Cyber-Security: The Use of Big Data Analytic Model for Network Intrusion Detection Classification

Johnson Olanrewaju V.
Department of Computer Science, The Federal Polytechnic, Ile-Oluji, Nigeria

Abstract

Cybersecurity is seen as a major player in the protection of Internet-connected systems, including hardware, software and data, from cyberattacks and other malicious crimes in today's densely connected world-Internet of Things (IoTs). The divers challenge facing Internet users as private and business entities is being advocated as not enough hinderance to seamless interfacing of Mobile Computing and Internet Applications presently making waves. Technology such as Intrusion Detection Systems (IDS) application into cyber-security is an evolving computing mechanism designed as a counter-measure to incessant network threats and intruders. It is one of most reliable pro-defensive tools and has gained significance over time. Meanwhile network traffic data being generated within the context of enormous Internet users requires the application of big data analytical tools for its analysis. This paper, therefore, employs the use of big data analytical tools with its machine learning algorithm on an open-source data set-KDD'99. The full data set was used in the analysis. Predictive model was built in less than 5 minutes time with 99.91% prediction accuracy. Computational challenge and only 10% data set usage, which could only be accounted for in previous research were overcome. Therefore, IDS could be better designed with integration of this classification model result.

Keywords: Cyber-Security, Internet of Things, Intrusion, Mobile, Big data, network
DOI: 10.7176/CEIS/10-7-02
Publication date: November 30th 2019

1.0 INTRODUCTION

The Internet, having gained tremendous acceptance and dominance in every planet of the world's entities such as political, business, finance, education, agriculture to mention but a few, has become a focal point of security concerns. The Internet today is seen as the repository of vast unfathomed information. Its security is at the top of major talks, seminars and conferences. Security has turned out to be a serious issue of concerned as numerous developed internet applications exist today [1]. Cybersecurity is, therefore, playing a major role in the protection of Internet-connected systems, including hardware, software and data, from cyberattacks and other malicious crimes in today's densely connected world-Internet of Things (IoTs).

With much development in securing system over the years, computer security vulnerabilities still surface up always due to active engagement of intruders. An intruder with both passive and active techniques masquerades as legitimate user to steal critical resources of the network system [2]. This has made network managers or administration to be on their toes in order to keep abreast of defense and recovery mechanism, the unwelcomed (intruders or attackers) may be launching from time to time before they beep into the network system and perpetrate havoc [3].

Many are the havoc or threats perpetrated on network system in which intrusion is one. [4] reported that there are still many undetected intrusions despite proven security technologies such as Access Control, Firewall, Anti-malware, encryption and network policies. Thus the need form Intrusion Detection System (IDS).

An intrusion can be broadly defined as a deliberate attempt, evil intension or destructive threat to critical system and network resources in terms of information access, manipulation or inadvertent rendering of system unreliable or unusable. Offenses ranges from Denial of Services (DoS), worms or viruses to host network compromise. It is also viewed as assault, set of actions or potential abuses that breach security of network resources on the promise of integrity, confidentiality or availability [3], [5], [6].

Intrusion detection system therefore, plays a vital role to curtailing the dreaded operations of the attackers. [7] stated that "intrusion detection is the process of dynamically monitoring event occurring in a computer system or network, analyzing them for signs of possible incidents and often interdicting the unauthorized access". IDS which could be software based or a device must provide resistance to intruders [8]; detect anomalies in the network system [9]; prevent access to critical system resources; allows holistic intelligent agent-based monitoring or supervisory role [2]; and much more providing identification and reporting of malicious activities of attackers (machine or human) in a timely manner [10]. The common classification of IDS from research are: Misuse (Signature based) detection and Anomaly detection [5], [6].

In the case of misuse detection, malicious activities are reported by finding out signature patterns in the in-coming traffic. The detection is made to consider any signature detection activities that resemble known attack, while the anomaly exclusive the known previous normalcy of the network. Misuse intrusion system is effective in detecting worms as well as previously unknown attacks. Meanwhile the latter in known with its high false alarm rate.

Many classified statistical and machine learning models have been proposed and implemented on intrusion detection analysis and system design. Results from these works have also been surveyed with the KDD'99 result [3], [8], [11].

