Journal of Computer Science

Review

Toward an Online DoS/DDoS Classification: An Empirical Study for Network Intrusion Detection Systems

[1]Tran Hoang Hai, [1]Nguyen Trong Khiem and [2]Nguyen Huu Phuc

[1]School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam
[2]FPT Software, FPT Corporation, Hanoi, Vietnam

Article history
Received: 15-12-2020
Revised: 13-03-2021
Accepted: 16-03-2021

Corresponding Author:
Tran Hoang Hai
School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam
Email: hai.tranhoang@hust.edu.vn

Abstract: In recent years, Distributed Denial of Services (DDoS) attacks have caused significant losses to industry and government due to an increasing number of devices connected to the Internet. These devices use services-over-Internet more frequently with services characterized and provided seamlessly by 5G, Cloud and Edge Computing. According to Cisco Annual Internet Report, the frequency of DoS/DDoS attacks has increased more than 2.5 times over the last 3 years and the average size of attacks is increasing steadily and approaching 1 Gbps. Therefore, there are cyber threats continuing to grow even with the development of new protection technologies. Our work is strongly motivated from with the goal to study and evaluate four Machine Learning models toward development of an Online Network Intrusion Detection System (N-IDS). This article studies on the application on three feature selection algorithms combined with four machine learning models applied to N-IDS. We have implemented performance evaluation our proposed model on three up-to-date DoS/DDoS datasets. We have shown that Feature Importance and K-Nearest Neighbors' algorithm (KNN) can give better results in all benchmark datasets than previous work and the empirical results of all four machine learning models and three feature selection algorithms are also presented in detail.

Keywords: DDoS, Network Security, Intrusion Detection, Attack Classification

Introduction

In recent years, Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are among the most common attacks in cyber-attacks (Lima Filho *et al*., 2019). Internet provides an open environment in which any host can communicate with others. There is an increasing number of devices/mobiles connected to the Internet and using services-over-Internet due to the fast development of Ubiquitous Computing, Internet of Things (IoT) model with services characterized and provided seamlessly by 5G, Edge Computing. The main objective of an attacker is to take down services of the websites, Intranet's enterprises or prevent users using specific service over Internet. According to (Cisco Annual Internet Report, 2018-2023), the frequency of DoS/DDoS attacks has increased more than 2.5 times over the last 3 years and the average size of attacks is increasing steadily and approaching 1 Gbps. These attacks are strong enough to take most organizations completely offline. Denial of Service DDoS attacks can be performed using a botnet network controlled by the attacker. The owner of the compromised computer system is unaware the botnet is installed in their computers or they are part of a botnet network. With increasing number of Internet users, DDoS attack has become the second most significant threat after virus infection to the Internet users (Gupta *et al*., 2009). Moreover, the individual attack can launch the attack easy by open source tools quite easy on Internet. Gupta *et al*. (2009; 2010), the authors classify DDoS attacks into two broad categories by flooding attacks and logical attacks. Flooding attacks create huge of traffic at the victim side which makes the target computer incapable of handling request from the legitimate users. There are several types of flooding attacks such as SYN Flooding, ICMP, UDP Flooding,

etc. In logical attack, the attackers exploit known software bugs to implement such as Ping of Death, Teardrop and Land attack. Network Intrusion Detection System (N-IDS) plays an extremely important role in security management which can support network administrators to detect such type of DoS/DDoS about unusual behaviors where a traffic flow might be an attack or normal traffic flow. Currently, network administrators apply some solutions such as firewalls to prevent some unwanted traffics, but the network manager have to detection by its own technique. In the traditional rule-based N-IDS, the rules are usually pre-defined by the security experts and need to be updated regularly such as Snort, Suricata (Karthikeyan and Indra, 2010; Turner *et al*., 2016). The advantage of rule-based N-IDS is that we can detect specific attacks in detail to give better accuracy and reduce false alarms. However, with the increasing of traffic flow, it is difficult for network experts to follow the system (Srivastava *et al*., 2011). Therefore, we propose a smart N-IDS which can capture network traffic and able to analyze and detect network anomalies automatically. With the rapid development of machine learning models, several methods have been proposed to build a knowledge system on the IDS system (Gupta *et al*., 2009; 2010; Karthikeyan and Indra, 2010), where abnormal traffic can be detected and prevented automatically. Another type of N-IDS based on statistical analysis which analyze statistical behavior of users to find abnormal behaviors based on assumption that malicious traffic differs from typical user behavior traffic (Chellam *et al*., 2018). In recent years, both industry and academia make a huge effort to address this problem. There are several approaches, ranging from filtering-based approaches (Savage *et al*., 2000; Song and Perrig, 2001; Argyraki and Cheriton, 2005; Mahajan *et al*., 2002; Ioannidis and Bellovin, 2002; Liu *et al*., 2008), capability-based approaches (Yaar *et al*., 2004; Yang *et al*., 2008; Liu *et al*., 2010; 2016). We believe that a knowledge system which inherit latest development of machine learning models to combat the risks is extremely important (Panja *et al*., 2014; Zwass, 2018; Fuchsberger, 2005). Some related works based on statistical methods (Larranaga *et al*., 2013) and Bayes algorithm (Seraphim *et al*., 2018) are typical representative algorithms in this field. An expert system is currently the most feasible solution which uses artificial intelligence to solve problems in a field that requires human expertise. The application of machine learning algorithms is a breakthrough that provide us an efficient tool to apply N-IDS in practice and can be found in detail in (Khraisat *et al*., 2019; Dali *et al*., 2015). Moreover, Deep learning is a subset of machine learning that outperforms the traditional machine learning by learning to represent the data as a nested hierarchy of concepts. A general survey of Anomaly Detection using Deep Learning can be found in (Chalapathy and Chawla, 2019). The rest of paper is organized as follows. Section 2 gives an overview of related works and application of machine learnings models to N-IDS and its related benchmark datasets. Section 3 presents our proposed model for evaluation of machine learning models with several feature selection algorithms. Section 4 introduces performance comparison and finally, conclusions are given in section 5.

Related Work

Machine Learning Models to Network Anomaly Detection

In this study, we choose four popular machine learning algorithms which are k-Nearest Neighbors' algorithm (KNN) (Altman, 1992), Adaptive Boosting (AdaBoost) (Gandhi, 2018), Random Decision Forests (RandomForest) (Ho, 1995) and Support Vector Machine (Cortes and Vapnik, 1995) for our model. The motivation is that those algorithms are recent works on applying machine learning to N-IDS and they provide better results and lower processing time compared to others (Khraisat *et al*., 2019; Dali *et al*., 2015). In pattern recognition, the k-Nearest Neighbors' algorithm (k-NN) is a non-parametric method proposed by Thomas Cover used for classification and regression (Altman, 1992). Among all these data mining techniques, KNN was used prominently due to its better accuracy and detection rates. Wang *et al*. (2009), the authors use attribute normalization to improve the performance of intrusion detection on three methods, KNN, Principal Component Analysis (PCA) (Jolliffe, 1986) and SVM. KDD Cup 1999 dataset is used to evaluate the normalization schemes and the detection methods (UCI, 1999). Panda *et al*. (2012), the authors proposed a 2-class classification strategy on an early version of NSL-KDD dataset with 10-fold cross validation method. Their proposed model produced a high detection rate and low false alarm rate between normal and anomaly traffic. Jamshidi and Nezamabadi (2013), the authors introduced a nonlinear valuation function based on lattice based nearest neighbor classifier to tune the performance of the intrusion detection and was evaluated by an old KDD Cup'99 dataset. Tharwat *et al*. (2013), the authors designed and developed three different classifiers based on KNN classifier's concept for facial age estimation to achieve high efficiency. Rao and Swathi (2017) adapted two fast KNN classification algorithms i.e., Indexed Partial Distance Search k-Nearest Neighbor (IKPDS), Partial Distance Search k-Nearest Neighbor (KPDS) and comparing with traditional KNN classification for Network Intrusion Detection on NSL-KDD dataset 2017

(NSL-KDD, 2009). Benaddi *et al*. (2018), the authors propose to use PCA-fuzzy Clustering-KNN method which ensemble of Analysis of Principal Component and Fuzzy Clustering with K-Nearest Neighbor feature selection technics to detect anomalies.

