Assembly Classifier Approach to Analyze Intrusion Detection Dataset in Networks by Using Data Mining Techniques

Ayad Mohammed Mahyoub Ghaleb[1], Samani A. Talab[2]

[1]Nilean University, College of Computer Sciences, Department of Technology of Information, Khartoum, Sudan

[2]Professor, Nilean university, College of Computer Sciences, Department of technology of information, Khartoum, Sudan

Abstract: ***The Intrusion Detection system analyzes network traffic to detect the attacks. The attack detection methods used by these systems are of two types: anomaly detection and misuse detection methods. Intrusion detection is a type of security management system forcomputer networks. An Intrusion detection system analyzes information within a computer network to identify possible security breaches, which include both anomaly and misuse. In this paper I studied the performance of number of data mining algorithms and chose best three algorithms for building multi classifier from decision tree classifier, naïve Bayes classifier and Multilayer Perceptron classifier.I evaluated performance classifier by account accuracy and error rate.***

Keywords: Intrusion Detection, Intrusion Detection System (IDS), Probe attacks, Dos (Denial of Service) attacks, R2L(Remote to Local) attack, U2R (User to Root) attack, Weka, NLS-KDD 99 data set.

1. Introduction

Intrusion Detection concept was introduced by James Anderson in 1980[1], defined an “Intrusion attempt or threat to be potential possibility of a deliberate unauthorized attempt to access information, manipulate information, or render a system unreliable or unusable”.

Security of a network is important; it monitors all traffic passing on the segment in network. Protecting the network against intruder’s confidentiality, Integrity, Availability, Authentication and Nonrepudiation are the objectives for IDS.

Anderson discussed a frame work investigation of intrusions and intrusion detection. He discussed definition of fundamental terms Threat, Risk, Vulnerability, Attack and Penetration.

1) **Risk:** Accidental or unpredictable violation of operations integrity or exposure of information due to the malfunction of hardware or incomplete or incorrect software design.
2) **Threat:** The potential possibility of unauthorized, a deliberate attempt to:
3) Access information, manipulate information, and render a system unusable or unreliable [2].
4) **Vulnerability:** A known or suspected flaw in the operation or software orhardware of a system that exposes the system to penetration or its information to accidental disclosure.
5) **Attack:** Execution of a plan or specificformulation to carry out a threat. Penetration: A successful attack, the ability to obtain undetected access to control state of a computer system or files and programs.

2. Intrusion Detection

Identifying unauthorized use, attacks and misuse on information systems is defined as intrusion detection [3] [4]. The most popular way to detect intrusions has been done by using audit data generated by operating systems and by networks. Since all activities are logged on a system, it is possible that a manual inspection of these logs would allow intrusions to be detected. An intrusion detection system can be used to analyze audit data for such insights. This makes IDS a valuable real-time detection and prevention tool as well as a forensic analysis tool.

2.1 Misuse Detection

The idea of misuse detection is to represent attacks in the form of a pattern or a signature so that the same attack can be detected and prevented in future. These systems can detect many or all known attack patterns [4], but they are of little use for detecting naive attack methods. The main issues of misuse detection is how to build signatures that include possible signatures of attacks build a signature that includes all possible variations of the pertinent attack to avoid false negatives, and how to build signatures that do not match non-intrusive activities to avoid false positives.

2.2 Anomaly Detection

The idea here is that if we can establish a normal activity profile for a system, in theory we can flag all system states varying from the established profile as intrusion attempts. However, if the set of intrusive activities is not identical to the set of anomalous activities, the situation becomes more interesting instead of being exactly the same, we find few interesting possibilities. False positives are anomalous activities that are not intrusive are flagged as intrusive. False negatives are actual intrusive activities that go undetected. One of the main issues of anomaly detection systems is the selection of threshold levels so that neither of the above problems is unreasonably magnified. Anomaly detection is usually computationally expensive because of the overhead of keeping track of and possibly updating several system profiles [5].