[3] did a comparative analyzing of various machine learning tools on KDD' 99 benchmark dataset. Two types of experiment were carried out in their paper. One with the full attributes of 41 and the other with 11. The work claimed that considerable cutback in resources were achieved using 11 reduced features of the dataset.

[12] provided a tabular reviewed of scheme of well-known machine learning. The work reveals some of the pros and cons of ML algorithm and fuzzy logic. They opined that it was difficult to choose a particular method to implement an intrusion detection on the other. A more comprehensive survey on IDS was carried out in [8].

[1] applied JRIP and Reptree algorithms from WEKA (Waikato Environment for Knowledge Analysis) on the extracted KDD'99 dataset. User to Root (U2R) and Remote to local (R2L) were the major attack considered. The argument was based on the fact that mining result mostly show a less consideration for the two when combined with the rest attacks in the full dataset. Meanwhile, these two attacks were considered most dangerous [4]. The algorithms used performed better with performance metrics and rules that can be implemented in a real IDS design.

An improved Naive Bayes algorithm based on Principal Component Analysis (PCA) was proposed by [13]. The PCA was used to obtain new set of attributes serving as input for the Naïve Bayes classifier. Improved weighted Naive Bayes classification were obtained showing a better performance of the approach adopted.

Other area of focus of IDS has also be on a single event stream detection. In this regard, network traffic is directed to monitor a server host or access logs produced by a server application. With this approach, a state full model has not be considered to analyze different events streams thereby providing an integrated state full analysis of multiple event streams [7].

Another major area of concern is that several research works only considered a subset (10%) out of the full KDD'99 dataset. The data set is of the size 743MB containing 4, 898, 431 records, with its testing data set in which many classical running techniques might not be able to handle in terms of time and space complexity of their algorithms. Aside from this, the other challenge that most study on the intrusion defection classifier had, was that training carried out on the small subset of the data did not represent the network pattern training well enough. This has led to identifying attack patterns from normal ones as more false positives are generated.

Big data analytics, a current trend in computing serving as umbrella for machine learning, statistical and visualizing tools for large data set, with its 5Vs acronyms (Volume, Velocity, Variety, Veracity and Value), is employed as a phenomenal solution to the problem of dealing with relative and very large dataset. Coupled with cutting edge advances in clustering and high-performance computing, a vast big data tools are available today such as Hadoop Infrastructure, Spark, H_2O, MapR, and MapReduce. These tools have seamless integration with languages such as Julia, R, Python, Scala, etc.

"Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making." [14].

This paper therefore, takes a direction to addressing the challenges earlier mentioned by relying on the intended promises of big data analytics such as scalability, robustness, massive support of machine learning algorithms, in-memory parallel processing and massive large dataset support. In our model, we propose Random Forests Classifier implemented on H2O platform running on R.

2.0 MACHINE LEARNING

Machine learning (ML), a subfield in computer science had evolved more than a decade with promises of finding solution to problems in Pattern Recognition, Natural Language Processing (NLP), Data Mining and Extraction, Computational Theory in Artificial Intelligence world. The computer is given to learn without explicit programming. Construction of algorithms that can learn from hidden interestingly patterns and makes useful predictions provided. These algorithms accepts input as explanatory features to which predictable or devisable outputs were generated.

ML has long be classified as supervised and unsupervised learning with recent exploration in reinforcement learning. Supervised learning approach learns by training a labeled data to predict a target class, while unsupervised and reinforcement learning build mining results (clustering or grouping similar kind) from unlabeled dataset. Figure 1.0 provides a simple overview of machine learning categories.

Most of existing classical ML algorithms has various limitation of dealing with data sample. Some works better when data sample attributes have uniform data scale. Categorical data attribute can be handled well by some while others can perform better on small data samples.

Our approach in this work was to consider a modern machine learning algorithm- Random Forests, from big data analytical perspective on the KDD'99 full dataset, thus overcoming the challenge of data scalability.

3.0 RESEARCH METHODS

3.1 Dataset

The KDD Cup 1999 dataset was made available by NSL for the third international knowledge and data mining tools competition [15] on intrusion detection. It has since then become the benchmark for various research works on intrusion detection systems. The dataset were provided in different sizes with both the training and test set (see Table 1.0).

