International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Analysis of an Ensemble Model for Network Intrusion Detection

Rahul R S[1], Rithvik M[2], Gururaja H S[3], Vikram K[4]

[1]Department of Information Science, B.M.S College of Engineering, Bengaluru, India
rahulrs.is17[at]bmsce.ac.in

[2]Department of Information Science, B.M.S College of Engineering, Bengaluru, India
rithvikmohan.is17[at]bmsce.ac.in

[3]Department of Information Science, B.M.S College of Engineering, Bengaluru, India
gururajhs[at]bmsce.ac.in

[4]Department of Information Science, B.M.S College of Engineering, Bengaluru, India
vikram.is17[at]bmsce.ac.in

Abstract: ***Network security is extremely important and mission-critical not just only for business continuity but also for thousands of other huge and increasing number of systems and applications running over network continuously to deliver services. One of the ways network security is implemented and enforced is via intrusion detection or prevention systems. Traditional intrusion detection systems are usually rule-based and are not effective in detecting new and previously unknown intrusion events. Data mining techniques and machine algorithms have recently gained attention as an alternative approach to proactively detect network security breaches. In this project, these data mining algorithms: Decision Tree and Random Forest, Naive Baye, K-Nearest Neighbor (KNN) and Logistic Regression classifiers were implemented to detect and classify network intrusion using NSL-KDD dataset. The results obtained generally indicate that models are biased towards classes with low distribution in the dataset.***

Keywords: data science, machine learning, network intrusion, IDS, naive bayes, decision tree, NLS, KDD, K-Nearest Neighbor, KNN, data mining, random forest, cybersecurity, computer science, network security, ensemble model

1. Introduction

Intrusion generally refers to malicious activities directed at computer network system to compromise its integrity, availability and confidentiality. Network security is important because modern information technology relies on it to drive businesses and services. Security can be enforced on the network through intrusion detection systems (IDS). These are security devices or software usually implemented by large and medium organizations to enforce security policies and monitor network perimeter against security threats and malicious activities. Other associated systems include Firewall and Intrusion Prevention System (IPS). Essentially, intrusion detection device or application scrutinizes every incoming or outgoing network traffic and analyses packets (both header and payload) for known and unknown events. Detected known events and violations are logged usually in a central security information and event management (SIEM) system. Malicious activities or unknown events may be set up to alert system administrator or the related packets dropped depending on the configurations enabled on the intrusion detection system.

Prevention of security breaches cannot be completely avoided. Hence, effective intrusion detection becomes important for organizations to proactively deal with security threats in their networks. However, many existing intrusion detection systems are rule-based and are not quite effective in detecting a new intrusion event that has not been encoded in the existing rules. Besides, intrusion detection rules development is time consuming and it is limited to knowledge of known intrusions only.

Data mining techniques, on the other hand, through supervised and unsupervised learning algorithms have been shown to be effective in identifying and differentiating known and new intrusions from network event records or data. It is therefore worthwhile to explore application of data mining techniques as an effective alternative approach to detect known and potential network intrusions. To make the experience as organic as possible after the gesture has been detected, it relays the output to an audio device to speak the translated output. This makes the communication feel more natural instead of just reading out text on a screen.

2. Literature Survey

With the popularity of Internet and due to the widespread usage of networks, the number of attacks has increased, and numerous hacking tools and intrusive methods have gained traction. Within a network, one way to counter suspicious activity is by using an intrusion detection system (IDS).

IDSs can be termed as misuse/anomaly detectors by sorting out broadly based on the models of their detection. Mishandling detectors depend on understanding the models of known attacks, whereas irregularity detection creates profiles for users as the key use case of detection, and sorts the uniqueness of the normal and abnormal (anomaly) ones as incursion [1].

Conversely, the sheer amount of the intrusions increased drastically as the speed and complexity of networks expand swiftly, this is seen when such networks are unlocked/opened to the general public. Intrusion detection aims to catch network attacks. To try and work out network

security problems, it is one of the essential ways. The two major signs to examine intrusion detection systems (IDS) are Detection precision and stability. It is becoming difficult for any intrusion detection system to suggest a trustworthy repair with the varying technology and the huge growth of Internet traffic it has been created that a behavioral model is present in the attacks that can be gotten to know from former study.[3]