Adaptive Boosting (AdaBoost) is a machine learning meta-algorithm proposed by Yoav Freund and Robert Schapire who won the 2003 Gödel Prize for their work (Zhang *et al*., 2005). AdaBoost is classified as boosting class (or sometimes referred as ensemble learning approach) because it aims to convert weak classification algorithms, correct previous algorithm errors then finally get a strong classifier. Shahraki *et al*. (2020), the authors investigate the feasibility of N-IDS by means of the most famous version of the boosting algorithms, including Real AdaBoost (Schapire and Singer, 1999), Gentle AdaBoost (Friedman *et al*., 2000) and Modest AdaBoost (Vezhnevets and Vezhnevets, 2005) on five public IDS datasets.

Random Forests (RF) is an ensemble learning method for classification, regression which construct a multitude of decision trees at training time and outputting which is classification or regression of the individual trees (Ho, 1998; Biau, 2012). Following (Resende and Drummond, 2018), RD uses multiple decision trees for layering. The algorithm assumes that if a sample is layered by multiple decision trees, whichever layer is chosen by most trees, then this sample will be assigned to that class. Efron (1979), several authors show that RF model applied in N-IDS is efficient with low false alarm rate and high detection rate. For more accuracy, RF uses a process which is called Bootstrapping. Bootstrapping is a statistical resampling technique that involves random sampling of a dataset with replacement (Peng *et al*., 2002). In addition, to make sure the decision trees are different, RF will randomly skip a few questions when building a decision tree. In this case, if the best question is not selected, the next question will be selected to build the tree. This process is called attribute sampling.

Support Vector Machine (SVM) is an efficient classification technique in a wide variety of problems (Thai *et al*., 2012) which often provides considerable improvement over competing methods. Winter *et al*. (2011), the authors proposed a lightweight IDS that uses a one-class SVM to analyze incoming net-flows for analysis. Goeschel (2016), the authors proposed to combine a linear SVM, decision trees and Naïve Bayes to reduce the number of false alarms of the IDS and evaluation model on the old KDD Cup'99 dataset (UCI, (1999). Lee *et al*. (2005), the authors proposed an IDS model which consists of a one-class SVM for anomaly detection during an initial analysis, a multi-class SVM for traffic classification in the four classes (i.e., Denial-of-Service, Remote to local, User to root and Probing attacks) and a final clustering process. Khan *et al*. (2007), the authors proposed a new approach for enhancing the training process of SVM when dealing with large training datasets. This work combines the use of SVM and clustering analysis to reduce the number of instances used during the computation of the support vector margin, which, in turn, reduces the training time without affecting the results. Sahu *et al*. (2019), the authors used an ensemble approach of supervised (SVM) and unsupervised (K-Means) to detect the anomaly patterns which provides more than 99% on three benchmarked datasets.

Feature Selection

Our work is strongly motivated by (Lima Filho *et al*., 2019) where the model does not process on all features of a network traffic flow which implement feature selection before training phase. Feature selection is an important step in the pattern recognition process and consists of defining the smallest possible set of variables capable of efficiently describing a set of classes (Ganapathy *et al*., 2013). According to (Miao and Niu, 2016), due to presence of noisy, redundant and irrelevant dimensions of large-scale data which can not only make learning algorithms very slow and even degenerate the performance of learning tasks. Benefits of feature selection are preventing overfitting (less redundant data helps to reduce opportunity to result decisions based on noise) and reducing training time (fewer data reduces algorithm complexity). In this study, we proposed to use Principal Component Analysis (PCA) (Wold *et al*., 1987; Shlens, 2014), Feature Importance (Abualigah *et al*., 2017), Univariate Selection (Rahman and Xu, 2004) instead of Recursive Feature Elimination with Cross-Validation technique used in (Lima Filho *et al*., 2019). PCA is a statistical method which projects data in a higher dimensional space into a lower dimensional space by maximizing the variance of each dimension. PCA builds new coordinate axes, has the ability to represent data equally well and ensures the variability of the data on each new dimension. PCA can discover the relation of feature in the new space, it is difficult to detect if placed in the old space because these relations are not visible. Feature Importance refers to the feature selection technique which gives us a score for each feature from network traffic data in which the higher the score more important or relevant is the feature towards our output variable. The importance characteristic is very useful to understand more about the data, the model and we can choose to remove features has lower importance and vice versa. Univariate feature selection examines each feature individually to determine the strength of the relationship of the feature with the response variable. Both FI and US help us choose redundant features to remove, sampling fewer features of traffic, saving system storage space. These methods are simple and particularly good for us to

obtain a better understanding of data (Rahman and Xu, 2004). With FI, we propose using Extra Trees to estimate importance for each feature because Extra Trees uses random split can save more computation time than Random Forest using best split and then, we set threshold by SelectFromModel to keep the number of feature important. Features have higher importance or equal this threshold will be selected. US also calculate the score for each feature, so in this proposed system, we use SelectKBest in Pycharm to select features what we would like to keep by setting a threshold.

Benchmark DoS/ DDoS Datasets

In this study, we have used three benchmark DoS/DDoS datasets which are (NSL-KDD, 2009), CICIDS 2017 (Sharafaldin *et al*., 2018) and simulated traffic in (Lima Filho *et al*., 2019). NSL-KDD 2019 is the up-to-date dataset we choose for testing the model since it has a lot of improvement compared to KDD CUP 99. There has been a competition called the KDD Cup, an international competition for knowledge mining and data mining tools. The mission of the competition is to design a network intrusion detection system which aims to be a predictive model that can distinguish "normal" or "abnormal" connection. The results of the competition collected several network traffic records and gathered a dataset called KDD'99 and since then NSL-KDD dataset was created which is an optimized version KDD'99 from the University of New Brunswick (NSL-KDD, 2009). Finally, the complete dataset NSL-KDD 2019 is an up-to-date dataset which contains new types of attacks and removed duplicates from the KDD'99 dataset. This resulting dataset contains about 150,000 data points and is divided into predefined training and test subsets which are KDDTest+, KDDTest-21, KDDTrain+, KDDTrain+_20% where KDDTest-21 and KDDTrain + _20% are subsets of KDDTest+ and KDDTrain+. KDDTrain+ is considered a training data and KDDTest+ is considered a testing data. KDDTest-21 is a subset of the testing data which removes the most difficult data records (point 21). KDDTrain _20% is a subset of the training data where the number of records equal to 20% of the total number of records in the training data. In other words, the records in KDDTest-21 and KDDTrain+_20% are included in testing and training data and no records exist in both datasets at the same time which make the evaluation of anomaly detection more accurately. CICIDS 2017 was created within an emulated environment over a period of 5 days and contains network traffic in packet-based and bidirectional flow-based format. For each flow, the authors extracted more than 80 attributes/features and provide additional metadata about IP addresses and attacks. This dataset contains a wide range of attack types such as SSH brute force, heartbleed, botnet, DoS, DDoS, web and infiltration attacks. In the original article (Sharafaldin *et al*., 2018) studied by the CIC organization that published this dataset, Iman Sharafaldin and colleagues used the RandomForestRegressor algorithm to select the best characteristics for each specific type of attack in CICIDS dataset. To select these features, they calculated the weight of each feature corresponding to each attack type. Finally, the selected features will be tested for performance and accuracy with seven machine learning algorithms. The results show that the ID3 model providing the highest F1 index which reaches up to 98%. Aksu *et al*. (2018), Doğukan Aksu and colleagues proposed a model using fisher score algorithm to select 30 optimal features from 80 of CICIDS2017 dataset. Then, they applied several machine learning algorithms such as KNN, SVM, Decision Tree to test and evaluate the results in which F1 measures 0.99 for the DDos dataset. The simulated network data in (Lima Filho *et al*., 2019) has done by simulating several VLANs which can connect to the Internet. The authors plan to create every 30 min an attack and there are 48 attack events in 24 h, starting at 00:00 and ending at 23:59. The attack tools are parameterized to create sneaky low-volume, medium-volume or light mode and massive high-volume attacks which result to this simulation datasets containing 73 features for a single record. Each record is clearly labeled as "normal" and "attack".