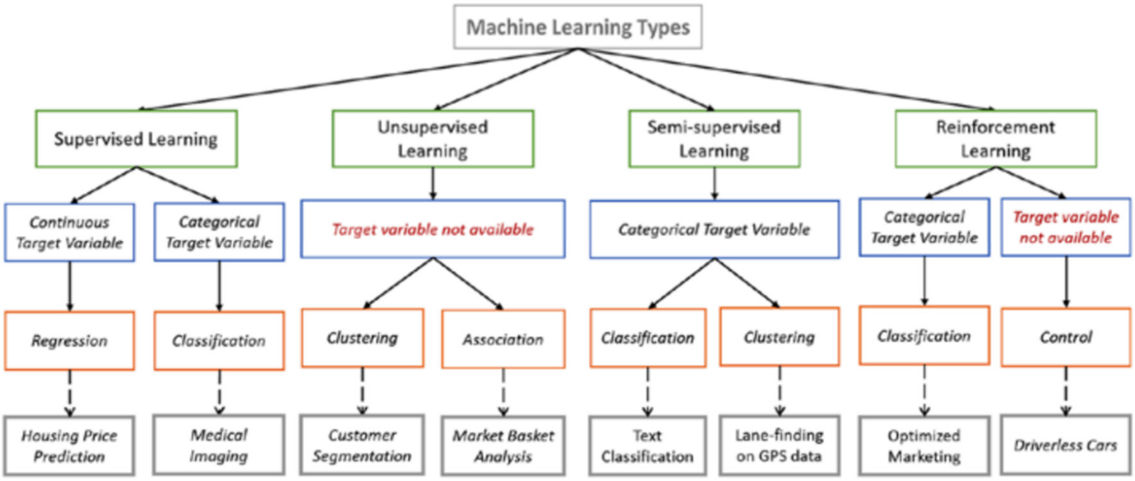

Figure 1.0 Overview of Machine Learning [16]

This is owing to the fact that most algorithms cannot scale up the whole dataset, while some research argument reported that 78% training and 75% testing records are duplicated. This causes bias towards minority attach such U2R and R2L. Meanwhile duplication of attack records could be a pointer to the fact that such attack is frequent more than the other. But it was noted that frequent attacks are less dangerous than the less frequent ones, meaning reasonable attention need be given to the less frequent ones too.

In this paper, we implemented the full data set by classifying attacks into five categories including normal traffic (See Table 2.0), and by solving the problem of imbalanced intrusion using over sampling techniques on the minority attack since the majority attacks has large number of instances as recorded in the dataset.

3.2 Data Preprocessing

The original full data set of KDD'99 consist of 42 features, in which 41 are the exploratory attributes while 42nd attribute is the target class. The 41 attributes consist both continuous, nominal and categorical data values. Since attack types can be classified into unique classes and for the purpose of effective computation, there was the need to reprocess the data into the following categories:

- Normal: These are network traffics seen has non-harmful connections in the network
- Denial of service(DOS): This type of traffic tries to prevent legitimate users access to system and network services e.g. smurf, back, neptune, teardrop, pod and land
- Probing: This attack target the host to exploit information e.g. satan, ipsweep, portsweep and nmap
- User to root (U2R): Super user privileges on local machine of users (victims) is the target of this type of attack e.g. buffer_overflow, rootkit, loadmodule and perl
- Remote-to-local (R2L): Uses various technique to gain access by not having the account of the target host. e.g. guess_passwd, ftp_write, multihop, phf, spy, imap, warezclient and warezmaster.

As earlier mentioned that most research efforts use only a subset (10%) of the full data set, because of computational consideration since the dataset is very large. This is the major focal point of this work. For computational scalability, our proposed model, in one part decided to analyze the whole dataset without any preprocessing and on the other hand carried out different stages of data preprocessing: Missing value imputation, features selection, discretization, and sampling techniques.

3.2.1 Feature Selection

There are 41 features in the KDD'99 dataset numbered from 1 to 41. It is established that not all the attributes of a dataset contribute meaningfully and effectively to the mining process. Some attributes are reductant or less important. The Random Forests (RF) classifier implemented in this work has and allow parameters turning for better performance. It also support feature selection. The RF in-built feature selection was therefore, used to calculate the value of variable importance of the training dataset

3.2.2 Missing value

Dataset are not in most cases totally free of missing values and for the ML algorithm to train well, the missing values must in way be fixed. The training KDD'99 was subjected to missing value test and missing values were fixed using the in-built missing value method of the RF.

