ensemble classifiers (Decision Tree, Logistic Regression, Naive Bayes, and an MLP). These classifiers belong to various machine learning families of algorithms and are widely considered to be reliable [41]. CatBoost, LightGBM, and XGBoost were implemented with their respective self-named Python libraries, while the other classifiers were implemented with Scikit-learn [2].

CatBoost, LightGBM, and XGBoost are *Gradient-Boosted Decision Tree*s (GBDTs) [42], which are ensembles of sequentially trained trees. An ensemble classifier [43], [44] combines weak algorithms, or instances of an algorithm, into a strong learner. CatBoost relies on Ordered Boosting to order the instances used for fitting Decision Trees. LightGBM 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 learning [45] to support pruning and merging tasks. Random Forest is also an ensemble of decision trees, but unlike the GBDTs, it uses a bagging technique [46].

Decision Tree 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. Logistic regression uses a sigmoidal function to generate probability values. Predictions for class membership are centered on a specified probability threshold. Naive Bayes 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.

B. Training and Testing

As stated previously, our dataset contains 556 instances (477 from the Bot-IoT Normal category and 79 from the Bot-IoT Information Theft category). For all classifiers, we perform hyperparameter tuning. Due to space limitations, we are unable to provide more details on the selection of the optimum parameters. The train-test process is implemented

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

TABLE 2: Dataset classification results

Classifier	AUC	AUPRC
CatBoost	0.98346	0.99268
Decision Tree	0.96162	0.97427
LightGBM	**0.98785**	**0.99463**
Logistic Regression	0.97378	0.98233
Naive Bayes	0.95598	0.82905
MLP	0.97389	0.99106
Random Forest	0.97181	0.98493
XGBoost	0.97758	0.99254

TABLE 3: One-factor ANOVA for AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Classifier	7	0.04	0.01	8.51	0.0000
Residuals	392	0.25	0.00		

via 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 [47]. Because we randomly shuffle instances before cross-validation, certain algorithms such as Logistic Regression may yield different results when the order of instances is changed [48]. One way of addressing the undesirable effect of this randomness is by performing several iterations, as we have done [49]. Note that each value shown in Table 2 is an average of the 50 performance scores.

In our work, we employ more than one performance metric (AUC and AUPRC) to better assess the challenge of evaluating machine learning models for detecting IoT information theft. 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 [50] 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 [51].

TABLE 4: One-factor ANOVA for AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Classifier	7	1.11	0.16	283.29	0.0000
Residuals	392	0.22	0.00		

TABLE 5: HSD groups for AUC

Group a consists of: LightGBM, CatBoost
Group ab consists of: XGBoost
Group abc consists of: MLP, Logistic Regression, Random Forest
Group bc consists of: Decision Tree
Group c consists of: Naive Bayes

TABLE 6: HSD groups for AUPRC

Group a consists of: LightGBM, CatBoost, XGBoost
Group ab consists of: MLP, Random Forest, Logistic Regression
Group b consists of: Decision Tree
Group c consists of: Naive Bayes

5. Results and Discussion

Mean AUC and AUPRC scores for the eight models are shown in Table 2. For each performance metric in the table, the highest score among the classifiers has been boldfaced. LightGBM is the top performer with scores of 0.99 for both AUC and AUPRC, and Naive Bayes sits at the bottom with AUC and AUPRC scores of 0.96 and 0.83, respectively. Across the board, performance scores generally appear robust, thus indicating that all eight models may be acceptable for detecting information theft instances in Bot-IoT. These solid scores also suggest that it is not necessary to use specialized techniques to address the low level of class imbalance in this study.

To understand the statistical significance of the performance scores in Table 2, we perform one-factor *ANalysis Of VAriance* (ANOVA) tests. ANOVA establishes whether there is a significant difference between group means [52]. A 95% (α = 0.05) confidence level is used for our ANOVA tests. The results are shown in Tables 3 and 4, 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 single factor for our ANOVA tests is the group of classifiers (eight levels). As this factor has a p-value of less than 0.05 in Tables 3 and 4, this indicates that the classifiers are a significant factor for both AUC and AUPRC metrics. Therefore, we perform Tukey's *Honestly Significant Difference* (HSD) tests [53] to determine which groups of classifiers are significantly different from each other. Letter groups assigned via the Tukey method indicate similarity or significant differences in performance results within a factor. The best-performing group is assigned the letter 'a' and the worst-performing group, the letter 'd'.