Classes:	DoS	Probe	U2R	R2L
Sub-Classes:	• apache2 • back • land • neptune • mailbomb • pod • processtable • smurf • teardrop • udpstorm • worm	• ipsweep • mscan • nmap • portsweep • saint • satan	• buffer_overflow • loadmodule • perl • ps • rootkit • sqlattack • xterm	• ftp_write • guess_passwd • httptunnel • imap • multihop • named • phf • sendmail • Snmpgetattack • spy • snmpguess • warezclient • warezmaster • xlock • xsnoop
Total:	11	6	7	15

3. Project Implementation

a) KDD Dataset

The dataset used in this project is the NSL-KDD dataset from the University of New Brunswick, Canada. The dataset is an improved version of the KDDCUP'99 dataset from DARPA Intrusion Detection Evaluation Program. The original KDD dataset is perhaps the most widely used dataset for machine learning intrusion detection tasks. However, the results of statistical analysis conducted by Tavallaee et al. revealed that the dataset is fraught with redundant records that can lead to poor evaluation of anomaly detection tasks. A new dataset, NSL-KDD, devoid of the deficiencies noted in KDDCUP'99 data has since been proposed by Tavallaee et al. The NSL-KDD dataset used in this project consists of 125,973 records training set and 22,544 records test set with 42 attributes/features for each connection sample including class label containing the attack types.[7]

b) Data Load and Pre-processing

Mapping intrusion types to attack classes:

After loading the dataset into Python (Jupyter Notebook) development environment, the first task performed was mapping various attack types in the dataset into four attack classes as described by Tavallaee et al.

1) *Denial of Service (DoS):* is an attack in which an adversary directed a deluge of traffic requests to a system in order to make the computing or memory resource too busy or too full to handle legitimate requests and in the process, denies legitimate users access to a machine.
2) *Probing Attack (Probe):* probing network of computers to gather information to be used to compromise its security controls.
3) *User to Root Attack (U2R):* a class of exploit in which the adversary starts out with access to a normal user account on the system (gained either by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to
4) gain root access to the system.
5) *Remote to Local Attack (R2L):* occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine.

Exploratory Data Analysis

Basic exploratory data analyses were carried out among other things to understand the descriptive statistics of the dataset, find instances of missing values and redundant features, explore the data type and structure and investigate the distribution of attack class in the dataset.

	Class Distribution	
	Before Sampling	After Sampling
Normal	67343	67343
DoS	45927	67343
Probe	11656	67343
L2R	995	67343
U2R	52	67343

Data Sampling Table

Intrusion types and subclasses

Standardization of Numerical Attributes

The numerical features in the dataset were extracted and standardized to have zero mean and unit variance. This is a common requirement for many machine learning algorithms implemented in Scikit-learn python module

Enconding Categorical Attributes

The categorical features in the dataset were encoded to integers. This is also a common requirement for many machine learning algorithms implemented in Scikit-learn.[12]

a) Data Sampling

The sparse distribution of certain attack classes such as U2R and L2R in the dataset while others such as Normal, DoS and Probe are significantly represented inherently leads to the situation of imbalance dataset. While this scenario is not unexpected in data mining tasks involving identification or classification of instances of deviations from normal patterns in a given dataset, research has shown that supervised learning algorithms are often biased against the target class that is weakly represented in a given dataset.

Certain approaches such as random data sampling and cost-sensitive learning method [8] have been suggested to address the problem of imbalance dataset. The use of sampling involves modification of an imbalanced data set by some mechanisms in order to provide a balanced distribution. While oversampling replicates and increases data in class label with low distribution, random under sampling removes data from the original dataset with high class frequency [4]

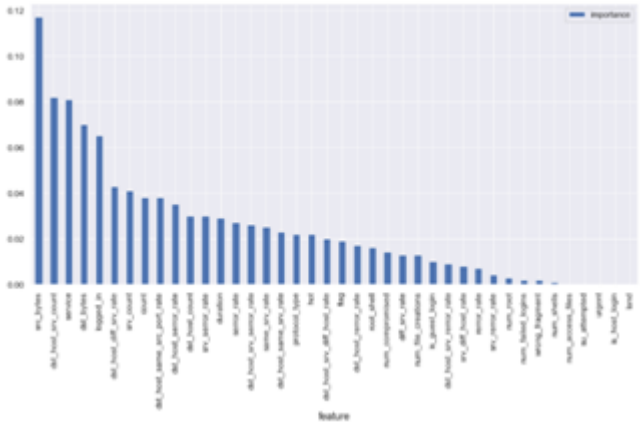
Cost-sensitive learning focuses on the imbalanced learning problem by using different cost matrices that describe the costs for misclassifying any particular data example rather than creating balanced data distributions through different sampling techniques [8]. Studies have shown that for several base classifiers, a balanced data set provides improved overall classification performance compared to an imbalanced data set [9]. Popular sampling techniques includes synthetic minority oversampling technique (SMOTE) and Random Oversampling and Under sampling.