Proposed Model

In this study, the author focuses on applying several feature selection techniques specific to a network flow and use different machine learning algorithms to create a different training system. We will evaluate the execution time, the accuracy of DDoS attack detection between the models. Therefore, we illustrate the empirical study in the Fig. 1. The input of our proposed model use three benchmark DoS/DDoS which are NSL-KDD 2019, CICIDS 2017 and simulated traffic in (Lima Filho *et al*., 2019). After receiving the input data, this feature selection block use three different feature selection techniques: PCA, Feature Importance (using Extra-tree and SelectFromModel of scikit-learn []), Univariate Selection (using SelectKBest with chi-squared algorithm) where:

- PCA: We recalculate the relationship between features and reducing the number of data dimensions to the quantity we want. In order to find the appropriate number of dimensions, we have to repeat this step several times
- Feature Importance and Univariate Selection: Although there are different ways of evaluating individual feature, however there are two techniques to calculate "points" for each feature and then retain the features with "points" higher than the pre-set threshold

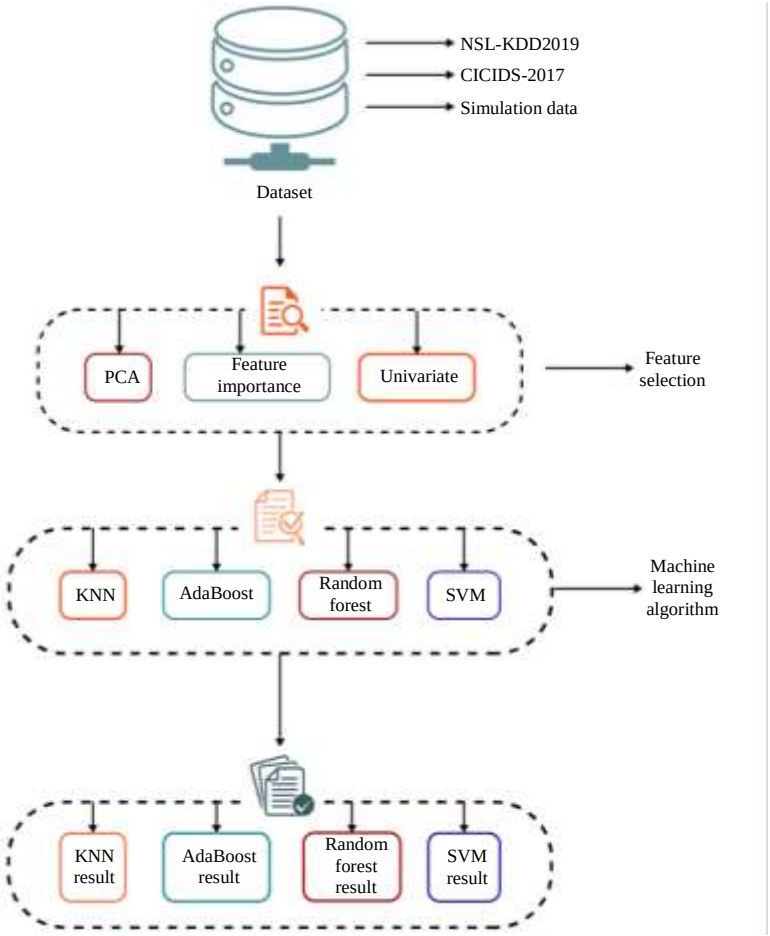

Fig. 1: Proposed model

Table 1: The number of dimensions after the data being reduced

Name of dataset	Number of dimensions
NSL-KDD	21
CIC-IDS 2017	23
Simulated data	20

With the PCA algorithm, the value of the n_components parameter corresponds to the dimension of the data when projecting old data to the new data space. With the algorithm ExtraTree and Chi-Square, the number of dimensions of data depends on the feature importance value of the features that calculated based on these two algorithms. In order to compare the correlation between the classification execution time and the classification performance, the data after reducing dimension through 3 proposed algorithms must have the same number of data dimensions. A match value is given based on the correlation between accuracy, algorithm execution time and the number of dimensions of the data. According to Table 1 in the paper, after applying the data dimension reduction algorithms, the number of dimensions of the NSL-KDD dataset decreased from 42 to 21. The number of dimensions of the 2017 CIC-IDS dataset decreased from 68 to 23 and the number of dimensions of the simulated data set decreased from 73 to 20. Thus, these algorithms have significantly reduced the dimension of data, saving computer resources and computation time.

After dealing with data dimension reduction with feature selection techniques, we have a new data set with the dimension much smaller than the original data. Then, we train this dataset with each machine learning algorithm (KNN, AdaBoost, Random Forest and SVM) to classify DoS/DDoS attacks.

Experimental Classification Results and Analysis

The experimental environment is implemented in the system with the following configuration:

- Operating System: Window 10 64 bit
- CPU: Chip Intel i7 8750H, 6 cores 12 threads
- GPU: Intel UHD Graphic 630 4Gb
- RAM: 8Gb RAM DDR4
- Tool: Pycharm professional 64bit

Evaluation of NSL-KDD 2019 Dataset

NSL-KDD 2019 consists of Internet traffic records observed by a simple intrusion detection network and contains 43 attributes in each record where 41 attributes related to the traffic and the last 2 attributes are label (attack or non-attack) and attack level. In the NSL-KDD 2019 dataset, there are 4 attack classes including Denial of Services (DoS), Reconnaissance (Probe), User to Root (U2R) and Remote to Local (R2L). In this study, the dataset is reprocessed as follows:

- Include two sub-data sets KDDTest + and KDDTrain +
- Remove four layers of Reconnaissance, User to Root and Remote to Local

After performing the preprocessing data, the training data set becomes 77054 normal records and 53385 DDoS attack records have been clearly labeled for each type of attack. The results of applying PCA in NSL-KDD 2019 dataset are shown in Table 2 and the overall result is illustrated in Table 3. We can see that the computation time after data dimension reduction including processing time with PCA technique has been significantly reduced. The reason for using PCA is to recalculate the relationship between features to move from a dimensional space to a less data dimension space. Therefore, every time a network traffic goes through, the system needs to change the data direction of that traffic and then analyze whether the traffic is normal or attack. We also found that the accuracy of the system using the AdaBoost algorithm after reducing the data dimension by PCA increased significantly, but we also noticed that the execution time of both AdaBoost and Random Forest models increased dramatically, showing that recalculating the relationship between features has a good effect on increasing accuracy but makes the processing time of these two algorithms significantly increase. This result shows that this NSL-KDD 2019 is not suitable for the proposed data reduction model by PCA and using AdaBoost and Random Forest to detect attacks. The execution time of the KNN is reduced because fewer dimensions of the data must be calculated, the faster process of KNN. Therefore, we obtain the results showing that the model using KNN and PCA techniques achieved good results on this NSL-KDD 2019 dataset.

Moreover, the results of applying Feature Importance in NSL-KDD 2019 dataset are shown in Table 5 and the overall result is illustrated in Table 4. After performing a dimensional reduction with the Feature Importance technique using the Extra Tree to calculate the Importance of each feature and using SelectFromModel algorithm to select the features that meet user-defined conditions, we can remove 20 redundant characteristic and only 21 features being used. The remaining features are ‘is_host_login’, ‘num_outbound_cmds’, ‘num_shells’, ‘urgent’, ‘num_failed_logins’, ‘num_root’, ‘num_file_creations’, ‘su_attempted’, ‘num_access_file’, ‘root_shell’, ‘is_guest_login’, ‘land’, ‘dst_host_srv_diff_host_rate’, ‘dst_bytes’, ‘duration’, ‘dst_host_diff_srv_rate’, ‘srv_diff_host_rate’, ‘hot’, ‘num_compromised’, ‘service’, ‘dst_host_same_src_port_rate’. With the obtained results, we found that this training data set is not suitable for the AdaBoost algorithm. In addition, the system which makes the selection of high-important "features" in this training data set achieves very good results. We found that processing speed could be significantly improved if the system uses the KNN algorithm to detect network attacks with the accuracy of the system mitigating but the execution time is much faster than not eliminating unnecessary features.