Our HSD test results are shown in Tables 5 and 6. The best AUC group, 'a', includes LightGBM and CatBoost, while the best AUPRC group, 'a', includes LightGBM, CatBoost, and XGBoost. The ensemble classifier, Random Forest, is in the 'abc' group for AUC and the 'ab' group for AUPRC. For both metrics, Naive Bayes is the only classifier in the worst-performing group, 'c'. The box plots shown in Figures 1 and 2 provide visualization of the results. In Figure 1, the AUC medians are highest for LightGBM, CatBoost, and XGBoost and lowest for Naive Bayes. Furthermore, the box for Naive Bayes does not overlap with box medians from the other classifiers. From these plots, it is easy to see that LightGBM, CatBoost, and XGBoost are the top performers and Naive

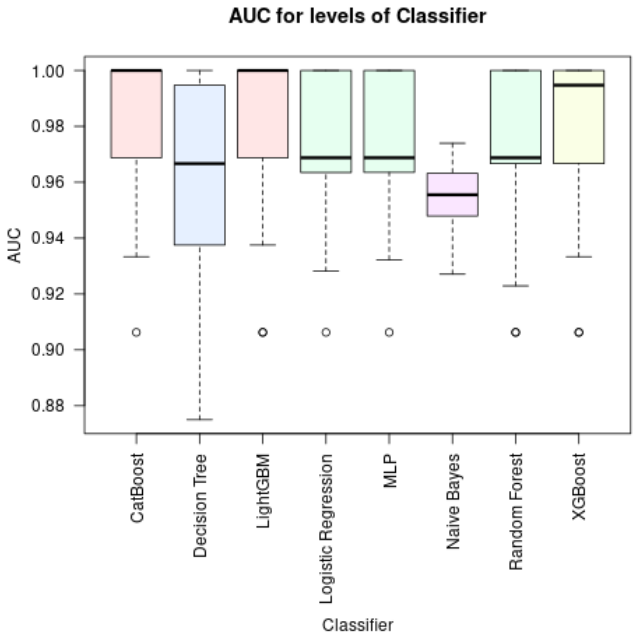

Fig. 1: AUC box plots for classifier performance

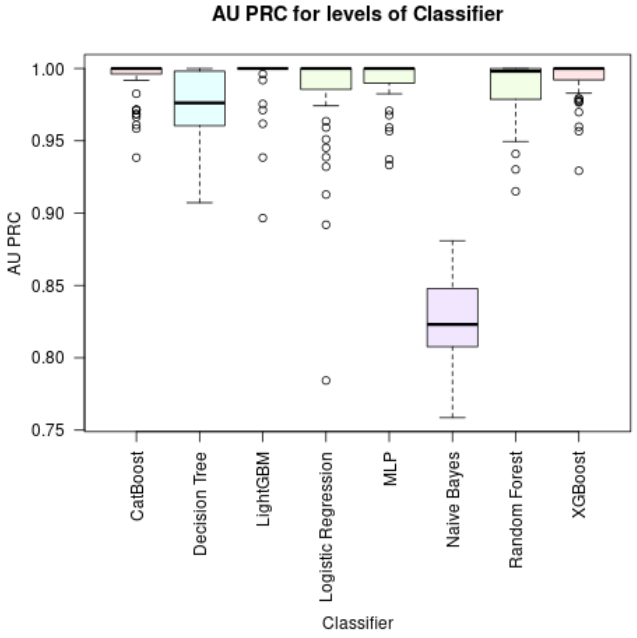

Fig. 2: AUPRC box plots for classifier performance

Bayes is the worst. In Figure 2, the AUPRC medians are highest for LightGBM, CatBoost, XGBoost, Logistic Regression, and MLP. The Naive Bayes box, along with its median, occupies the lowest position. From these plots, we observe that LightGBM, CatBoost, and XGBoost are among the top performers, while Naive Bayes is the worst.

Based on the Tukey's HSD tests and box plots, it is evident that the ensemble classifiers outperformed the non-ensemble classifiers. This finding indicates that CatBoost, LightGBM, XGBoost, and even perhaps Random Forest, should be the predictive models of choice for detecting information theft attacks in Bot-IoT.

6. Conclusion

The recent surge in the number of IoT devices has been matched by a growing security risk. Bot-IoT, an IoT-centric dataset for network intrusion detection, is a tool for countering such risk. The aim of our research is to build a predictive model from Bot-IoT that efficiently identifies instances of information theft attacks. To accomplish this, we use 477 instances from the Bot-IoT Normal category and 79 from the Bot-IoT Information Theft category. The eight classifiers trained and tested via cross-validation are generally viewed as reliable. Based on our AUC and AUPRC results, the ensemble classifiers, especially CatBoost, LightGBM, and XGBoost, are the best models for detecting IoT information theft traffic.

