Anomaly-Based Network Intrusion Detection System through Feature Selection and Hybrid Machine Learning Technique

Apichit Pattawaro
Master of Science in Information Technology,
Department of Computer Science, Faculty of Science
Srinakharinwirot University, Bangkok, Thailand
Apichit.pattawaro@g.swu.ac.th

Chantri Polprasert
Master of Science in Information Technology,
Department of Computer Science, Faculty of Science
Srinakharinwirot University, Bangkok, Thailand
Chantri@g.swu.ac.th

Abstract—**In this paper, we propose an anomaly-based network intrusion detection system based on a combination of feature selection, K-Means clustering and XGBoost classification model. We test the performance of our proposed system over NSL-KDD dataset using KDDTest$^+$ dataset. A feature selection method based on attribute ratio (AR) [14] is applied to construct a reduced feature subset of NSL-KDD dataset. After applying K-Means clustering, hyperparameter tuning of each classification model corresponding to each cluster is implemented. Using only 2 clusters, our proposed model obtains accuracy equal to 84.41% with detection rate equal to 86.36% and false alarm rate equal to 18.20% for KDDTest$^+$ dataset. The performance of our proposed model outperforms those obtained using the recurrent neural network (RNN)-based deep neural network and other tree-based classifiers. In addition, due to feature selection, our proposed model employs only 75 out of 122 features (61.47%) to achieve this level of performance comparable to those using full number of features to train the model.**

Keywords—*Hybrid Clustering and Classification, NSL-KDD, network security*

I. Introduction

Network Intrusion Detection Systems (NIDS) [1, 2] play a crucial role in the protection of computer systems from network-based malicious attacks that could disrupt the services of the system. Providing powerful and robust NIDS is a very challenging task due to several factors. For instance, with the growth of the Internet traffic which consisting of a variety of data type traversing the network, NIDS must be able to analyze these huge volume of traffic and differentiate between normal and malicious behavior with acceptable accuracy. Typically, NIDS systems are broadly categorized into 2 types consisting of 1) misuse-based system (sometimes called signature-based) 2) anomaly-based system.

A misuse-based NIDS system relies on an extensive database of attack signatures. Each signature is a set of rules corresponding to intrusion attacks occurred in the past. Therefore, with up-to-date NIDS signature database, a misused-based system is very powerful in detecting past attacks. However, this system is vulnerable to zero-day attack and long processing time. An anomaly-based IDS system detects attacks in the computer system through observing unusual traffic statistics or pattern. This system could be used to detect zero-day attack and results from the discovery of the attack could be added into the database for future detection through misused-based. However, an anomaly-based system suffers from high false alarm rate since much normal traffic exhibiting unusual behavior could trigger the alarm of the system. In practical system, a hybrid of the misused-based and anomaly-based is usually employed to mitigate the impact of both zero-day attack and high false-alarm rate.

Recently, anomaly-based NIDS employing machine learning techniques has gained widespread attentions. A number of classification models e.g. KNN [3], genetic algorithms [4], Random Forest (RF) [5,6], Support Vector Machine (SVM) [7] or Extreme Gradient Boosting (XGBoost) [8] have been used to differentiate between normal or suspicious traffic. However, feasibility of this approach is still limited due to poor detection performance. This could be due many reasons such as diverse nature of traffic, imbalance traffic classes and ineffective feature selection processes. To overcome this limitation, a number of papers utilize deep neural network (DNN) methods such as recurrent neural network (RNN) [9], long short-term memory (LSTM) [10] and convolutional neural network (CNN) [11] for anomaly-based NIDS. Even though the DNN approaches exhibit enhanced detection performance, it requires significant amount of training time and large volume of effective train dataset.

A hybrid approach for anomaly-based NID [12] is another promising alternative. By combining traditional ML model together, significant performance improvement in terms of accuracy, precision, false alarm rate (FAR) is exhibited with acceptable additional computational complexity.