Intrusion NLS-KDDCUP'99 dataset

Since 1999, KDD'99 [3] has been the most wildly used data set for the evaluation of intrusion detection methods. KDD' 99 is prepared by Stolfo et al [5] and is built based on the data captured in DARPA'98 intrusion detection system evaluation program [6].NLS-KDD training dataset consists of approximately 4,900,000 single connection vectors each of which contains 41 features and is labeled as either normal or an attack, with exactly one specific attack type. The simulated attacks fall in one of four categories Denial of Service, Remote to User, User to Root and Probing.KDD'99 features can be classified into three groups:

1) **Basic features** category encapsulates all the attributes that can be extracted from a TCP/IP connection. These features leading to an implicit delay in detection.
2) **Traffic features** category includes features that are computed with respect to a window interval and isdivided into two groups "Same host and "same service" features
3) **Content features** unlike most of the DOSandProbing attacks, the R2L and U2R attacks don't have any intrusion frequent sequential patterns. This is because the DOS and Probing attacks involve many connections to some hosts in a very short period of time; however the R2L and U2R attacks are embedded in the data portions of the packets, and normally involves only a single connection. To detect these kinds of attacks, we need some features to be able to look forsuspicious behavior in the data portion, e.g., number of failed login attempts [6].

Attributes in NLS-KDD99 dataset divided into three groups of features:
1) Basic features of network connection, which includes the service, duration, prototype, number of bytes from source IP addresses or from destination IP addresses, and some flags in TCP connections**.**
2) Second group of attributes in NLS-KDD99 is composed of the content features of network connections.
3) Third group is composed of the statistical features that are computed either by a time window or a window of certain kind of connections.

The attributes selection in NLS-KDD99 dataset has been widely used as a standard method for network-based intrusion detection learning, and it was found that all 41 attributes of NLS-KDD99 dataset are not the best ones for intrusion detection learning. Therefore the performance of intrusion detection system may be further improved by studying new attribute selection methods [7]

Probing

Is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits. There are different types of probes someof them abuse the computer's legitimate features and some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise. Different types of probe attacks are shown in Table 1.

Table 1: Different types of probe attacks

Attack type	*Service*	*Mechanism*	*Effect of the attack*
Ipsweep	Icmp	Abuse of feature	Identifies active machines
Mscan	Many	Abuse of Feature	Looks for known Vulnerabilities
Nmap	Many	Abuse of Feature	Identifies active ports on a Machine.
Saint	Many	Abuse of Feature	Looks for known Vulnerabilities
Satan	Many	Abuse of Feature	Looks for known Vulnerabilities

3. Denial of Service Attacks (DOS)

Is attacker makes some computing or memory resource too full or too busy to handle legitimate requests, thus denying legitimate users access to a machine. There are different ways to launch denial of service attacks:
1. By abusing the computers, legitimate features.
2. By targeting the implementations bugs.
3. By exploiting the system's misconfigurations.

DOS attacks are classified based on the services that an attacker renders unavailable to legitimate users.

Table 2: Different types of DOS attacks

Attack type	Service	Mechanism	Effect of the attack
Apache2	http	Abuse	Crashes httpd
Back	http	Abuse/Bug Slows	Slows down server response
Land	http	Bug	Freezes the machine
Mail	bomb	N/A Abuse	Annoyance
SYN flood	TCP	Abuse	Denies service on one or more ports
Ping of death	Icmp	Bug	None
Process table	TCP	Abuse	Denies new processes
Smurf	Icmp	Abuse	Slows down the network
Syslogd	Syslog	Bug	Kills the Syslogd
Teardrop	N/A	Bug	Reboots the machine
Udpstrom	Echo/ Chargen	Abuse	Slows down the network

3.1 User to root attack (U2R)

Is a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Most common exploits in this class of attacks are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions.

Table 3: Different types of U2R attacks

Attack type	Service	Mechanism	Effect of the attack
Eject	User session	Buffer overflow	Gains root shell
Ffbconfig	User session	Buffer overflow	Gains root shell
Loadmodule	User session	Poor environment	Gains root shell

		Sanitation	
Perl	User session	Poor environment Sanitation	Gains root shell
Ps	User session	Poor temp file management	Gains root shell
Xterm	User session	Buffer overflow	Gains root shell

3.2 Remote to user attack s(R2L)

Is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine's vulnerability to illegally gain local access as a user. There are different types of R2U attacks: the most common attack in this class is done using social engineering.