b) Feature Selection and Data Partioning

Feature selection is a key data preprocessing step in data mining task involving selection of important features as a subset of original features according to certain criteria to reduce dimension in order to improve the efficiency of data mining algorithms. Most of the data includes irrelevant, redundant, or noisy features. Feature selection reduces the number of features, removes irrelevant, redundant, or noisy features, and brings about palpable effects on applications: speeding up a data mining algorithm, improving learning accuracy, and leading to better model comprehensibility [10].

There are two common approaches to select or reduce the features; a wrapper uses the intended learning algorithm itself to evaluate the usefulness of features, and a filter evaluates features according to heuristics based on general characteristics of the data.[11]

The wrapper approach is generally considered to produce better feature subsets but runs much more slowly than a filter. In this project, a wrapper approach was used wherein a Random Forest classifier algorithm [2] with a function to define feature importance was trained to extract feature importance from the training dataset. A second step implemented involved using a recursive feature extraction also based on Random Forest algorithm to extract top 10 features relevant to achieve accurate classification of classes in the training set.



Feature Selection and Importance

After the selection of features, the dataset that had been then resampled and partitioned into two target classes (normal and attack) for all the attack classes in the dataset to make way for binary classification. For multiclass classification no such partition was required and this process was skipped for that approach.

c) Train Models

The training dataset was utilized to train the following classifier algorithms: Support Vector Machine (SVM), Decision Tree, Naive Bayes, Logistic Regression and k-Nearest Neighbour (KNN) [5][6]. Also, an **ensemble classifier** was trained to add in and average out the prediction results from the individually made classifiers. The ingenuity behind the classifier is to combine conceptually different machine learning models and utilize a major vote (hard vote) or the average prediction probability (soft vote) to predict the labels for each class. Such a classifier can be useful for a list of equally well performing classifiers so as to almost equalize their unique weaknesses.

Two approaches were employed, one being a multiclass classification where the dataset as a whole with the 5 classes (i.e., Normal, DoS, Probe, R2L, U2R) was used to train the models. Since Support Vector Machine and Logistic Regression are primarily designed for binary classification, hence these 2 methods were omitted in this approach. The second approach involved creation of separate binary classifiers for each attack group. To achieve this the dataset is partitioned into 4 groups each with 2 classes (Normal and an attack group) as mentioned earlier. For each such group all the above-mentioned classifier algorithms along with the ensemble of these classifiers were trained. This improves the detection rates at the cost of a model being able to detect only the attack group it was trained on.

d) Evaluate Models

The trained models were evaluated using a 5-fold cross validation technique. The concept of k-fold cross validation involves generation of validation set out of training dataset to assess the model before it is exposed to test data. In k-fold cross-validation, the training dataset is randomly divided into k equal sized subsamples. Out of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k − 1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged or combined to generate a single value. The advantage of this method is that all samples are used for both training and validation

4. Results and Discussions

The models that were trained were utilized to classify labels in a test dataset that was not previously shown to the algorithms. Performance metrics such as accuracy, confusion matrix and classification report were generated for each of the models for both the binary classifier and multiclass classifier methods. From the results obtained, all the models evaluated achieved an average of 99% on training set while model's performance on test set indicates an average of more than 80% across all the four attack groups investigated. The test accuracy for 'Normal_U2R' across all the models showed a value of more than 90%. However, a review of performance as shown by the confusion matrix indicated that the 'U2R' detection rate was very poor across Decision Tree, Random Forest, KNN and Logistic Regression models. The attack class 'R2L' detection rate is also not far from being poor as 'Normal' and other classes with higher distribution got more attention of the models to the detriment of the low distribution classes. The results generally showed that sampling may improve model training accuracy. However, a poor detection rate in detecting U2R and R2L minority attacks are inevitable because of their large bias available in the dataset. Here follow the results for both the binary and multiclass classifier:

	Normal_DoS		Normal_Probe	
	Trained Model Evaluation	*Model Test Performance*	*Trained Model Evaluation*	*Model Test Performance*
	Detection Accuracy (%)	Detection Accuracy (%)	Detection Accuracy (%)	Detection Accuracy (%)
SVM	98.91	83.66	98.20	85.98
Naive Bayes	97.37	83.36	95.20	82.73
Decision Tree	98.75	81.65	99.99	80.17
Random Forest	99.33	82.95	99.99	82.05
KNN	98.77	85.06	99.82	85.52
Logistic Regression	98.08	84.18	96.80	84.02
Voting Classifier	99.85	85.13	97.26	87.81

	Normal_R2L		Normal_U2R	
	Trained Model Evaluation	*Model Test Performance*	*Trained Model Evaluation*	*Model Test Performance*
	Detection Accuracy (%)	Detection Accuracy (%)	Detection Accuracy (%)	Detection Accuracy (%)
SVM	97.75	79.94	99.52	97.83
Naive Bayes	94.58	75.91	93.32	91.03
Decision Tree	95.59	78.20	99.99	97.98
Random Forest	99.93	77.90	99.99	97.98
KNN	99.83	78.69	99.96	97.94
Logistic Regression	96.55	79.85	95.84	97.97
Voting Classifier	99.78	80.87	99.92	98.96

Binary classifier detection rates

SVM

Confusion Matrix		Predicted Group: DoS	Predicted Group: Normal
Actual Group	DoS	5272	2186
	Normal	619	9092

Confusion Matrix		Predicted Group: Normal	Predicted Group: Probe
Actual Group	Normal	8801	910
	Probe	790	1631

Confusion Matrix		Predicted Group: Normal	Predicted Group: R2L
Actual Group	Normal	9707	4
	R2L	2496	258

Confusion Matrix		Predicted Group: Normal	Predicted Group: U2R
Actual Group	Normal	9692	19
	U2R	196	4

SVM Confusion Matrix

Naive Bayes

Confusion Matrix		Predicted Group: DoS	Predicted Group: Normal
Actual Group	DoS	5487	1971
	Normal	885	8826

Confusion Matrix		Predicted Group: Normal	Predicted Group: Probe
Actual Group	Normal	8133	1578
	Probe	517	1904

Confusion Matrix		Predicted Group: Normal	Predicted Group: R2L
Actual Group	Normal	8963	748
	R2L	2254	500

Confusion Matrix		Predicted Group: Normal	Predicted Group: U2R
Actual Group	Normal	8982	729
	U2R	160	40

Naïve Bayes Confusion Matrix

Decision Tree

Confusion Matrix		Predicted Group: DoS	Predicted Group: Normal
Actual Group	DoS	5591	1867
	Normal	1282	8429

Confusion Matrix		Predicted Group: Normal	Predicted Group: Probe
Actual Group	Normal	7658	2053
	Probe	352	2069

Confusion Matrix		Predicted Group: R2L	Predicted Group: Normal
Actual Group	R2L	9082	629
	Normal	2088	666

Confusion Matrix		Predicted Group: Normal	Predicted Group: U2R
Actual Group	Normal	9711	0
	U2R	200	0

Decision Tree Confusion Matrix

Random Forest

Confusion Matrix		Predicted Group: DoS	Predicted Group: Normal
Actual Group	DoS	5489	1969
	Normal	958	8753

Confusion Matrix		Predicted Group: Normal	Predicted Group: Probe
Actual Group	Normal	8519	1192
	Probe	985	1436

Confusion Matrix		Predicted Group: Normal	Predicted Group: R2L
Actual Group	Normal	9711	0
	R2L	2754	0

Confusion Matrix		Predicted Group: Normal	Predicted Group: U2R
Actual Group	Normal	9711	0
	U2R	200	0

Random Forest Confusion Matrix

Logistic Regression

Confusion Matrix		Predicted Group: DoS	Predicted Group: Normal
Actual Group	DoS	5963	1495
	Normal	1220	8491

Confusion Matrix		Predicted Group: Normal	Predicted Group: Probe
Actual Group	Normal	8370	1341
	Probe	597	1824

Confusion Matrix		Predicted Group: Normal	Predicted Group: R2L
Actual Group	Normal	9542	169
	R2L	2342	412

Confusion Matrix		Predicted Group: Normal	Predicted Group: U2R
Actual Group	Normal	9708	3
	U2R	198	2