In this paper, we focus on NID's binary classification problem where the system differentiates between normal or attack activities. We propose an anomaly-based network intrusion detection system based on a combination of feature selection, K-Means clustering and XGBoost classification model. The reason behind selection of XGBoost classifier model is due to its strong performance, a variety of hyperparameter selection, fast implementation and popularity among machine learning communities. We test the performance of our proposed system over NSL-KDD dataset [13] using KDDTest$^+$ dataset. A feature selection method based on Attribute Ratio (AR) [14] is applied to construct a reduced feature subset of NSL-KDD dataset. After applying K-Means clustering, hyperparameter tuning of each classification model corresponding to each cluster is implemented. Using only 2 clusters, our proposed model obtains the best accuracy equal to 84.41%, TPR equal to 86.36%, FPR = 18.20 and AUC=0.922 for KDDTest$^+$ dataset. In addition, the performance of our proposed model outperforms those obtained from the RNN-based deep neural network and other tree-based classifiers. Moreover, due to feature selection, our proposed model employs only 75 out of 122 features (61.47%) to achieve this level of performance comparable to those using full number of features to train the model.

The remainder of this paper is organized as follows. Section 2 discusses NSL-KDD dataset. The proposed system is explained in Section 3. Results are evaluated and discussed in Section 4 and our findings in this paper is summarized in Section 5

2. NSL-KDD DATASET

Previously, KDD-Cup 99 dataset [15] has been widely used to test the performance of an anomaly-based intrusion detection system. However, researchers [15]pointed out 2 critical issues based on statistical analysis of the dataset leading to over-simplistic prediction results. To circumvent this problem, they proposed NSL-KDD dataset which has the following advantages compared to KDD-Cup 99 as follows.

1) In NSL-KDD, many redundant and duplicate data encountered in KDD-Cup 99 are removed from the datasets.

2) To achieve more accurate evaluation of different learning techniques, the number of selected records from each difficulty-level group is inversely proportional to the percentage of records in the original KDD dataset.

NSL-KDD dataset is categorized into 4 separate dataset including:

1) KDDTrain$^+$: This is the overall train dataset consisting of 125, 973 recordsKDDTrain$^+$ 20Percent: This is the train dataset consisting of only 20% of the total train dataset and has 25,192 records.

2) KDDTrain$^+$20Percent: This is the train dataset consisting of only 20% of the total train dataset and has 25,192 records.

3) KDDTest$^+$: This is the test dataset consisting of 22,544 records.

4) KDDTest^{-21}: This dataset consists of 11,850 records. This dataset is obtained by applying 21 machine learning model on KDDTest$^+$ dataset to predict the label of the dataset (Normal/Attack) and those that are accurately predicted by all 21 models are discarded from the dataset.

Table 2 lists the ratio of normal/attack for each type of dataset in NSL-KDD. The NSL-KDD categorizes attacks into 4 types consisting of Denial-of-Service, Probe, Root to Local and Unauthorized to Root as presented in Table 1. Each type of attack can be explained as follows:

*1) Denial-of-Service(DoS):*This type of attack overwhelms the targets' resources (Network, CPU or Memory) so that typical operations cannot be performed as expected. Examples of this attack include sending huge number of packets to the targeted server so that normal users cannot access.

*2) Probe:*This type of attack involves port scanning to identify vulnerabilities in computer systems for further attacks.

3) Root to Local (R2L): The attackers try to access the unauthorized computer resources in order to destroy or modify operations of the targeted computer systems

4) Unauthorized to Root (U2R): The attackers try to gain accesses to unauthorized resources using root privileges.