Table 4: Different types of R2L attacks

Attack type	Service	Mechanism	Effect of the attack
Dictionary	Telnet, rlogin, pop, ftp, imap	Abuse feature	Gains user Access
Ftp-write	Ftp	Misconfig	Gains user Access
Guest	Telnet, rlogin	Misconfig	Gains user access
Imap	Imap	Bug	Gains root access
Named	Dns	Bug	Gains root access
Phf	Http	Bug	Executes commands as http user
Sendmail	Smtp	Bug	Executes commands as root
Xlock	Smtp	Misconfig.	Spoof user to obtain password
Xnsoop	Smtp	Misconfig.	Monitor key stokes remotely

3.3 Other Attacks

These are attacks R2lclass not presents in abovee.g: snmpget attack, mailbomb, snmpguess, mscan [8]. Corresponding to the four attack groups (Probe, DoS,R2L, and U2R) and other attacks given in the KDD 99 Data Set I select different features for different layers based upon the type of attack the layer is trained to detect. Hence I have a four independent modules corresponding to the four attack groups. I am selecting different features to train different layers in our framework. Hence, I use domain knowledge to select features for all the four attack classes see appendixes.

4. Machine learning algorithms applied to intrusion detection

4.1 C4.5 Algorithm

It is supervised learning. It a mapping from attribute values to classes that can be applied to classify new and unseen instances. This algorithm is more applicable for continuous and discrete value attributes [9].
Intrusion Detection Algorithm Based on C4.5 Intrusioncan be divided into three stages [10]:

Stage 1: Construct decision tree
Algorithm: C4.5 Tree generates a decision tree from the given training data. Input: training sample set T, the collection of candidate attribute. Attribute-list. Output: A decision tree. Create a root *node N;*

1) if T belong to the same category C, then return N as a leaf node, and mark it as a class C.
2) if a trribute-list is empty or the remainder sample of T is less than a given value, then return N as a leaf node, and mark it as a category which appears most frequently.
3) In attribute-list, for each attribute, calculate its information gain ratio.
4) Suppose test-attribute is the testing attribute of N, then test attribute=the attribute which has the highest information gain ratio in attribute-list .
5) if the testing attribute is continuous, then find its division threshold.
6) for each new leaf node grown by node N.
7) Calculate the classification error rate of each node, and then prune the tree.

Stage 2: Extract classification rules.
For decision tree, each branch represents a test output, and each leaf node represents category or category distribution. We just need to follow every path from root node to leaf node, the conjunction of each attribute-value constitutes the antecedent of rules, and the leaf node constitutes the consequent of rules. So decision tree can easily be converted into IF-THEN rules.

Stage 3: Determine network behavior.
For new network behavior, determine whether it intrudes or not according to classification rules. In building decision tree, there are two different methods for pruning it: pre-pruning and post-pruning. The power of post-pruning is obvious in situations in which two attributes individually seem to have nothing to contribute, but they are robust predictor when fused [11]. There are three post-pruning techniques: sub-tree replacement, sub tree raising, and reduced error pruning.
Weka classifier package has its own version of C4.5 known as J48 optimized implementation of C4.5 rev. 8.

4.2 NaïveBays classifier

The NaïveBays [12] classifier provides a simple approach, with clear semantics, to learning and representing probabilistic knowledge. It is termed naïve because is relies on two important simplifying assumes that the predictive attributes are conditionally independent given the class, and it posits that no hidden or latent attributes influence the prediction process.

4.3 Multilayer Perceptron (MLP):

Multilayer perceptron (MLP) [12] is one of the most commonly used neural network classification algorithms. The architecture used for the MLP during simulations with KDD dataset consisted of a three layer feed-forward neural network: one input, one hidden, and one output layer. Selected parameters for the model are: learning Rate = 0.3; momentum = 0.2; random Seed = 0; validation Threshold = 20.

4.5 Why doAssembly work?

Dietterich(2002) showed that Assembly overcome three problems:

- **The Statistical Problem** arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data!.
- **The Computational Problem** arises when the learning algorithm cannot guarantee finding the best hypothesis.
- **The Representational Problem** arises when the hypothesis space does not contain any good approximation of the target class(es)[13].