Logistic Regression Confusion Matrix

K-Nearest Neighbor

Confusion Matrix		Predicted Group: DoS	Predicted Group: Normal
Actual Group	DoS	5787	1671
	Normal	619	9092

Confusion Matrix		Predicted Group: Normal	Predicted Group: Probe
Actual Group	Normal	9024	687
	Probe	1069	1352

Confusion Matrix		Predicted Group: Normal	Predicted Group: R2L
Actual Group	Normal	9650	61
	R2L	2595	159

Confusion Matrix		Predicted Group: Normal	Predicted Group: U2R
Actual Group	Normal	9707	4
	U2R	200	0

K-Nearest Neighbor Confusion Matrix

Ensemble Model

Confusion Matrix		Predicted Group: DoS	Predicted Group: Normal
Actual Group	DoS	5604	1854
	Normal	699	9012

Confusion Matrix		Predicted Group: Normal	Predicted Group: Probe
Actual Group	Normal	8766	945
	Probe	533	1888

Confusion Matrix		Predicted Group: Normal	Predicted Group: R2L
Actual Group	Normal	9705	6
	R2L	2567	187

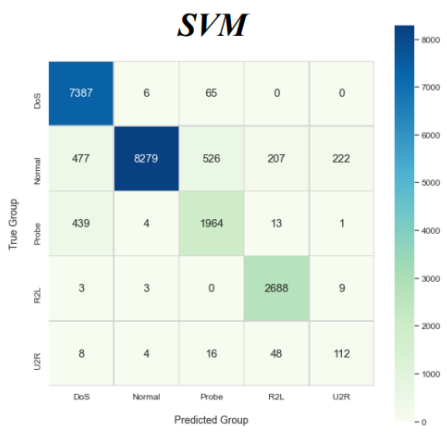
Confusion Matrix		Predicted Group: Normal	Predicted Group: U2R
Actual Group	Normal	9711	0
	U2R	200	0

Ensemble Confusion Matrix

Decision Rates for Multiclass classifier

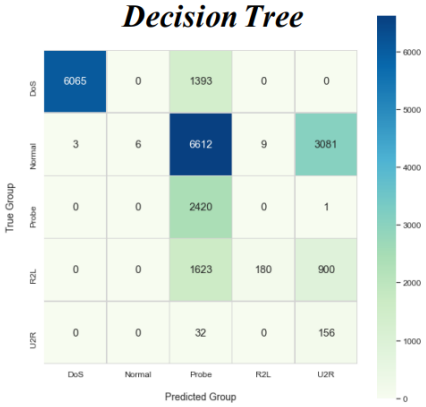
	Trained Model Evaluation	Model Test Performance
	Detection Accuracy (%)	Detection Accuracy (%)
Naïve Bayes	85.24	90.08
Decision Tree	84.74	39.26
Random Forest	87.08	66.24
KNN	95.76	87.80
Voting Classifier	96.23	93.03

Multiclass classifier Decision Rates

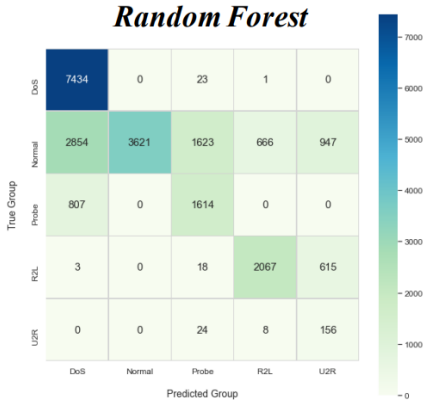


Confusion Matrix Model Test Performance for SVM

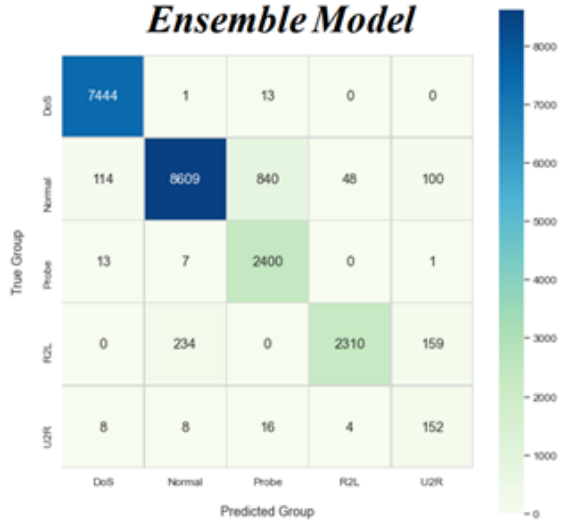
Confusion Matrix Model Test Performance for KNN