Table 1. Type of Attacks

Category	***Attacks***
DoS (Denial of Service)	Neptune, pod, smurf, teardrop, process table,warezmaster,apache2, mail bomb, back
Probe	multihop, http tunnel, ftp_write, root kit, ps buffer overflow, xterm
R2L (Root to Local)	named, snmpgetattack,xlock, send mail, guess_passwd
U2R (Unauthorized to Root)	ipsweep, nmap, port sweep, satan, mscan, saint

Table2.Ratio of Normal/Attack in each type of NSL-KDD dataset

Dataset	***Classify***	***Total***	***Normal***	***Anomaly***
KDDTrain$^+$	Number	125973	67343	58630
	Percent	100%	53.46%	46.54%
KDDTest$^+$	Number	22544	9711	12833
	Percent	100%	43.08%	56.92%
KDDTest^{-21}	Number	11850	4342	7508
	Percent	100%	36.64%	63.36%

NSL-KDD dataset contains 41 features categorizing into 3 types consisting of 3 nominal features, 6 binary features and 32 numeric features [13].

3. METHODS

Figure 1 illustrates the block diagram of the proposed NIDS model. We employ KDDTrain$^+$ dataset to train the proposed ML model and evaluate its performance in terms of accuracy, AUC, precision and recall using KDDTest$^+$ dataset.

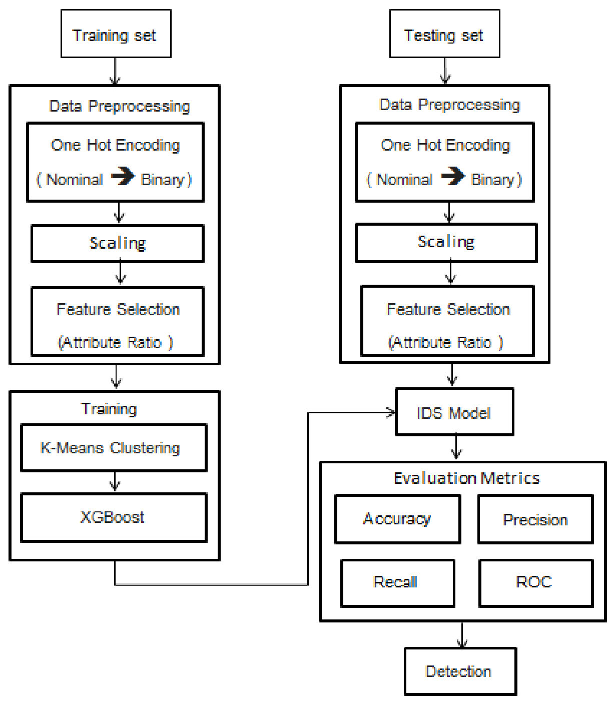


Figure1.Block diagram of the proposed NIDS model.

3.1) Data Preprocessing

There are three sub-processes within the Data Preprocessing consisting of One-Hot Encoding, Scaling and Feature Selection

3.1.1) One Hot Encoding and Scaling

We exercise One-Hot Encoding to transform 3 nominal features listed as protocal_type, service and flag into 84 binary features (protocol has 3 features, service has 70 features and flag has 11 features). In summary, after One-Hot Encoding process, there are 122 features entering Normalization process. During Normalization process, we scale the dataset so that the mean of every feature is equal to zero and standard deviation is equal to one.

3.1.2) Feature Selection

To enhance model efficiency, reduce computational complexity and remove irrelevant features, we implement feature selection based on calculating the average AR. This value will be used to determine the feature importance of every feature and its calculation can be explained as follows. In AR approach, we employ attribute average and frequency for numeric and binary features, respectively. The AR of the jth feature AR (j) can be calculated as

$$\mathrm{AR(j)} = \underset{i\in[0,4]}{\mathrm{Max}}(\mathrm{CR_i(j)}) \qquad (1)$$

where CR$_i$(j) is Class Ratio of the jth feature of the ith label(i∈ [0,4] where i = 0 for Normal, i=1 for DoS, i=2 for Probe, i=3 for R2L and i=4 for U2R class). For jth numeric feature, CRi(j) can be expressed as