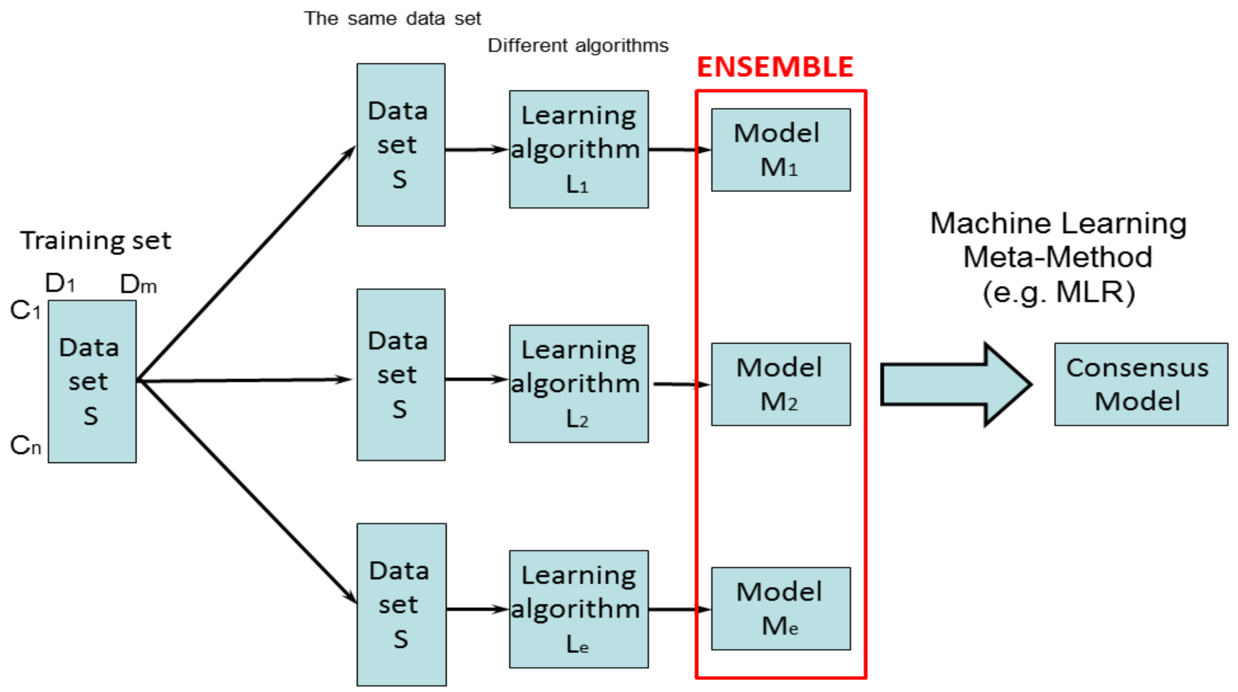

Figure 1: assembly classifier

..periments

4.5 Preprocessing dataset

The huge number of redundant records is one of the most important deficiencies in the KDD data set,so that the learning algorithms to be biased towards the frequent records, and thus prevent learning non redundant records such as U2R and R2L attacks which are usually more harmful to networks. In addition, the existence of these redundant records in the test set will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records[14]. Hence preprocessing was required before pattern classification models could be built. Preprocessing consisted of three steps:

First step I eliminated the single valued attributes *num_ outbound_cmds and is host login*

Second step involved mapping symbolic-valued attributes to numeric-valued attributes. Attack names *(like buffer_overflow, guess_passwd, etc.)* were first mapped to one of the five classes, 0 for Normal, 1 for Probe, 2 for DoS, 3 for U2R, and 4 for R2L, as described in [15]. Symbolic features like protocol_type (3 different symbols), service (70 different symbols), and flag (11 different symbols) were mapped to integer values ranging from 0 to N-1 where N is the number of symbols.

Third step implemented scaling: each of these features was linearly scaled to the range [0.0, 1.0]. Features having smaller integer value ranges like *duration [0, 58329], wrong_fragment [0, 3], urgent [0, 14], hot [0, 101], num_failed_logins [0, 5], num_compromised [0, 9], su_attempted [0, 2], num_root [0, 7468], num_file_creations [0, 100], num_shells [0, 5], num_access_files [0, 9], count [0, 511], srv_count [0, 511], dst_host_count [0, 255], and dst_host_srv_count [0, 255]* were also scaled linearly to the range [0.0, 1.0]. All other features were either boolean, like logged_in, having values (0 or 1), or continuous, like diff_srv_rate, in the range [0.0, 1.0]. Hence scaling was not necessary for these attributes.