Future work will evaluate other classifiers trained on the same normal and information theft instances of Bot-IoT. There is also an opportunity to investigate classifier performance, with regard to information theft detection, on other IoT intrusion detection datasets.

References

[1] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[2] Microsoft, "Microsoft report shows increasing sophistication of cyber threats," https://blogs.microsoft.com/on-the-issues/2020/09/29/microsoft-digital-defense-report-cyber-threats/.

[3] Forbes, "5 iot trends to watch in 2021," https://www.forbes.com/sites/danielnewman/2020/11/25/5-iot-trends-to-watch-in-2021/?sh=4643ae1201b3.

[4] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[5] Argus, "Argus," https://openargus.org/.

[6] Y. Fu, B. Husain, and R. R. Brooks, "Analysis of botnet counter-countermeasures," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, 2015, pp. 1–4.

[7] F. Ullah, M. Edwards, R. Ramdhany, R. Chitchyan, M. A. Babar, and A. Rashid, "Data exfiltration: A review of external attack vectors and countermeasures," *Journal of Network and Computer Applications*, vol. 101, pp. 18–54, 2018.

[8] J. Hancock and T. M. Khoshgoftaar, "Performance of catboost and xgboost in medicare fraud detection," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020, pp. 572–579.

[9] J. L. Leevy, J. Hancock, R. Zuech, and T. M. Khoshgoftaar, "Detecting cybersecurity attacks across different network features and learners," *Journal of Big Data*, vol. 8, no. 1, pp. 1–29, 2021.

[10] J. L. Leevy, J. Hancock, R. Zuech, and T. M. Khoshgoftaar, "Detecting cybersecurity attacks using different network features with lightgbm and xgboost learners," in *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*. IEEE, 2020, pp. 190–197.

[11] J. T. Hancock and T. M. Khoshgoftaar, "Catboost for big data: an interdisciplinary review," *Journal of big data*, vol. 7, no. 1, pp. 1–45, 2020.

[12] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, 2017, pp. 3146–3154.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[14] T. M. Khoshgoftaar, M. Golawala, and J. Van Hulse, "An empirical study of learning from imbalanced data using random forest," in *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, vol. 2. IEEE, 2007, pp. 310–317.

[15] C. Kemp, C. Calvert, and T. M. Khoshgoftaar, "Detection methods of slow read dos using full packet capture data," in *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 2020, pp. 9–16.

[16] T. Rymarczyk, E. Kozłowski, G. Kłosowski, and K. Niderla, "Logistic regression for machine learning in process tomography," *Sensors*, vol. 19, no. 15, p. 3400, 2019.

[17] M. M. Saritas and A. Yasar, "Performance analysis of ann and naive bayes classification algorithm for data classification," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 7, no. 2, pp. 88–91, 2019.

[18] J. Rynkiewicz, "Asymptotic statistics for multilayer perceptron with relu hidden units," *Neurocomputing*, vol. 342, pp. 16–23, 2019.

[19] N. Koroniotis, N. Moustafa, and E. Sitnikova, "A new network forensic framework based on deep learning for internet of things networks: A particle deep framework," *Future Generation Computer Systems*, vol. 110, pp. 91–106, 2020.

[20] G. C. Amaizu, C. I. Nwakanma, J.-M. Lee, and D.-S. Kim, "Investigating network intrusion detection datasets using machine learning," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2020, pp. 1325–1328.

[21] A. J. Malik and F. A. Khan, "A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection," *Cluster Computing*, vol. 21, no. 1, pp. 667–680, 2018.

[22] M. De Cock, R. Dowsley, A. C. Nascimento, D. Railsback, J. Shen, and A. Todoki, "High performance logistic regression for privacy-preserving genome analysis," *BMC Medical Genomics*, vol. 14, no. 1, pp. 1–18, 2021.

[23] G. Ceddia, L. N. Martino, A. Parodi, P. Secchi, S. Campaner, and M. Masseroli, "Association rule mining to identify transcription factor interactions in genomic regions," *Bioinformatics*, vol. 36, no. 4, pp. 1007–1013, 2020.

[24] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE access*, vol. 6, pp. 33 789–33 795, 2018.

[25] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[26] P. Lin, K. Ye, and C.-Z. Xu, "Dynamic network anomaly detection system by using deep learning techniques," in *International Conference on Cloud Computing*. Springer, 2019, pp. 161–176.