3.2.3Discretization

Different discretization techniques were employed on the data set since each record consist of 32 continuous, 3 categorical and 6 nominal attributes. The *min-max* data transformation method was used to normalize numeric values to a range $[0,1]$ as in equation 1.0 coding while re-coding techniques was used on categorical values to integer range $(0,1,2\ldots,n)$.

$$\bar{x} = \frac{x-\min(x)}{\max(x)-\min(x)} \quad \ldots\ldots\ldots\ldots\ldots\ldots1.0$$

Table 2.0 provides the re-coding of target class (attack types) into the five different class using the same re-coding mechanism.

Table 1.0: KDD'CUP 99 data set version and sizes

S/N	Dataset Sample (KDD'99)	Size
1	Full Dataset	743MB
2	10% Dataset	75MB
3	New Test data unlabeled	45MB
4	Full Test data unlabeled	430MB
5	Test data unlabeled	45MB

Table 2.0: Re-coding of attack types

Class of Attack Type	Code
Normal	0
DoS	1
Probing	2
U2R	3
R2L	4

3.2.4 Sampling

Further in the data preprocessing is the sampling of the attacks type. The attack types (Probe, U2R and R2L) with less connections in the full data set are considered. Over-sampling method was carried out on these less attack types in order to overcome any bias the other attack with large connections might be having on them. Random replication was employed on each of the connections. The analysis of our replication is presented in the section 5.

4.0 PROPOSED MODEL: RANDOM FORESTS (RF) AND H2O

In this paper, we implemented a RF classifier on H2O platform. RF classifier is an ensemble machine learning method for classification, regression and other tasks [17]. The classifier works by adding addition layer of randomness to bagging. It further constructs each tree using a different bootstrap technique sample of the data and changes the classification or regression of trees constructed. It has proved to be efficient and scalable in handing dataset with varying attributes types and sizes compare to other classification trees [8]. Algorithm 1.0 below describes the procedure of RF.

```
Algorithm 1.0: Random Forest
Precondition: A training set $T := (x_1, y_1), \dots, (x_n, y_n)$, features $F$,
              and number of trees in Forest $B$
    function RANDOMFOREST($T, F$)
        $H \leftarrow \emptyset$
        for $i \in 1, \dots, B$ do
            $T^{(i)} \leftarrow$ A bootstrap from $T$
            $h_i \leftarrow$ RANDOMISEDTREELEARN($T^{(i)}, F$)
            $H \leftarrow H \cup \{h_i\}$
        end for
        return $H$
    end function
    function RANDOMISEDTREELEARN($T, F$)
        At each node:
            $f \leftarrow$ very samll subset of $F$
            split on best feature in $f$
        return the learned tree
    end function
```

4.1 H2O Platform

Big data analysis, with advances in high performance computing, opens up vast development cutting across software and hardware tools; one of which is H2O. H2O is a fast, scalable, open source machine learning tools for big data and smarter applications. Advanced and classical algorithms were readily provided to solving diverse problems in machine learning [18]. Figure 2.0 describes the architectural framework our proposed model using H2O technologies integration with *data.table* package both implemented on R language [19]. The following cutting-edge features make H2O widely acceptable and applicable to machine learning for big data.

i. Best of breed in open source technology.
ii. Support for web UI and easy interfacing.
iii. All round support of common database, file type and data using integrated development environment (IDE), data compression and all data platforms.
iv. Massively scalable bid data support in real time manner.
v. It provides a real-time Data Scoring with the implementation of Nano fast scoring engine.
vi. Vast Machine Learning algorithm in a parallel and distributed built approach were readily supported. The interfacing was easy for parameters tuning
vii. There is also an ever-growing support for native integration with widely accepted languages such as R, java, Scala and Python. The REST API is robust for easy work flow among the tools.
viii. In memory parallel processing one can stop using sampling data and start using the whole data set available for the analysis. This enables data size of large when readily captured for machine learning processes.