The KDD 1999 Cup dataset has a very large number of duplicate records[6].. For the purpose of training different classifier models, these duplicates were removed from the datasets. The total number of records in the original labeled training dataset is 972,780 for Normal, 41,102 for Probe, 3,883,370 for DoS, 52 for U2R, and 1,126 for R2L attack classes. After filtering out the duplicate records, there were a total of 812,813 records for Normal, 13,860 for Probe, 247,267 for DOS, 52 for U2R, and 999 for R2L attack. In this study the total number ofrecords of training NLS-KDD 99 data set show in table(4a) below and total number of records of testing data set show in table(4b) below:

Table (4a): The training records

Attack type	Training records
Normal	3449
DOS	9234
Probe	2289
R2L	209
U2R	11
Total	25190

Table (4b): The testing records

Attack type	Testing records
Normal	9711
DOS	**7458**
Probe	2421
R2L	2887
U2R	67
Total	22544

4.6 Experiments using C4.5 algorithm

Perhaps C4.5 algorithm is the most popular tree classifier. Weka classifier package has it's own version of C4.5 known as J48. J48 is an optimized implementation of C4.5 rev. 8. J48 is experimented is this study with the parameters: confidence Factor = 0.25; numFolds = 3; seed = 1; unpruned = False table (5) chow

Table 5: Result of experiment using C4.5 classifier

Attack type	TP	FP	Precision	Accuracy
Normal	0.97	0.37	0.67	0.97
DOS	0.81	0.03	0.94	0.81
PROBE	0.59	0.03	0.71	0.59
R2L	0.001	0	0.33	0.001
U2R	0.00	0	0	0.00

4.7 Experiments using NaïveBays

The NaïveBays classifier provides a simple approach, with clear semantics, to representing and learning probabilitistic knowledge. It is termed naïve because is relies on two important simplifying assumes that the predictive attributes are conditionally independent given the class, and it posits that no hidden or latent attributes influence the prediction process.

Table 6: Result of experiment using NaïveBays classifier

Attack type	TP	FP	Precision	Accuracy
Normal	0.86	0.24	0.73	0.86
DOS	0.71	0.37	0.91	0.71
PROBE	0.92	0.06	0.65	0.92
R2L	0.09	0.01	0.67	0.09
U2R	0.33	0.06	0.02	0.33

4.8 Experiments using multilayer perceptronnetworks(MLP)

The network was set to train until the desired mean square error of 0.001 was met. During the training process the goal was met at 100 epochs for backpropagation. As multilayer perceptron networks are capable of multiclass classifications, we partition the data into five classes (Normal, Probe, DOS, and U2R and R2L.

Table7: Result of experiment using MLP classifier.

Attack type	TP	FP	Precision	Accuracy
Normal	0.93	0.42	0.62	0.93
DOS	0.74	0.05	0.89	0.74
PROBE	0.64	0.01	0.87	0.64
R2L	0.032	0	0.97	0.032
U2R	0.00	0	0	0.00

4.9 Experiments using Assembly classifier

In this study I used three base classifier C4.5 classifier, NaïveBays Classifier and MLP classifier for building my Assembly classifier . Table (8) shows that 9415of the actual 'normal' test set were detected to be normal; the last column indicates that 97% of the actual 'normal' data points were detected correctly. In the same way, for 'DOS' 5365 of the actual 'attack' test set were correctly detected; the last column indicates that 72% of the actual 'DOS' data points were detected correctly. In the same way, for 'Probe' 1746 of the actual 'attack' test set were correctly detected; the last column indicates that 72% of the actual 'Probe' data points were detected. In the same way, for 'R2L' 708 of the actual 'attack' test set were correctly detected; the last column indicates that 25% of the actual 'R2L' data points were detected. In the same way, for 'U2R' 34 of the actual 'attack' test set were correctly detected; the last column indicates that 51% of the actual 'U2R' data points were detected. The bottom row shows that 67% of the test set said to be normal indeed were normal and 97%, 83%, 91%, 25% of the test set indeed belongs to DOS, Probe, R2L and U2R consecutively.

Table 8: Confusion matrix of Assembly classifier

Confusion Matrix		Predicted class					
		Normal	DOS	PROBE	R2L	U2R	%
Actual Class	normal	9415	47	218	22	9	97%
	DOS	1946	5365	45	19	83	72%
	PROBE	503	146	1746	24	2	72%
	R2L	2088	0	86	708	5	25%
	U2R	28	0	0	5	34	51%
	%	67%	97%	83%	91%	25%	

In table(9) accuracy for five classes that produced by experiment Assembly classifier that show the accuracy 97% for normal ,72% for DOS attacks ,72% for PROBE attacks ,25% for R2L attacks and 51% for U2R.