$$CR_i(j) = \frac{AVG_{i,j}}{AVG_{T,j}} \qquad (2)$$

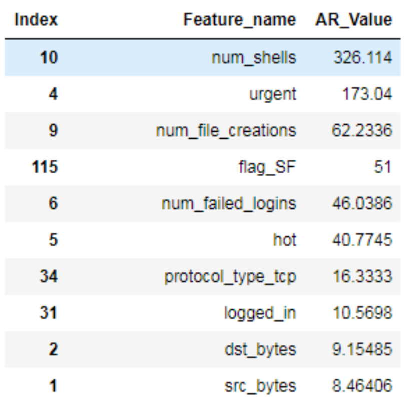
whereAVGi,j=Ci,j/Ni,j. Ci,j is the sum of the jth feature corresponding to ith label and Ni,j is the number of records of the jth features corresponding to the ith label.

$$AVG_{T,j} = \frac{\sum_i C_{i,j}}{N} \qquad (3)$$

is the sum of jth feature divided by the total number of jth feature (Nj). For jth binary feature, CRi(j) can be written as

$$CR_i(j) = \frac{Freq(1)_{i,j}}{Freq(0)_{i,j}} \qquad (4)$$

where Freq(1)i,j is the number of ithrecords whose jth feature is equal to one and Freq(0)i,j is the number of ith records whose jth feature is equal to zero. Figure 2 displays top ten highest-important features obtained from Eq. (1). Features whose AR values less than 0.01 are removed from the analysis. The threshold 0.01 is judiciously selected to obtain the best performance with acceptable computational complexity. After applying feature selection method with threshold equal to 0.01, only 75 out of 122 features (61.47%) are left to be used to train the model.



Index	Feature_name	AR_Value
10	num_shells	326.114
4	urgent	173.04
9	num_file_creations	62.2336
115	flag_SF	51
6	num_failed_logins	46.0386
5	hot	40.7745
34	protocol_type_tcp	16.3333
31	logged_in	10.5698
2	dst_bytes	9.15485
1	src_bytes	8.46406

Figure2.A list of top ten highest-important features.

3.1.3) Clustering

The main objective of applying K-Means clustering to NSL-KDD dataset is to group a set of normal and attack traffic that exhibit similar pattern into the same partitions. Then, ML model corresponding to each partition is trained to differentiate normal or attack data within that group. To determine the number of clusters (K), we implement K-Means clustering on NSL-KDD dataset using different number of clusters and evaluate performance using sum square error (SSE). Figure 3 shows SSE of K-Means clustering over a range of K where random_state is equal to 3425. From the figure, SSE yields highest drop (23.89%) when K is increased from 1 to 2. SSE gradually decreases for higher values of K

(SSE drop is reduced to 9.30% when K is increased from 2 to 3). In addition, no significant numbers of records are presented in a new group when K is greater than 10.

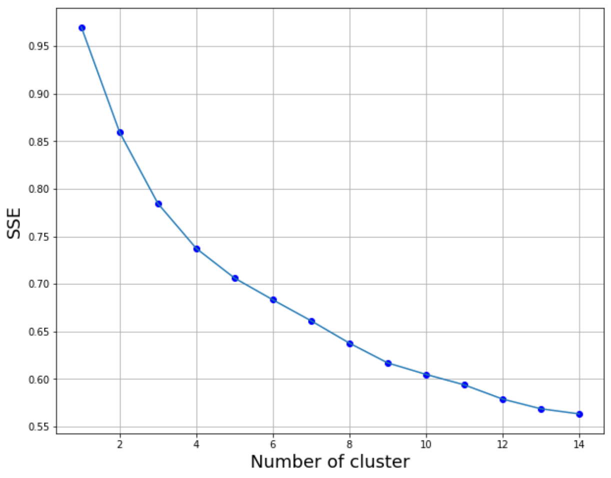


Figure3. SSE of K-Means clustering vs. a number of clusters (K)

3.1.4) XGBoostClassifier

XGBoost was designed for speed and performance based on gradient-boosted decision trees algorithms. It provides the benefit of algorithm enhancement, model tuning, and can also be deployed in different computing environments. In addition, it allows the addition or tuning of regularization parameters to mitigate the impact of over-fitting.