Table 2: The result of detecting each type of attack on NSL-KDD data set when reducing data dimension by PCA

	KNN		AdaBoost		Random Forest		SVM	
Attack	Accuracy level (before PCA) (%)	Accuracy level (after PCA) (%)	Accuracy level (before PCA) (%)	Accuracy level (after PCA) (%)	Accuracy level (before PCA) (%)	Accuracy level (after PCA) (%)	Accuracy level (before PCA) (%)	Accuracy level (after PCA) (%)
Apache2	98.47	99.35	0.00	100.00	100.00	100.00	99.32	99.31
Back	92.10	90.83	0.00	95.07	100.00	93.35	31.02	32.45
Land	100.00	100.00	100.00	100.00	50.00	50.00	100.00	40.00
Mailbomb	93.54	100.00	0.00	100.00	100.00	100.00	89.06	100.00
Neptune	99.67	99.98	97.42	100.00	99.98	100.00	99.98	99.97
Normal	99.71	99.68	98.00	99.90	99.92	99.83	99.78	99.79
Pod	96.22	96.29	0.00	96.36	91.67	97.56	96.49	94.64
Processtable	98.57	97.56	0.00	97.63	100.00	97.10	100.00	100.00
Smurf	99.54	99.53	0.00	99.84	100.00	99.38	99.20	98.82
Teardrop	96.62	100.00	0.00	96.29	98.31	98.10	98.95	98.44

Table 3: The results of the whole model after features reduced by PCA on NSL-KDD 2019

Machine learning	Accuracy level (before PCA)	Accuracy level (after PCA)	Processing time before PCA (ms)	Processing time after PCA (ms)
KNN	99.67	99.66	99424.66	72476.28
AdaBoost	92.06	99.84	21197.64	627577.96
Random Forest	99.91	99.77	7504.45	38984.94
SVM	99.15	99.11	34103.84	20579.71

Table 4: The results of the whole model after features reduced by feature importance on NSL-KDD 2019

Machine learning	Accuracy level (before FI)	Accuracy level (after FI)	Processing time before FI (ms)	Processing time after FI (ms)
KNN	99.61	99.50	101808.01	38171.09
AdaBoost	92.26	94.17	18701.45	12145.49
Random Forest	99.93	99.93	7635.63	5623.96
SVM	99.08	98.99	27682.66	17538.80

Table 5: The result of detecting each type of attack on NSL-KDD data set when reducing data dimension by Feature Importance

	KNN		AdaBoost		Random Forest		SVM	
Attack	Accuracy level (before FI) (%)	Accuracy level (after FI) (%)	Accuracy level (before FI) (%)	Accuracy level (after FI) (%)	Accuracy level (before PCA) (%)	Accuracy level (after PCA) (%)	Accuracy level (before PCA) (%)	Accuracy level (after PCA) (%)
Apache2	99.33	95.52	0.00	0.00	100.00	100.00	99.32	99.31
Back	87.82	86.39	0.00	0.00	100.00	93.35	31.02	32.45
Land	100.00	50.00	100.00	0.00	50.00	50.00	100.00	40.00
Mailbomb	100.00	74.66	0.00	0.00	100.00	100.00	89.06	100.00
Neptune	99.96	99.98	97.63	97.70	99.98	100.00	99.98	99.97
Normal	99.62	99.64	97.89	96.83	99.92	99.83	99.78	99.79
Pod	97.73	94.33	0.00	96.55	91.67	97.56	96.49	94.64
Processtable	97.58	98.47	0.00	0.00	100.00	97.10	100.00	100.00
Smurf	99.84	99.69	0.00	100.00	100.00	99.38	99.20	98.82
Teardrop	98.95	99.84	0.00	0.00	98.31	98.10	98.95	98.44

Table 6: The result of detecting each type of attack on NSL-KDD data set when reducing data dimension by Univariate Selection

	KNN		AdaBoost		Random Forest		SVM	
Attack	Accuracy level (before FI) (%)	Accuracy level (after FI) (%)	Accuracy level (before FI) (%)	Accuracy level (after FI) (%)	Accuracy level (before PCA) (%)	Accuracy level (after PCA) (%)	Accuracy level (before PCA) (%)	Accuracy level (after PCA) (%)
Apache2	99.33	95.52	0.00	0.00	100.00	100.00	99.32	99.31
Back	87.82	86.39	0.00	0.00	100.00	93.35	31.02	32.45
Land	100.00	50.00	100.00	0.00	50.00	50.00	100.00	40.00
Mailbomb	100.00	74.66	0.00	0.00	100.00	100.00	89.06	100.00
Neptune	99.96	99.98	97.63	97.70	99.98	100.00	99.98	99.97
Normal	99.62	99.64	97.89	96.83	99.92	99.83	99.78	99.79
Pod	97.73	94.33	0.00	96.55	91.67	97.56	96.49	94.64
Processtable	97.58	98.47	0.00	0.00	100.00	97.10	100.00	100.00
Smurf	99.84	99.69	0.00	100.00	100.00	99.38	99.20	98.82
Teardrop	98.95	99.84	0.00	0.00	98.31	98.10	98.95	98.44

Table 7: The results of the whole model after features reduced by Univariate Selection on NSL-KDD 2019

Machine learning	Accuracy level (before US)	Accuracy level (after US)	Processing time before US (ms)	Processing time after US (ms)
KNN	99.61	99.50	101808.01	38171.09
AdaBoost	92.26	94.17	18701.45	12145.49
Random Forest	99.93	99.93	7635.63	5623.96
SVM	99.08	98.99	27682.66	17538.80

We continue to apply Univariate Selection in NSL-KDD 2019 dataset and the results are shown in Table 6 and in Table 7. We perform data dimension reduction with the Univariate Selection technique that uses the chi-squared algorithm to calculate squared values for each feature in the data set, then sort them in descending order. We then set the characteristic parameters we want to keep to SelectKBest and the features are taken in order from high to low according to the chi-square parameters until sufficient. Finally, the remaining 21 features being used which are 'is_host_login', 'urgent', 'num_compromised', 'num_root', 'num_file_creations', 'src_bytes', 'num_shells', 'num_failed_logins', 'dst_bytes', 'num_access_files', 'sv_attempted', 'root_shell', 'hot', 'is_guest_login', 'dst_host_diff_srv_rate', 'diff_srv_rate', 'dst_host_srv_diff_host_rate', 'service', 'protocol_type', 'duration', 'dst_host_count'. With the obtained results, we can see that AdaBoost is not working well with this dataset, same when we do feature selection with Feature Important. In summary, through the results achieved after implementing the proposed system on the NSL-KDD 2019 dataset, we realize that the data reduction system by PCA and intrusion detection using KNN algorithms bring the best results. However, we also implement this proposed model using SVM but the obtained result is worst. Moreover, the processing time is very slow, so it is not suitable for online attack detection system. The possible reason is that it is very time-consuming to analyze and compute hyperplane in SVM to classify the attacks.