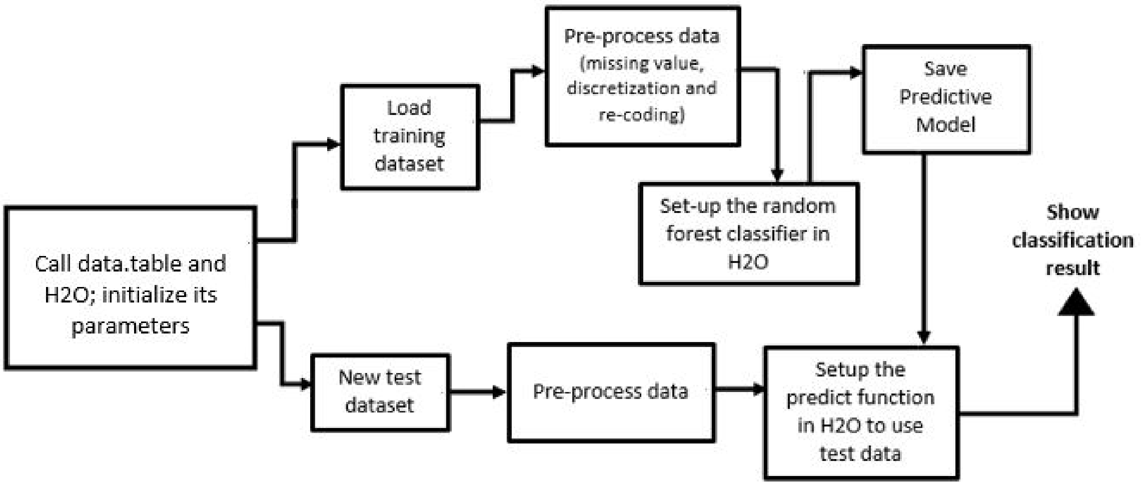
Call data.table and H2O; initialize its parameters
Load training dataset
Pre-process data (missing value, discretization and re-coding)
Set-up the random forest classifier in H2O
Save Predictive Model
Show classification result
New test dataset
Pre-process data
Setup the predict function in H2O to use test data

Figure 2.0 Proposed model of $H_2O$for mining process

5.0 RESULTS

5.1 Datasets

The RF classifier from the H2O platform was implemented on R. The KDD'99 full dataset totally 4, 898, 431 records was analyzed, re-processed and re-sampled prior to the training process. The dataset was re-sampled with total 1, 674, 595 connections making the total training set to 6, 573, 026 records. The test dataset has 431, 330 records. The details analysis of results were provided in the next sections. Figure 3.0 shows the distribution of attack types prior to replication and grouping. Figure 4.0 then shows how the distribution has fared after replication and grouping.

5.2 Timing and Duration

Our coding approach provided a template of measuring time of training the dataset. Table 3.0 shows the details. The training was completed in 14, 604.80s (4 hours 5mins) on HP ProBook 6460b core i5 2520M, 2.50GHz with 16 GB RAM running 64-bit R. This should provide basis for comparison with other architecture, big data tools and ML algorithms.

Table 3.0 Execution time for training the model

User	System	Training Elapsed (s)
165.00	*23.74*	*14604.80*

Figure 5.0 illustrates the duration of each attack type prior to reprocessing. Portsweep was observed to have the highest duration of time in attacking operations followed by warezclient while land, pod are the least observed.

5.2 Model Metrics

The RF model was training with basic parameters settings of 500 number of trees and 100 depth. The outputs generated by the classifier as the details metrics used and produced by the model were shown in Table 4.0, 5.0 and 6.0. Figure 6.0 was generated to show classification error rate in relation to each training trees used. Classification error rate was minimized as the tree increases meaning larger tree yields less error. Thus RF fulfills its mandate by building ensemble (small trees) from the Forests to minimize error.

Table 4.0 Details of model metrics-Trees, Model size and Leaves

No. of Trees	No. of internal trees	Model Size in bytes	Min. depth	Max. depth	Mean depth	Min leaves	Max leaves	Mean leaves
500	*2500*	*12436096*	*13*	*51*	*27.03*	*69*	*1045*	*364.77*

Table 5.0 Training Set Metrics

Mean Squared Error	*0.0007999*
Root Mean Squared Error	*0.0282816*
Logloss	*0.0049005*
Mean per Class Error	*0.0072603*

Table 6.0 Hit Ratio Table: Top 5

K	Hit Ratio
1	*0.9999130*
2	*0.9999995*
3	*1.0000000*
4	*1.0000000*
5	*1.0000000*

5.3 Feature Selection

The Random Forests (RF) classifier implemented in this work has and allow parameters turning for efficient computation. It also support feature selection. The RF in-built feature selection was used to calculate the value of variable importance of the training dataset. Figure 7.0 shows that V3: Service is the most importance attribute, followed by V23: count while V21: is_host_login and V6:dst_byte are least important (see Table 7.0 for details attribute description and coding).