3.2) Evaluation Metrics

We evaluate the performance of our proposed model using Accuracy, True Positive Rate (TPR) or Recall, False Positive Rate (FPR). In addition, we usedArea Under the Curve (AUC)for overall measure of performance across all possible classification thresholds. Accuracy metric can be written as

$$\text{Accuracy} = \frac{TP + TN}{(\text{TP} + \text{FN} + \text{TN} + \text{FP})} \qquad (5)$$

Where*TP*, *TN*, *FP* and FN represent True Positive, True Negative, False Positive and False Negative, respectively. Recall or TPR is the ratio of items correctly classified as attack to all items classified as attack and can be written as

$$\text{TPR} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \qquad (6)$$

FPR is the ratio of items incorrectly classified as attack to all items that belong to normal and can be written as

$$\text{FPR} = \frac{\text{FP}}{(\text{FP} + \text{TN})} \qquad (7)$$

4. Results

We compare the performance of our proposed hybrid ML system in terms of accuracy, TPR, FPR and AUC with the RNN approach and other tree-based classifiers. We use KDDTrain$^+$ dataset to train our model and evaluate the performance over KDDTest$^+$ dataset. Our hybrid model employs feature selection, K-Means clustering followed by XGBoost prediction model to differentiate between normal or attack traffic. We perform feature selection based on AR values where features with AR value less than 0.01 are discarded from analysis. This threshold is judiciously selected to obtain maximum performance. For K-Means clustering, as presented in Fig. 3, we use two clusters (K=2) for K-Means clustering due to its steepest drop in SSE. For hyperparameter tuning in each XGBoost model corresponding to each cluster, we employ RandomizedSearchCV algorithm to select hyperparameters that yield the best model's performance in term of accuracy. A set of hyperparameters we are interested in is presented in Figure 4. The followings are some hyperparameters we use in our model:

- n_estimators: The total number of trees used in the model.
- max_depth: The highest number of tree hierarchies. The higher the value, the more complex the model becomes.
- learning_rate: This parameter mitigates over-fitting problem. It controls step-size shrinkage and weighting factors for corrections when adding new trees to the model.
- subsample: The fraction of samples to be randomly selected for each tree.
- colsample_bytree: The fraction of columns to be randomly sampled for each tree.
- colsample_bylevel : The subsample ratio of column for each feature split, in each layer.
- min_child_weight: The minimum sum of weights of all observations required in a child. It is used to control over-fitting.
- gamma: The minimum loss reduction required to make a split.
- reg_lambda: L2 regularization term on weights It is used to manage the regularization part of XGBoost loss function

For each set of hyperparameter, we implement 5-fold cross-validation to validate the performance of our model in each cluster. Table 3 lists a set of hyperparameter of XGBoost model for each cluster. From Table 3, after hyperparameter tuning, both clusters yield identical hyperparameters.

With 2 clusters employing hyperparameters as listed in Table 3, the performance of our proposed model is exhibited in Table 4. From the table, the proposed model with K=2 yields accuracy equal to 84.41%, TPR = 86.36%, FPR = 18.20% and AUC = 0.84. We compare the performance of our model over a range of cluster groups and found that those with K=2 yields highest accuracy and AUC. TPR and FPR are on the same order over a range of a number of clusters as presented in Table 4. This justifies our selection with K = 2.

```
param_grid = {
        'n_estimators': [100, 200],
        'max_depth': [5, 10, 15, 20],
        'learning_rate': [0.001, 0.01, 0.1, 0.2, 0,3],
        'subsample': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'colsample_bytree': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'colsample_bylevel': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'min_child_weight': [0.4, 0.5, 1.0, 3.0, 5.0, 7.0, 10.0],
        'gamma': [0, 0.25, 0.5, 1.0],
        'reg_lambda': [0.1, 1.0, 5.0, 10.0, 50.0, 100.0]
        }
```

Figure 4.List of XGBoostHyperparameters.