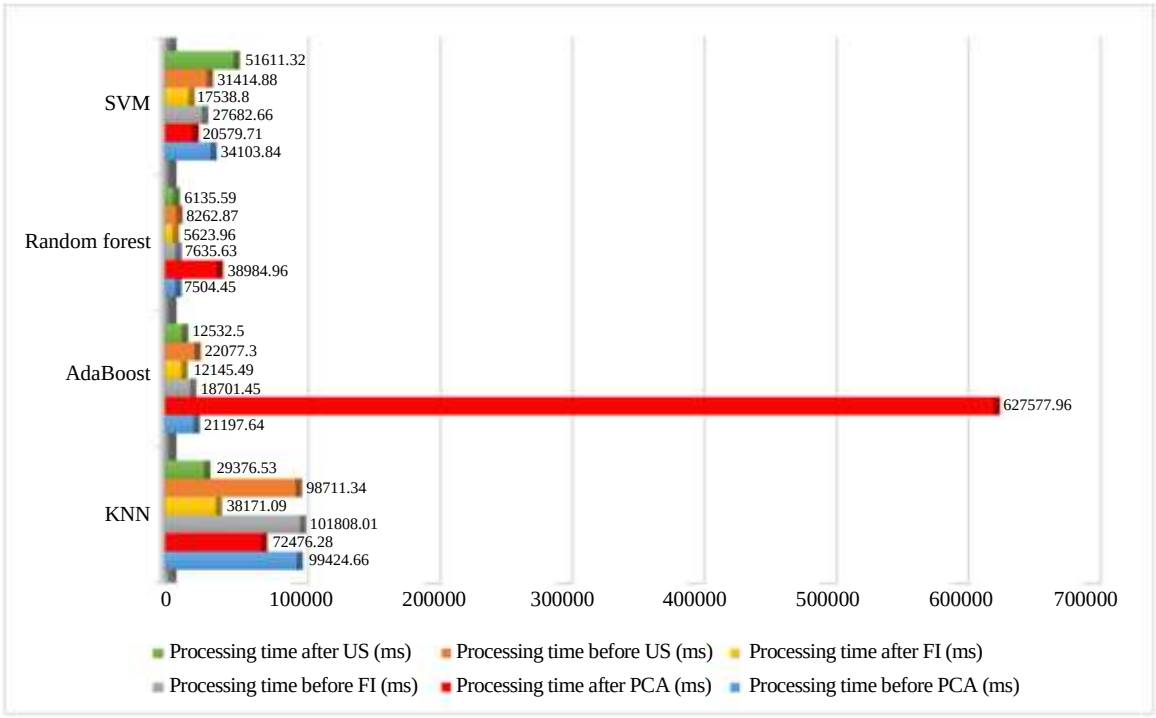

Fig. 2: Processing time on NSL KDD 2019 data set

Based on the results obtained on the execution time as shown in Fig. 2, we can see that KNN provides best results together with all feature selection algorithms. The main reason is that the less the number of dimensions to be computed, the faster the KNN algorithm is processed. But with AdaBoost and Random Forest processing with PCA, the results are not in our expectation. It is probably the correlation between the features of an object and AdaBoost and Random Forest are classified as black box models. It is possible that the reduced correlation between features makes splitting feature to build the tree of both algorithms more difficult and more computationally time consuming.

Evaluation of CICIDS 2017 Dataset

CICIDS 2017 dataset contains normal traffic records and common attacks with real data packets in PCAP. It also includes network traffic analysis results using CICFlowMeter with timestamped flows, source and destination IP, source and destination port, protocol and attack (CSV file). The data collection time begin at 9 am, Monday, July 3, 2017 and ended at 5 pm, Friday 7 July 2017 for a total of 5 days monitoring. Monday is a normal day and only includes valid traffic. Attacks are carried out (including Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS) in the morning and afternoon on Tuesday, Wednesday, Thursday and Friday. In this study, we use attack data on Friday mornings (Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv). Initially, this dataset has 78 features, after pre-processing the data, we remove10 features with 0 variance. Moreover, the dataset had 68 features and was labeled with Benign and DDoS. The data set contains records with values of some features such as NaN, Infinity, which have no effect in the calculation, so they are deleted. After processing the dataset, it includes 97686 Benign records and 128025 DDoS records. After, we then used the Min Max Scaling (Han *et al*., 2011) to normalize the data with characteristic values from -1 to 1 to serve the performance evaluation. The results of applying PCA in CICIDS 2017 dataset are shown in Table 8 when we collected 23 features from the original 68 features of the dataset. We can see that except Random Forest algorithm, the execution time has increased significantly and the implementation time has decreased significantly. This is because PCA has transformed the data set into a new dataset, which makes the structure of newly constructed trees different from the original tree. In general, the accuracy will decrease after reducing the data dimension, but this reduction is acceptable compared to the execution time, i.e., the accuracy decreases about 0.01-0.03% but training and testing times reducing it 2-3 times. Overall, KNN still provides good results with the execution time is much faster and still gives the system a relatively high accuracy. The CIC-IDS 2017 data set is made up of a lot of different traffic files and there are many traffics of different types of DoS/DDoS attacks. In this study, we only use a single file (Friday-

WorkingHours-Afternoon-DDos.pcap_ISCX.csv) as test data. This traffic file has only 2 data types: Benign and DoS. Therefore, we do not have a table of performance for classifying DoS attacks for this dataset.

Moreover, the results of applying Feature Importance in CICIDS dataset are shown in Table 9. After performing a dimensional reduction with the Feature Importance, we get a new data with 23 features which are 'ECE Flag Count', 'RST Flag Count', 'Active Std', 'Active Max', 'Active Min', 'Bwd IAT Min', 'Active Mean', 'Flow Bytes/s', 'Flow IAT Min', 'Fwd IAT Min', 'Bwd Packets/s', 'FIN Flag Count', 'Init_Win_bytes_backward', 'Total Length of Bwd Packets', 'Bwd IAT Mean', 'Idle Std', 'Fwd Packets/s', 'Bwd Header Length', 'Subflow Bwd Bytes', 'Flow Packets/s', 'Idle Min', 'Subflow Fwd Bytes', 'Flow IAT Mean'. Moreover, we find the results to be very positive. The accuracy of the model slightly decreased but the implementation time was much reduced. In addition, when using the Feature Importance, we get only 23 features that need to be processed. Therefore, the network administrator only needs to set the rules so that only 23 features are collected from a network flow which can help reducing data sampling and increasing the processing speed for the overall system.

We continue to apply Univariate Selection on CICIDS 2017 dataset and the result is illustrated in Table 10. Same as previous work, we obtain 23 feature after using Univariate Selection which are 'ECE Flag Count', 'RST Flag Count', 'Active Std', 'Active Max', 'Active Min', 'Bwd IAT Min', 'Active Mean', 'Flow Bytes/s', 'Flow IAT Min', 'Fwd IAT Min', 'Bwd Packets/s', 'FIN Flag Count', 'Init_Win_bytes_backward', 'Total Length of Bwd Packets', 'Bwd IAT Mean', 'Idle Std', 'Fwd Packets/s', 'Bwd Header Length', 'Subflow Bwd Bytes', 'Flow Packets/s', 'Idle Min', 'Subflow Fwd Bytes', 'Flow IAT Mean'.

In Fig. 3, we see that combining feature selection methods gives a good result on each individual model. However, with RF in combination with PCA and SVM in combination with US, the result is not good in terms of processing time.

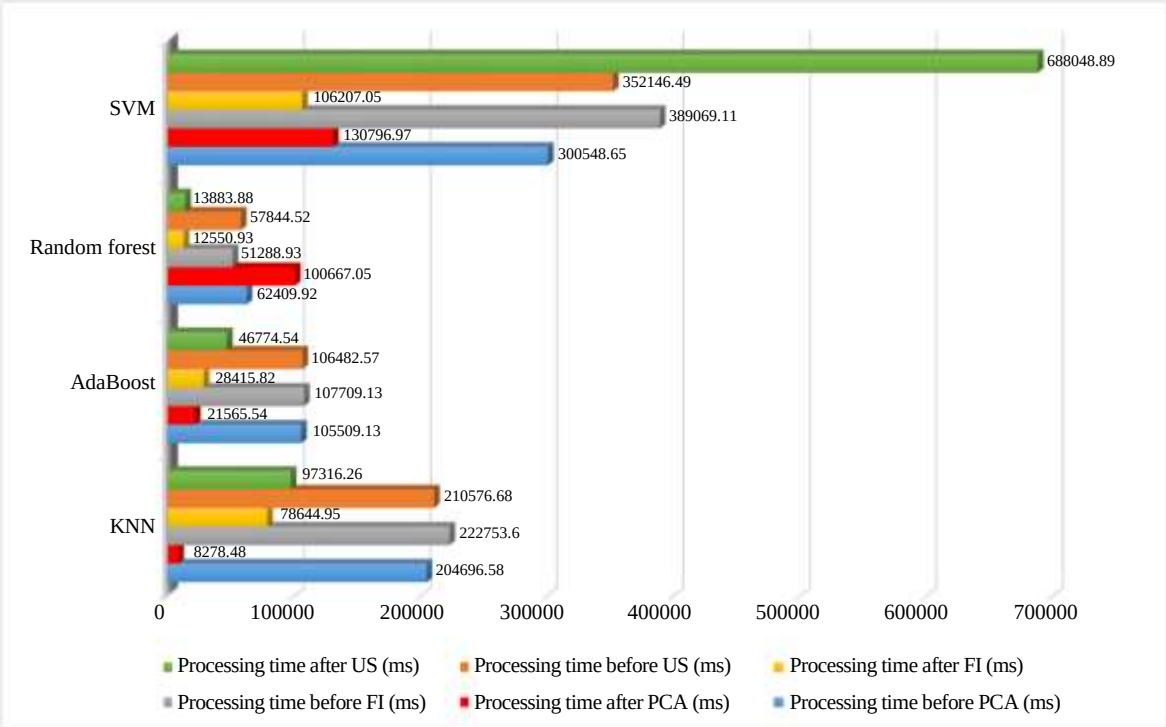

Fig. 3: Processing time on CICIDS 2017 Dataset

Table 8: The results of the whole model after features reduced by PCA on CICIDS 2017