[27] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep learning-based intrusion detection for iot networks," in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2019, pp. 256–25 609.

[28] S. Varsamopoulos, B. Criger, and K. Bertels, "Decoding small surface codes with feedforward neural networks," *Quantum Science and Technology*, vol. 3, no. 1, p. 015004, 2017.

[29] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, pp. 1–54, 2019.

[30] S. Liaqat, A. Akhunzada, F. S. Shaikh, A. Giannetsos, and M. A. Jan, "Sdn orchestration to combat evolving cyber threats in internet of medical things (iomt)," *Computer Communications*, vol. 160, pp. 697–705, 2020.

[31] S. Nakayama and S. Arai, "Dnn-lstm-crf model for automatic audio chord recognition," in *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*, 2018, pp. 82–88.

[32] M. S. Santos, J. P. Soares, P. H. Abreu, H. Araujo, and J. Santos, "Cross-validation for imbalanced datasets: avoiding overoptimistic and overfitting approaches [research frontier]," *ieee ComputatioNal iNtelligeNCe magaziNe*, vol. 13, no. 4, pp. 59–76, 2018.

[33] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data*, vol. 5, no. 1, p. 42, 2018.

[34] M. Mulyanto, M. Faisal, S. W. Prakosa, and J.-S. Leu, "Effectiveness of focal loss for minority classification in network intrusion detection systems," *Symmetry*, vol. 13, no. 1, p. 4, 2021.

[35] K. Nemoto, R. Hamaguchi, T. Imaizumi, and S. Hikosaka, "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, pp. 4663–4666.

[36] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2019.

[37] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International journal of advanced research in computer and communication engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[38] H. Shamsudin, U. K. Yusof, A. Jayalakshmi, and M. N. A. Khalid, "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, pp. 803–808.

[39] Y. N. Soe, P. I. Santosa, and R. Hartanto, "Ddos attack detection based on simple ann with smote for iot environment," in *2019 Fourth International Conference on Informatics and Computing (ICIC)*. IEEE, 2019, pp. 1–5.

[40] J. M. Peterson, J. L. Leevy, and T. M. Khoshgoftaar, "A review and analysis of the bot-iot dataset," in *2021 IEEE International Conference on Service-Oriented System Engineering*. IEEE, 2021, pp. 10–17.

[41] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Mining data with rare events: a case study," in *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, vol. 2. IEEE, 2007, pp. 132–139.

[42] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[43] R. Zuech, J. Hancock, and T. M. Khoshgoftaar, "Detecting web attacks using random undersampling and ensemble learners," *Journal of Big Data*, vol. 8, no. 1, pp. 1–20, 2021.

[44] R. Zuech, J. Hancock, and T. M. Khoshgoftaar, "Investigating rarity in web attacks with ensemble learners," *Journal of Big Data*, vol. 8, no. 1, pp. 1–27, 2021.

[45] A. Gupta, V. Nagarajan, and R. Ravi, "Approximation algorithms for optimal decision trees and adaptive tsp problems," *Mathematics of Operations Research*, vol. 42, no. 3, pp. 876–896, 2017.

[46] S. González, S. García, J. Del Ser, L. Rokach, and F. Herrera, "A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities," *Information Fusion*, vol. 64, pp. 205–237, 2020.

[47] J. T. Hancock and T. M. Khoshgoftaar, "Gradient boosted decision tree algorithms for medicare fraud detection," *SN Computer Science*, vol. 2, no. 4, pp. 1–12, 2021.

[48] S. Suzuki, T. Yamashita, T. Sakama, T. Arita, N. Yagi, T. Otsuka, H. Semba, H. Kano, S. Matsuno, Y. Kato *et al.*, "Comparison of risk models for mortality and cardiovascular events between machine learning and conventional logistic regression analysis," *PLoS One*, vol. 14, no. 9, p. e0221911, 2019.

[49] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "An empirical comparison of repetitive undersampling techniques," in *2009 IEEE International Conference on Information Reuse & Integration*. IEEE, 2009, pp. 29–34.

[50] N. Seliya, T. M. Khoshgoftaar, and J. Van Hulse, "A study on the relationships of classifier performance metrics," in *2009 21st IEEE international conference on tools with artificial intelligence*. IEEE, 2009, pp. 59–66.

[51] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PloS one*, vol. 10, no. 3, p. e0118432, 2015.

[52] G. R. Iversen, A. R. Wildt, H. Norpoth, and H. P. Norpoth, *Analysis of variance*. Sage, 1987, no. 1.

[53] J. W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, pp. 99–114, 1949.