Confusion Matrix Model Test Performance for Decision Tree



Confusion Matrix Model Test Performance for Random Forest



Confusion Matrix Model Test Performance for Ensemble Model

5. Conclusion

One of the key lessons learned in this project is that, for classification models, accuracy is not a good measure of a model performance where there is an imbalance dataset. Accuracy value may be high for the model but the class with lower samples may not be effectively classified. If input data is such that 10% of the samples represent attacks and the model predicts all the data samples not to be an attack then theoretically it would have an accuracy of 90%, but obviously such a model is in fact of no use since it failed to detect any attacks. Thus, other metrics such as Confusion Matrix is more realistic in evaluating classifier model performance than accuracy measure.

While building these models in the course of this project, we saw that the amount of data points and classification types for attacks like R2L and U2R were significantly in lesser numbers than the other categories. Hence, to not allow such an attack from being neglected, more data needs to be collected for these attack types for a better and a more well-rounded model. As of now, it is harder to make accurate predictions using limited data.

The use of machine learning models has revolutionized the measure and accuracy to which we can make predictions of attacks. To create an infallible system, one might need to use even more complex models such as- ANN and Deep learning techniques. These models would take a much deeper dive into the computational arithmetic that is involved and will come out with better results. These better results do come at a price though, with these so called- much better models having a very high cost of computational demands and advanced hardware in the running systems. We would need specially dedicated systems with advanced hardware to be able to run these deep learning or ANN models.

Overall, the scope for the future holds exciting innovations to explore and achieve better results.

References

[1] S. Peddabachigiri, A. Abraham., C. Grosan and J. Thomas, "Modeling of Intrusion Detection System Using Hybrid Intelligent Systems" , Journals of Network Computer Application, 2007. Available from: https://www.researchgate.net/publication/222527188_Modeling_intrusion_detection_system_using_hybrid_intelligent_systems [accessed Jul 11, 2017].
[2] J. Zhang, M. Zulkernine, A. Haque "Random-Forests-Based Network Intrusion Detection Systems", IEEE Trans. Syst. Man Cybernet. Part C Appl. Rev. 8:649–659, 2008.
[3] M. Panda, A. Abraham, S. Das, M.R. Patra, "Network Intrusion Detection System: A Machine Learning Approach", Intelligent Decision Technologies 5(4), 347–356, 2011. Available from: https://www.researchgate.net/publication/220468036_Network_intrusion_detection_system_A_machine_learning_approach [accessed Jul 11, 2017].
[4] Paul, D, et al. "Data mining for network intrusion detection." Proc. NSF Workshop on Next Generation Data Mining. 2002.
[5] Data Science Association, "Introduction to Machine Learning", The Wikipedia Guide, 2016. Avaialable from: http://www.datascienceassn.org/content/introduction-machine-learning [accessed Aug 10 2017].
[6] M. Swamynathan "Mastering Machine Learning with Python in Six Steps", ISBN-13: 978-1-4842-2865-4, Apress, 2017.
[7] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani, "A Detailed analysis of the KDD CUP 99 Data Set". In the Proc. Of the IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009), pp. 1-6, 2009.
[8] H. He and E.A. Garcia, "Learning from Imbalanced Data", IEEE Transactions On Knowledge And Data Engineering, Vol.21, No.9, 2009.
[9] G.M. Weiss and F. Provost, "The Effect of Class Distribution on Classifier Learning: An Empirical Study," Technical Report ML-TR-43, Dept. of Computer Science, Rutgers Univ., 2001.
[10] Y. Kim, W. N. Street, F. Menczer, and G. J. Russell, "Feature selection in data mining" in Data Mining: Opportunities and Challenges: J. Wang, Ed. Hershey, PA: Idea Group Publishing, 2003, pp. 80–105.
[11] Liu H ,Setiono R, Motoda H, Zhao Z, Feature Selection: An Ever Evolving Frontier in Data Mining, JMLR: Workshop and 17 Conference Proceedings 10, pp. 4-13, 2010.
[12] Pedregosa et al., "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research 12, pp. 2825-2830, 2011.