Table 3.A list of XGBoosthyperparameter for each cluster

Cluster	sub sample	reg_ lambda	n_ estimators	min_child _weight	max _depth	learning _rate	gamma	colsample _bytree	colsample _bylevel
0	0.6	5	200	0.4	5	0.2	0	1	0.6
1	0.6	5	200	0.4	5	0.2	0	1	0.6

Table4. Performance of the proposed model in terms of accuracy, TPR and FPR over a range of K-Means clusters

Number of K in K-Means	*KDDTrain $^+$*	*KDDTest $^+$*
K=10	Accuracy = 99.77%	Accuracy = 83.56%
	TPR = 99.83%	TPR = 83.49%
	FPR = 0.28%	FPR = 16.34%
	AUC = 0.9976	AUC = 0.8282
K=8	Accuracy = 99.77%	Accuracy = 83.23%
	TPR = 99.86%	TPR = 83.89%
	FPR = 0.30%	FPR = 17.72%
	AUC = 0.9977	AUC = 0.8263
K=6	Accuracy = 99.75%	Accuracy = 80.32%
	TPR = 99.79%	TPR = 78.37%
	FPR = 0.28%	FPR = 15.99%
	AUC = 0.9975	AUC = 0.7876
K=4	Accuracy = 99.83%	Accuracy = 83.69%
	TPR = 99.84%	TPR = 83.90%
	FPR = 0.18%	FPR = 16.62%
	AUC = 0.9982	AUC = 0.8297
K=2	**Accuracy = 99.85%**	**Accuracy = 84.41%**
	TPR = 99.87%	**TPR = 86.36%**
	FPR = 0.18%	**FPR = 18.20%**
	AUC = 0.9984	**AUC = 0.9227**
(Don't Clustering)	Accuracy = 99.90%	Accuracy = 82.66%
	TPR = 99.92%	TPR = 87.32%
	FPR = 0.11%	FPR = 22.60%
	AUC = 0.9990	AUC = 0.8287

Table 5 lists top ten highest-important features of both clusters. From the table, both clusters exhibit similar pattern where src_bytes feature yields highest importance for both clusters.

Table 5. A list of top ten highest-important feature of cluster 0 and 1 on KDDTrain$^+$ dataset

Feature Name	*Cluster 0*	*Cluster 1*
src_bytes	0.140192	0.132889
dst_bytes	0.074614	0.062317
dst_host_srv_count	0.063247	0.070306
dst_host_diff_srv_rate	0.056252	0.060719
dst_host_count	0.052171	0.057257
dst_host_same_src_port_rate	0.050131	0.045806
duration	0.042845	0.047137
dst_host_rerror_rate	0.041387	0.040213
dst_host_same_srv_rate	0.041096	0.045806
dst_host_srv_diff_host_rate	0.037015	0.033289

The performance of our proposed model in term of accuracy is compared with those from RNN and other tree-based classifiers (Random Forest and Adaboost) in Table 6. From the table, our proposed model obtains highest accuracy compared to others for both KDDTrain$^+$ and KDDTest$^+$ datasets. use to clustering, XGBoost classifier is trained using data exhibiting similar pattern in each cluster and this improves the detection performance compared to those obtained by training the classifier without clustering [16] (Accuracy = 77%, TPR = 62% and FPR = 3%). This could be one of the main reasons that contribute to its superiority over that of the RNN model. For comparison with RF and Adaboost models, with more customizable hyperparameter selection, the proposed model yields superior performance compared to others. In addition, by implementing feature selection using AR gain, our proposed model uses only 75 out of 122 features (61.47%) to achieve strong performance in comparable to those using full 122 features to train the model.