Table 9: Accuracy for Assembly classifier

Attack type	*TP*	*FP*	*Precision*	*Accuracy%*
Normal	0.97	0.36	0.67	97
DOS	0.72	0.01	0.97	72
PROBE	0.72	0.02	0.83	72
R2L	0.25	0.004	0.91	25
U2R	0.51	0.04	0.26	51

Results in table (9) suggest that the performance can be improved if Assembly classifier model is built that has sub-classifiers trained using different algorithms for each attack category as in figure 2. The best algorithms for each attack category: MLP model for probing, C4.5 for

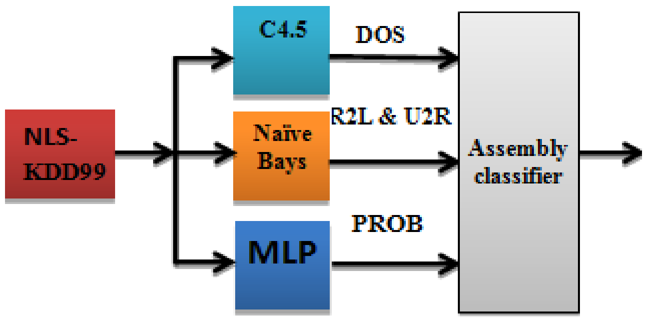

Figure 2: Assembly classifier (C4.5&Naivebays&MLP)

DOS and Naivebyes model for U2R and R2L. This observation can be readily mapped to Assembly classifier topology as in figures 3. Table (9)

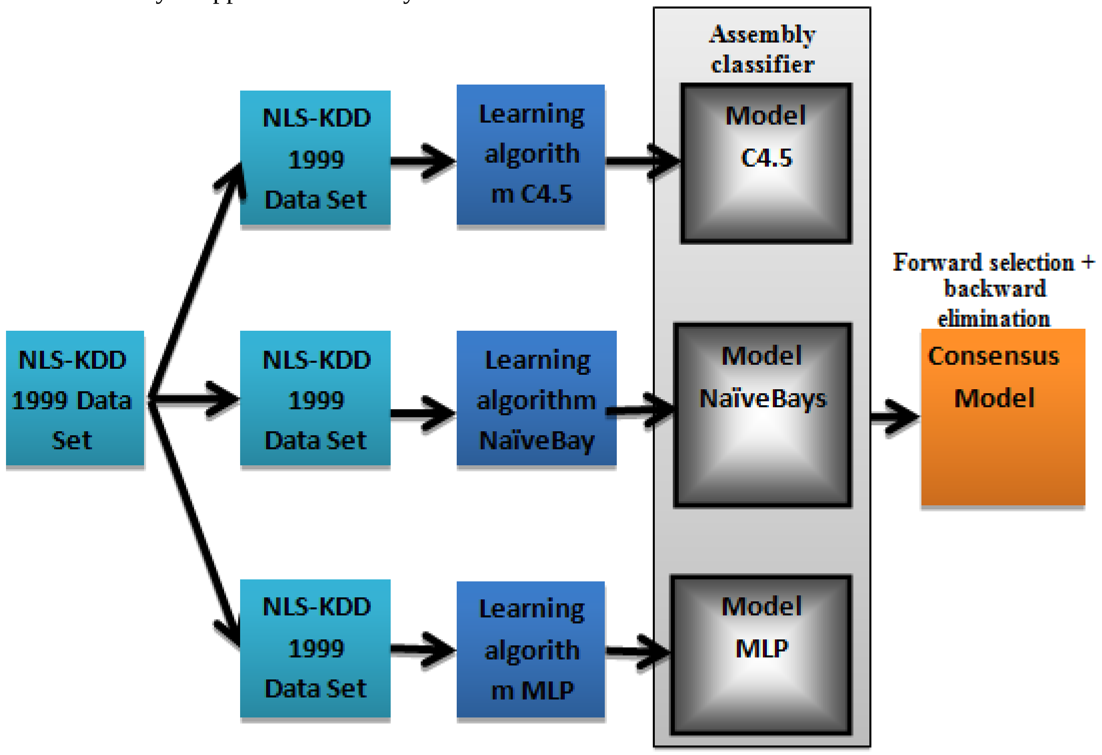

Figure 3: Assembly classifier topology

Suggests that the Assembly classifier model showed significant improvement inaccuracy rates for attack categories. Also the False Positive (FP) was reasonably small for all attack categories.