5.4 Confusion Matrix

The confusion matrix is used to determine and summarize the performance classifier on test data such as accuracy, error rate, sensitivity (recall) and precision. A typical confusion matrix is provided in Table 8.0 below. The performance metrics that can be deduced from the matrix are as follows.

i. Accuracy: The proportion of classification of the whole dataset that were correct by 100%.

$$Acc = \frac{tp+tn}{tp+fp+tn+fn} \times 100 \text{……….2.0}$$

ii. Precision: The measure of correct positive classifications out of the total positive classifications.

$$p = \frac{tp}{tp+fp} \text{……….3.0}$$

iii. Recall: The measure of positives classified correctly

$$r = \frac{tp}{tp+tn} \text{……….4.0}$$

iv. F-measure: This is the balance between precision and recall. It is the actual measure of harmonic mean of precision and recall.

$$fm = \frac{2 \times p \times r}{p+r} \text{……5.0}$$

v. TP and TN rates: The measure of positive proportions identified as positives –TP, TN is the measure of negative proportions identified as negatives.

vi.

$$TP = \frac{tp}{tp+fn} \text{……..6.0}$$

$$TN = \frac{tn}{fp+tn} \text{…..…7.0}$$

The confusion matrix generated from the trained model is shown in Table 9.0, the overall performances metrics of equations (2.0) to (7.0) were provided in Table 10.0. 99.95% accuracy was achieved for Normal, 99.99% for DoS, 99.98% for Probing, 96.44% for U2R while 100% for R2L.