Machine learning	Accuracy level (before PCA)	Accuracy level (after PCA)	Processing time before PCA (ms)	Processing time after PCA (ms)
KNN	99.98	99.95	204696.58	8278.48
AdaBoost	99.98	99.97	105509.13	21565.54
Random Forest	99.99	99.98	62409.92	100667.05
SVM	99.02	99.08	300548.65	130796.97

Table 9: The results of the whole model after features reduced by Feature Importance on CICIDS 2017

Machine learning	Accuracy level (before FI)	Accuracy level (after FI)	Processing time before FI (ms)	Processing time after FI (ms)
KNN	99.98	99.95	222753.60	78644.95
AdaBoost	99.98	99.96	107709.13	28415.82
Random Forest	99.99	99.97	51288.93	12550.93
SVM	99.05	98.40	389069.11	106207.05

Table 10: The results of the whole model after features reduced by Univariate Selection on CICIDS 2017

Machine learning	Accuracy level (before US)	Accuracy level (after US)	Processing time before US (ms)	Processing time after US (ms)
KNN	99.98	99.95	222753.60	78644.95
AdaBoost	99.98	99.96	107709.13	28415.82
Random Forest	99.99	99.97	51288.93	12550.93
SVM	99.05	98.40	389069.11	106207.05

Evaluation of the Simulated Traffic

This dataset is quite similar to a real network operating environment (Lima Filho *et al*., 2019). Therefore, we chose to use this data set to do performance evaluation with the proposed model However, we find that this data set is not large enough which includes 45500 records (including 22412 attacks and 23088 normal records). The results of applying PCA in this dataset is illustrated in Table 11 when we collected 20 features from the original 73 features of this dataset. The computation time after data dimension reduction has taken into account the processing time with PCA due to the nature of this technique is to re-calculate the relationship between the features to move from multi-dimensional space to a less data dimensional space. Therefore, every time a network traffic goes through, the system needs to change the data direction of that traffic, then analyze whether the traffic is normal or attack. We found that except for the Random Forest and AdaBoost algorithms the execution time increases significantly, but the overall the execution time decreases significantly. In general, the accuracy will decrease after reducing the data dimension, but it is acceptable compared to the execution time. In addition, we find that KNN algorithm is very suitable for this training dataset since the execution time is much faster and still provides a relatively high accuracy level.

Moreover, the results of applying Feature Importance using Extra Tree in CICIDS dataset are shown in Table 12. After performing a dimensional reduction with the Feature Importance, we get a new data with 20 features which are ‘tcp_dataofs_median’, ‘tcp_dataofs_mean’, ‘tcp_flags_mean’, ‘ip_proto’, ‘ip_ttl_cv’, ‘tcp_flags_rte’, ‘ip_len_std’, ‘ip_ttl_std’, ‘tcp_flags_median’, ‘ip_len_entropy’, ‘sport_entropy’, ‘tcp_seq_mean’, ‘tcp_dataofs_rte’, ‘ip_len_cv’, ‘ip_ttl_cvq’, ‘tcp_ack_entropy’, ‘tcp_flags_cv’, ‘tcp_seq_entropy’, ‘tcp_ack_cvq’, ‘ip_len_mean’. Moreover, we also found that the result is quite positive since the implementation time is much reduced.

We continue to apply Univariate Selection on simulating dataset and the result is illustrated in Table 13. Same as previous work, we obtain 20 feature after using Univariate Selection which are ‘‘ip_ttl_cv’, ‘ip_len_cv’, ‘ip_len_cvq’, ‘ip_ttl_cvq’, ‘tcp_ack_rte’, ‘tcp_seq_cvq’, ‘tcp_seq_rte’, ‘tcp_dataofs_median’, ‘tcp_dataofs_mean’, ‘tcp_window_median’, ‘dport_cv’, ‘tcp_window_mean’, ‘tcp_flags_mean’, ‘tcp_flags_median’, ‘tcp_ack_cvq’, ‘tcp_seq_mean’, ‘tcp_seq_median’, ‘tcp_seq_cv’, ‘ip_ttl_std’, ‘ip_len_std’.

We found that with two datasets (CICIDS 2017 and simulating traffic (Lima Filho *et al*., 2019)) that contain only normal and attack labels, the proposed model all performed well except for SVM. Therefore, it could provide a solution for an online network intrusion detection but still give relatively high overall accuracy. With the NSL-KDD 2019 dataset, the accuracy of classifying individual attack when using the AdaBoost algorithm is not good. Most attacks classified in the NSL-KDD dataset by the AdaBoost algorithm have very low performance. The traffic types for which the AdaBoost algorithm can has a high classification probability such as Neptune, normal and pod all have lower value than the other algorithms. Specifically, the accuracy of Neptune traffic classification by the AdaBoost algorithm is 96.83%, 2.81% lower than that of KNN algorithm and 3% compared to the Random Forest algorithm. Thus, the classification ability of the AdaBoost algorithm on the NSL-KDD dataset is not good. The proposed model can provide high accuracy of anomaly detection but when classifying each specific attack type, the accuracy is relatively low and there are few false alarms. We find that the proposed system is special good for labeled data sets which are normal or attack. The two models using KNN and Random Forest combined with feature selection techniques have good results in both accuracy and implementation time. Finally, we find that the proposed algorithm to achieve the best results on all three data sets is the combination of KNN algorithm and the Feature Importance.

Table 11: The results of the whole model after features reduced by PCA on simulating traffic

Machine learning	Accuracy level (before PCA)	Accuracy level (after PCA)	Processing time before PCA (ms)	Processing time after PCA (ms)
KNN	99.96	99.94	6850.42	1428.97
AdaBoost	99.97	99.97	536.33	964.04
Random Forest	99.99	99.98	4121.98	6626.24
SVM	99.97	99.96	997.33	907.66

Table 12: The results of the whole model after features reduced by Feature Importance on simulating traffic

Machine learning	Accuracy level (before FI)	Accuracy level (after FI)	Processing time before FI (ms)	Processing time after FI (ms)
KNN	99.98	99.94	6166.51	808.83
AdaBoost	99.98	99.97	527.59	131.64
Random Forest	99.99	99.98	4123.95	1894.93
SVM	99.98	99.92	1017.28	389.99

Table 13: The results of the whole model after features reduced by Univariate Selection on CICIDS 2017

Machine learning	Accuracy level (before US)	Accuracy level (after US)	Processing time before US (ms)	Processing time after US (ms)
KNN	99.97	99.96	6765.79	1434.16
AdaBoost	99.97	99.96	443.76	174.48
Random Forest	99.99	99.98	4077.45	2441.52
SVM	99.96	99.95	1047.22	303.19