Table 10: Comparative detection performance of Assembly classifier.

Attack type	*C4.5%*	*Naïve Byes%*	*MNN %*	*Assembly classifier (C4.5, Naïve Byes, M LP)%*
Normal	97	86	93	97
DOS	81	71	74	72
PROBE	59	92	64	72
R2L	001	09	03	25
U2R	0	33	0	51

Table (10) shows the performance comparison of the proposed Assembly classifier model with others in literature. Table(10) summarizes the test results achieved for the five-class classification using C4.5 classifier, NaïveBays classifier, MLP classifier and the Assembly classifier of all three classifiers (C4.5, NaïveBays& MLP).

5. Conclusion

A simulation study was performed to assess the performance of a comprehensive set of machine learning algorithms on the NLS-KDD 1999 Cup intrusion detection dataset. Simulation results demonstrated that for a given attack category certain classifier algorithms performed better. Consequently, Assembly classifier model that was built using most promising classifiers for a given attack category was evaluated for probing, denial-of-service, user-to-root, and remote-to-local attack categories. The proposed Assembly classifier showed improvement in accuracy and false alarm rates for most of attack categories as compared to the NLS-KDD 1999 Cup winner. Furthermore, reduction in cost per test example was also achieved using the Assembly classifier model. However, none of the machine learning classifier algorithms evaluated was able to perform detection of user-to-root and remote-to-local attack categories significantly (no more than 51% detection for U2R and 25% for remote-to-local category). In conclusion, it is reasonable to assert that machine learning algorithms employed as classifiers for the NLS-KDD 1999 Cup data set do not offer much promise for detecting U2R and R2L attacks within the misuse detection context.

References

[1] Anderson. J. P. "*Computer Security Threat Monitoring and Surveillance*." Technical Report, James P Anderson Co., Fort Washington, Pennsylvania, 1980.
[2] Akbar, S., D.K.N. Rao, and Dr.J.A.Chandulal, *Intrusion Detection System Methodologies Based on Data Analysis.* International Journal of Computer Applications (0975 – 8887, 2010. **5– No.2, 2010**
[3] Denning D. An intrusion-detection model. IEEE Trans Software Engng 1987;SE-13(2):222–32.
[4] Kumar S, Spafford EH. *An application of pattern matching in intrusion detection.* Technical Report CSD-TR-94013, Purdue University; 1994a.
[5] Mahoney M, Chan PK. *An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection.* Sixth International Symposium on Recent Advances in Intrusion Detection; 2003. p. 220–37
[6] Tavallaee, M., et al,*"A detailed analysis of the KDD CUP 99 data set"*,Proceedings of the Second IEEE

Symposium on Computational Intelligence for Security and Defence Applications 2009, 2009.

[7] X. Xu, X.N. Wang, "Adaptive network intrusion detection method based on PCA and support vector machines," Lecture Notes in Artificial Intelligence, ADMA 2005, LNAI 3584, 2005, pp. 696-703.

[8] P. Gifty Jeya,M. Ravichandran,C. S. Ravichandran,"*Efficient Classifier for R2L and U2R Attacks*",International Journal of Computer Applications (0975 – 8887, 2012. **45**.

[9] PrabhjeetKaur ,Amit Kumar Sharma, Sudesh Kumar Prajapat*"MADAM ID FOR INTRUSION DETECTION USING DATA MINING*" IJRIM Volume 2, Issue 2 (2012) (ISSN 2231- 4334).

[10] Ron Kohavi and Ross Quinlan. *Decision Tree Discovery*. In Handbook of Data Mining and Knowledge Discovery

[11] I.H. Witthen, E. Frank, *"Data Mining: Practical Machine Learning Tools and techniques with Java Implementations*," Morgan Kaufmann Publishers, 1999.

[12] John, G.H., Langley, P.: *Estimating Continuous Distributions in Bayesian Classifiers*. In: Proc. of the 11th Conf. on Uncertainty in Artificial Intelligence (1995).

[13] Werbos, P.: Beyond Regression: *New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD Thesis, Harvard University (1974)

[14] http://iscx.cs.unb.ca/NSL-KDD/.

[15] Elkan, C., *Results of the KDD'99 classifier learning.* ACM SIGKDD Explorations Newsletter, 2000. **1**(2): p. 63-64.