Table6. Comparison of the accuracy metric

Model	*KDDTrain$^+$*	*KDDTest$^+$*
Baseline	99.81%	83.28%
K-Means+XGBoost (Our model)	**99.85%**	**84.41%**
K-Means+Random Forest	99.67%	75.67%
K-Means+Adaboost	99.61%	72.64%

5. Conclusions

We proposed a hybrid machine learning technique for network intrusion detection based on a combination of feature selection, K-Means clustering and XGBoost classification models. We test the performance of our proposed system over NSL-KDD (KDDTrain$^+$, KDDTest$^+$) dataset. A feature selection method based on AR is applied to construct a reduced feature subset of NSL-KDD dataset. After applying K-Means clustering, hyperparameter tuning of each classification model corresponding to each cluster is implemented. Our proposed model obtains the best accuracy equal to 84.41% with detection rate equal to 86.36%, false alarm rate equal to 18.20% and AUC equal to 0.84 for KDDTest$^+$ dataset. In addition, the performance of our proposed model in term of accuracy outperforms those obtained from the RNN-based deep neural network. Due to feature selection, our proposed model employs only 75 out of 122 features (61.47%) to achieve this level of performance comparable to those using full number of features to train the model.

REFERENCES

[1] Buczak, Anna L., and ErhanGuven. "Using Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection." International Journal of Recent Trends in Engineering and Research, vol. 3, no. 4, 2017, pp. 109–111.

[2] Vemuri, V Rao. "Cyber-Security and Cyber-Trust."Enhancing Computer Security with Smart Technology, 2005, pp. 1–8.

[3] Rao, B. Basaveswara, and K. Swathi. "Fast KNN Classifiers for Network Intrusion Detection System." Indian Journal of Science and Technology, vol. 10, no. 14, Jan. 2017, pp. 1–10.

[4] Li, Fan. "Hybrid Neural Network Intrusion Detection System Using Genetic Algorithm." 2010 International Conference on Multimedia Technology, 2010, pp. 597–602.

[5]Farnaaz, Nabila, and M.a.Jabbar. "Random Forest Modeling for Network Intrusion Detection System." Procedia Computer Science, vol. 89, 2016, pp. 213–217.

[6] Zhang, Jiong, et al. "Random-Forests-Based Network Intrusion Detection Systems." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 38, no. 5, 2008, pp. 649–659.

[7] Pervez, Muhammad Shakil, and Dewan Md. Farid. "Feature Selection and Intrusion Classification in NSL-KDD Cup 99 Dataset Employing SVMs." The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), 2014.

[8] Chen, Tianqi, and Carlos Guestrin. "XGBoost." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16, 2016, pp. 785–794.

[9] Yin, Chuanlong, et al. "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks." IEEE Access, vol. 5, 2017, pp. 21954–21961.

[10] Staudemeyer, Ralf C. "Applying Long Short-Term Memory Recurrent Neural Networks to Intrusion Detection." South African Computer Journal, vol. 56, no. 1, Nov. 2015, pp. 136–154.

[11] Li, Zhipeng, et al. "Intrusion Detection Using Convolutional Neural Networks for Representation Learning." Neural Information Processing Lecture Notes in Computer Science, 2017, pp. 858–866.

[12] Kuang, Fangjun, et al. "A Novel Hybrid KPCA and SVM with GA Model for Intrusion Detection." Applied Soft Computing, vol. 18, 2014, pp. 178–184.

[13] "Search UNB." University of New Brunswick Est.1785, www.unb.ca/cic/datasets/nsl.html.

[14] Sang-Hyun, Choi, and ChaeHee-Su. "Feature Selection Using Attribute Ratio in NSL-KDD Data." International Conference Data Mining, Civil and Mechanical Engineering (ICDMCME'2014), Feb 4-5, 2014 Bali (Indonesia), 4 Feb. 2014, pp. 90–92.

[15] Tavallaee, Mahbod, et al. "A Detailed Analysis of the KDD CUP 99 Data Set." 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009

[16] "Network Intrusion Detection." NYC Data Science Academy Blog, nycdatascience.com/blog/student-works/network-intrusion-detection-2.