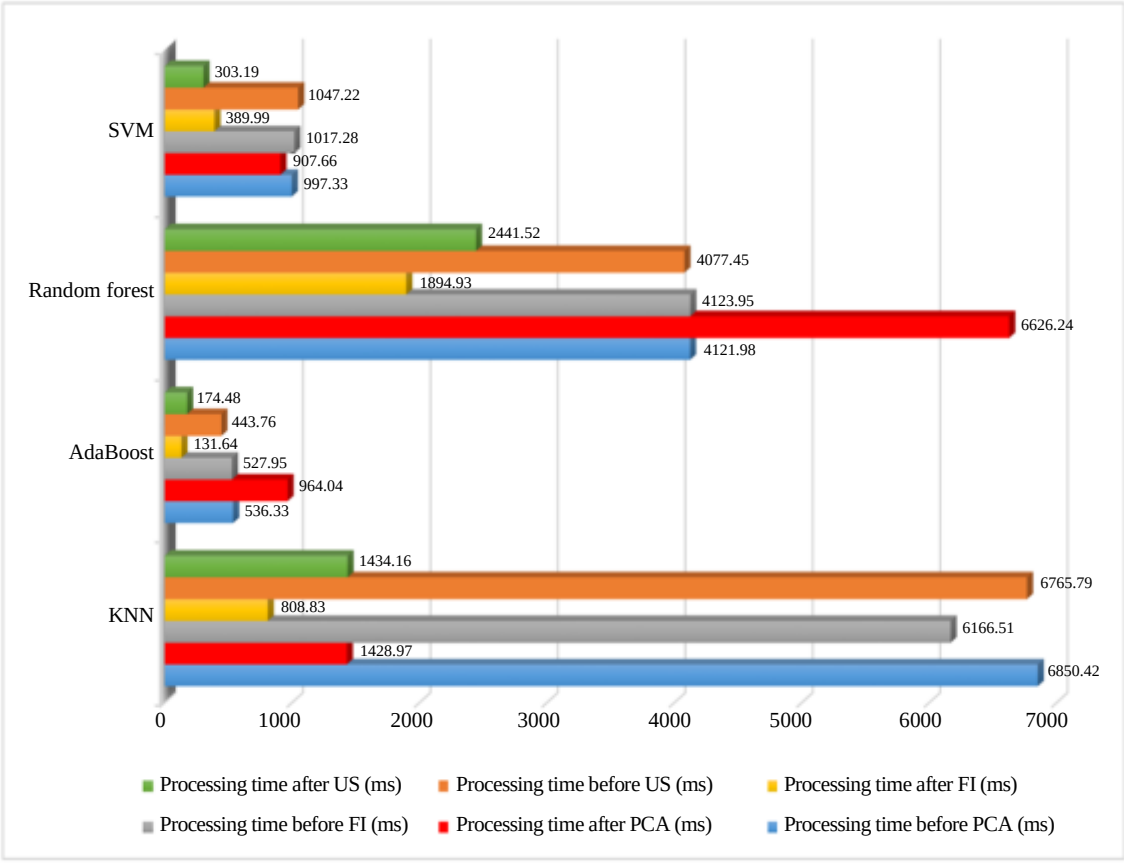

Fig. 4: Processing time on simulating data

In Fig. 4, we show that Random Forest algorithm gives the best results on three approaches of dimensionality reduction in accuracy but this one consumes system's runtime significantly. After using Importance technique,

the performance of KNN algorithm is much improved since only important features are retained. Moreover, the lower the number of data dimensions, the faster the calculation of KNN. Therefore, although the accuracy is slightly reduced, the calculation time is greatly reduced and this is acceptable for us.

Conclusion

In this study, we have proposed a model for empirical study for Machine Learning-based Network Intrusion Detection with Feature Selection algorithm which are PCA, Feature Importance and Univariate Selection. Our contribution is to study in detail of Machine Learning algorithms to work with Feature Selection techniques to evaluate the accuracy level of each combination of machine learning model and feature selection technique. Processing network traffic flow in an online manner is a difficult task especially when it contains a lot of redundant features/or characteristics. Moreover, we found that not all machine learning models can provide good results as in previous works, therefore we have evaluated the proposed models on three benchmark datasets which are NSL-KDD 2019, CICIDS 2017 and simulating traffic (Lima Filho *et al*., 2019). Lastly, we conclude that the combination of KDD and Feature Importance can provide a feasible solution toward an online network intrusion detection system.

Acknowledgement

This publication was supported by a research grant, T2020-PC-209, from Hanoi University of Science and Technology Research Fund.

Author's Contributions

Tran Hoang Hai: Designed the research plan and contributed to the writing of the manuscript.

Nguyen Trong Khiem: Did all experiments on SVM and Random Forest algorithms.

Nguyen Huu Phuc: Did all experiments on AdaBoost and KNN algorithms.

Ethics

We declare there isn't any ethical issues that may arise after the publication of this manuscript.

References

Abualigah, L. M., Khader, A. T., Al-Betar, M. A., & Alomari, O. A. (2017). Text feature selection with a robust weight scheme and dynamic dimension reduction to text document clustering. Expert Systems with Applications, 84, 24-36. https://doi.org/10.1016/j.eswa.2017.05.002

Aksu, D., Üstebay, S., Aydin, M. A., & Atmaca, T. (2018, September). Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm. In International Symposium on Computer and Information Sciences (pp. 141-149). Springer, Cham. https://doi.org/10.1007/978-3-030-00840-6_16

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 46(3), 175-185. https://doi.org/10.1080/00031305.1992.10475879

Argyraki, K. J., & Cheriton, D. R. (2005, April). Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. In USENIX annual technical conference, general track (Vol. 38). https://www.usenix.org/legacy/event/usenix05/tech/general/full_papers/argyraki/argyraki_html/

Benaddi, H., Ibrahimi, K., & Benslimane, A. (2018, October). Improving the intrusion detection system for nsl-kdd dataset based on pca-fuzzy clustering-knn. In 2018 6th International Conference on Wireless Networks and Mobile Communications (WINCOM) (pp. 1-6). IEEE. https://doi.org/10.1109/WINCOM.2018.8629718

Biau, G. (2012). Analysis of a random forests model. The Journal of Machine Learning Research, 13(1), 1063-1095. https://www.jmlr.org/papers/volume13/biau12a/biau12a.pdf

Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407. https://doi.org/10.1145/3394486.3406704

Chellam, A., Ramanathan, L., & Ramani, S. (2018). Intrusion detection in computer networks using lazy learning algorithm. Procedia Computer Science, 132, 928-936. https://doi.org/10.1016/j.procs.2018.05.108

Cisco Annual Internet Report (2018–2023). https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297. https://doi.org/10.1007/BF00994018

Dali, L., Bentajer, A., Abdelmajid, E., Abouelmehdi, K., …, & Abderahim, B. (2015). A survey of intrusion detection system. In 2nd World Symposium on Web Applications and Networking (WSWAN, (pp. 1-6). IEEE. https://doi.org/10.1109/WSWAN.2015.7210351

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. The Annals of Statistics, 7(1), 1-26. https://doi.org/10.1214/aos/1176344552

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Annals of Statistics, 28(2), 337-407. https://doi.org/10.1214/aos/1016218223

Fuchsberger, A. (2005). Intrusion detection systems and intrusion prevention systems. Information Security Technical Report, 10(3), 134-139. https://doi.org/10.1016/j.istr.2005.08.001

Ganapathy, S., Kulothungan, K., Muthurajkumar, S., Vijayalakshmi, M., Yogesh, P., & Kannan, A. (2013). Intelligent feature selection and classification techniques for intrusion detection in networks: a survey. EURASIP Journal on Wireless Communications and Networking, 2013(1), 1-16. https://doi.org/10.1186/1687-1499-2013-271

Gandhi, R. (2018). Boosting Algorithms: AdaBoost, Gradient Boosting and XGBoost. Retrieved from hackernoon. com.

Goeschel, K. (2016, March). Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees and naive Bayes for off-line analysis. In SoutheastCon 2016 (pp. 1-6). IEEE. https://doi.org/10.1109/SECON.2016.7506774

Gupta, B. B., Joshi, R. C., & Misra, M. (2009). Defending against distributed denial of service attacks: issues and challenges. Information Security Journal: A Global Perspective, 18(5), 224-247. https://doi.org/10.1080/19393550903317070

Gupta, B. B., Joshi, R. C., & Misra, M. (2010). Distributed denial of service prevention techniques. International Journal of Computer and Electrical Engineering, 2(2), 268-276. https://doi.org/10.7763/IJCEE.2010.V2.148

Han, J., Kamber, M., & Pei, J. (2011). Data transformation and data discretization. Data Mining: Concepts and Techniques. Elsevier, 111-118.

Ho, T. K. (1995, August). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282). IEEE. https://doi.org/10.1109/ICDAR.1995.598994

Ho, T. K. (1998). The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8), 832-844. https://doi.org/10.1109/34.709601

Ioannidis, J., & Bellovin, S. M. (2002). Implementing pushback: Router-based defense against DDoS attacks. https://academiccommons.columbia.edu/doi/10.7916/D8R78MXV

Jamshidi, Y., & Nezamabadi, P. H. (2013). A lattice based nearest neighbor classifier for anomaly intrusion detection. https://www.sid.ir/en/journal/ViewPaper.aspx?id=386781

Jolliffe, I. T. (1986). Principal Component Analysis. Springer Series in Statistics. Springer-Verlag. pp. 487. ISBN 978-0-387-95442-4.

Karthikeyan, R., & Indra, A. (2010). Intrusion Detection Tools and techniques–a Survey'. International Journal of Computer Theory and Engineering, 2(6), 1793-8201. https://doi.org/10.7763/IJCTE.2010.V2.260

Khan, L., Awad, M., & Thuraisingham, B. (2007). A new intrusion detection system using support vector machines and hierarchical clustering. The VLDB Journal, 16(4), 507-521. https://doi.org/10.1007/s00778-006-0002-5

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2(1), 1-22. https://doi.org/10.1186/s42400-019-0038-7

Larranaga, P., Karshenas, H., Bielza, C., & Santana, R. (2013). A review on evolutionary algorithms in Bayesian network learning and inference tasks. Information Sciences, 233, 109-125. https://doi.org/10.1016/j.ins.2012.12.051

Lee, H., Song, J., & Park, D. (2005, August). Intrusion detection system based on multi-class SVM. In International Workshop on Rough Sets, Fuzzy Sets, Data Mining and Granular-Soft Computing (pp. 511-519). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11548706_54

Lima Filho, F. S. D., Silveira, F. A., de Medeiros Brito Junior, A., Vargas-Solar, G., & Silveira, L. F. (2019). Smart detection: an online approach for DoS/DDoS attack detection using machine learning. Security and Communication Networks, 2019. https://www.hindawi.com/journals/scn/2019/1574749/

Liu, X., Yang, X., & Lu, Y. (2008, August). To filter or to authorize: Network-layer DoS defense against multimillion-node botnets. In Proceedings of the ACM SIGCOMM 2008 conference on Data communication (pp. 195-206). https://doi.org/10.1145/1402946.1402981

Liu, X., Yang, X., & Xia, Y. (2010). Netfence: preventing internet denial of service from inside out. ACM SIGCOMM Computer Communication Review, 40(4), 255-266. https://doi.org/10.1145/1851275.1851214

Liu, Z., Jin, H., Hu, Y. C., & Bailey, M. (2016, October). MiddlePolice: Toward enforcing destination-defined policies in the middle of the Internet. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 1268-1279). https://doi.org/10.1145/2976749.2978306

Mahajan, R., Bellovin, S. M., Floyd, S., Ioannidis, J., Paxson, V., & Shenker, S. (2002). Controlling high bandwidth aggregates in the network. ACM SIGCOMM Computer Communication Review, 32(3), 62-73. https://doi.org/10.1145/571697.571724

Miao, J., & Niu, L. (2016). A survey on feature selection. Procedia Computer Science, 91, 919-926. https://doi.org/10.1016/j.procs.2016.07.111

NSL-KDD. (2009). NSL-KDD data set for network-based intrusion detection systems. http://nsl.cs.unb.ca/KDD/NSLKDD.html

Panda, M., Abraham, A., & Patra, M. R. (2012). A hybrid intelligent approach for network intrusion detection. Procedia Engineering, 30, 1-9. https://doi.org/10.1016/j.proeng.2012.01.827

Panja, B., Ogunyanwo, O., & Meharia, P. (2014, June). Training of intelligent intrusion detection system using neuro fuzzy. In 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD) (pp. 1-6). IEEE. https://doi.org/10.1109/SNPD.2014.6888688

Peng, C. Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An introduction to logistic regression analysis and reporting. The Journal of Educational Research, 96(1), 3-14. https://doi.org/10.1080/00220670209598786

Rahman, S., & Xu, H. (2004). A univariate dimension-reduction method for multi-dimensional integration in stochastic mechanics. Probabilistic Engineering Mechanics, 19(4), 393-408. https://doi.org/10.1016/j.probengmech.2004.04.003

Rao, B. B., & Swathi, K. (2017). Fast KNN classifiers for network intrusion detection system. Indian Journal of Science and Technology, 10(14), 1-10. https://doi.org/10.17485/ijst/2017/v10i14/93690

Resende, P. A. A., & Drummond, A. C. (2018). A survey of random forest based methods for intrusion detection systems. ACM Computing Surveys (CSUR), 51(3), 1-36. https://doi.org/10.1145/3178582

Sahu, S. K., Katiyar, A., Kumari, K. M., Kumar, G., & Mohapatra, D. P. (2019). An SVM-based ensemble approach for intrusion detection. International Journal of Information Technology and Web Engineering (IJITWE), 14(1), 66-84. https://doi.org/10.4018/IJITWE.2019010104

Savage, S., Wetherall, D., Karlin, A., & Anderson, T. (2000, August). Practical network support for IP traceback. In Proceedings of the conference on Applications, Technologies, Architectures and Protocols for Computer Communication (pp. 295-306). https://doi.org/10.1145/347057.347560

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. Machine Learning, 37(3), 297-336. https://doi.org/10.1023/A:1007614523901

Seraphim, B. I., Palit, S., Srivastava, K., & Poovammal, E. (2018, December). A Survey on Machine Learning Techniques in Network Intrusion Detection System. In 2018 4th International Conference on Computing Communication and Automation (ICCCA) (pp. 1-5). IEEE. https://doi.org/10.1109/CCAA.2018.8777596

Shahraki, A., Abbasi, M., & Haugen, Ø. (2020). Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. Engineering Applications of Artificial Intelligence, 94, 103770. https://doi.org/10.1016/j.engappai.2020.103770.

Sharafaldin, I., Gharib, A., Lashkari, A. H., & Ghorbani, A. A. (2018). Towards a reliable intrusion detection benchmark dataset. Software Networking, 2018(1), 177-200. https://doi.org/10.13052/jsn2445-9739.2017.009.

Shlens, J. (2014). A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100. https://arxiv.org/abs/1404.1100

Song, D. X., & Perrig, A. (2001, April). Advanced and authenticated marking schemes for IP traceback. In Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213) (Vol. 2, pp. 878-886). IEEE. https://doi.org/10.1109/INFCOM.2001.916279

Srivastava, A., Gupta, B. B., Tyagi, A., Sharma, A., & Mishra, A. (2011, September). A recent survey on DDoS attacks and defense mechanisms. In International Conference on Parallel Distributed Computing Technologies and Applications (pp. 570-580). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24037-9_57

Thai, L. H., Hai, T. S., & Thuy, N. T. (2012). Image classification using support vector machine and artificial neural network. International Journal of Information Technology and Computer Science, 4(5), 32-38. https://doi.org/10.5815/ijitcs.2012.05.05

Tharwat, A., Ghanem, A. M., & Hassanien, A. E. (2013, December). Three different classifiers for facial age estimation based on k-nearest neighbor. In 2013 9th International Computer Engineering Conference (ICENCO) (pp. 55-60). IEEE. https://doi.org/10.1109/ICENCO.2013.6736476

Turner, C., Jeremiah, R., Richards, D., & Joseph, A. (2016). A rule status monitoring algorithm for rule-based intrusion detection and prevention systems. Procedia Computer Science, 95, 361-368. https://doi.org/10.1016/j.procs.2016.09.346.

UCI. (1999). KDD Cup 1999 Data. University of California, Irvine (UCI). https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Vezhnevets, A., & Vezhnevets, V. (2005, September). Modest AdaBoost-teaching AdaBoost to generalize better. In Graphicon (Vol. 12, No. 5, pp. 987-997). http://calvin-vision.net/bigstuff/hp_avezhnev/Pubs/ModestAdaBoost.pdf

Wang, W., Zhang, X., Gombault, S., & Knapskog, S. J. (2009, December). Attribute normalization in network intrusion detection. In 2009 10th International Symposium on Pervasive Systems, Algorithms and Networks (pp. 448-453). IEEE. https://doi.org/10.1109/I-SPAN.2009.49

Winter, P., Hermann, E., & Zeilinger, M. (2011, February). Inductive intrusion detection in flow-based network data using one-class support vector machines. In 2011 4th IFIP international conference on new technologies, mobility and security (pp. 1-5). IEEE. https://doi.org/10.1109/NTMS.2011.5720582

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. Chemometrics and Intelligent Laboratory Systems, 2(1-3), 37-52. https://doi.org/10.1016/0169-7439(87)80084-9

Yaar, A., Perrig, A., & Song, D. (2004, May). SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks. In IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004 (pp. 130-143). IEEE. https://doi.org/10.1109/SECPRI.2004.1301320.

Yang, X., Wetherall, D., & Anderson, T. (2008). TVA: A DoS-limiting network architecture. IEEE/ACM Transactions on Networking, 16(6), 1267-1280. https://doi.org/10.1109/TNET.2007.914506.

Zhang, C., Jiang, J., & Kamel, M. (2005). Intrusion detection using hierarchical neural networks. Pattern Recognition Letters, 26(6), 779-791. https://doi.org/10.1016/j.patrec.2004.09.045.

Zwass, V. (2018). Expert system. Computer Science, Enclyclopaedia Britannica, https://www.britannica.com/technology/expert-system.