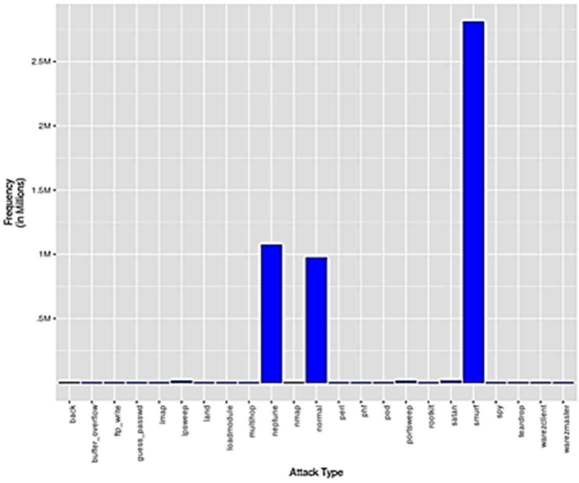

Figure 3.0 Frequency Distribution of Attack Types prior to replication

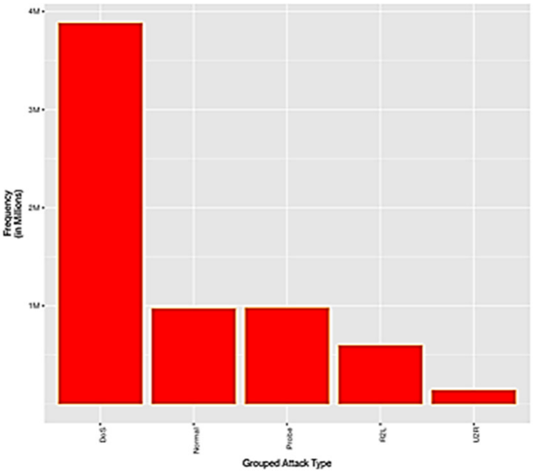

Figure 4.0 Frequency Distribution of Grouped Attack Types after replication of

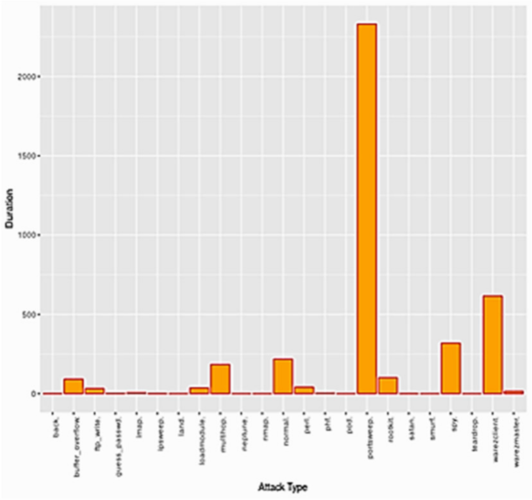

Figure 5.0 Duration of each attack types within the network system

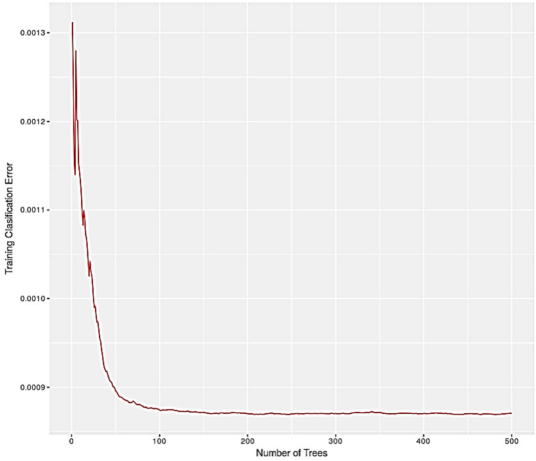

Figure 6.0 Training Classification Error as the number of tree grows

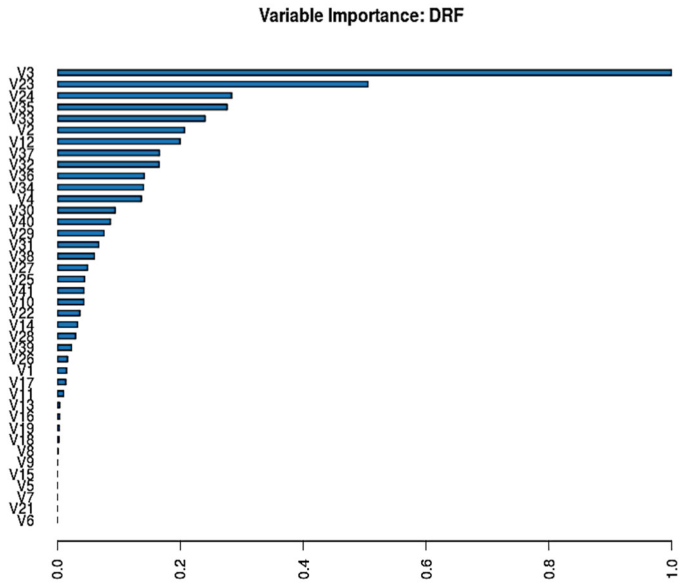

Figure 7.0 Variable importance plot of the trained model

Table 7.0: Attributes description and coding

Attributes	Code	Attributes	Code	Attributes	Code
Duration	V1	su_attempted	V15	same_srv_rate	V29
protocol_type	V2	num_root	V16	diff_srv_rate	V30
Service	V3	num_file_creations	V17	srv_diff_host_rate	V31
Flag	V4	num_shells	V18	dst_host_count	V32
src_bytes	V5	num_access_files	V19	dst_host_srv_count	V33
dst_bytes	V6	num_outbound_cmds	V20	dst_host_same_srv_rate	V34
Land	V7	is_host_login	V21	dst_host_diff_srv_rate	V35
wrong_fragment	V8	is_guest_login	V22	dst_host_same_src_port_rate	V36
urgent	V9	Count	V23	dst_host_srv_diff_host_rate	V37
Hot	V10	srv_count	V24	dst_host_serror_rate	V38
num_failed_logins	V11	serror_rate	V25	dst_host_srv_serror_rate	V39
logged_in	V12	srv_serror_rate	V26	dst_host_rerror_rate	V40
num_compromised	V13	rerror_rate	V27	dst_hosst_srv_rerror_rate	V41
root_shell	V14	srv_rerror_rate	V28	attack class	V42

Table 8.0 Confusion Matrix Definition

Class	**Yes**	**No**
Yes	*True Positive (TP)*	*False Negative (FN)*
No	*False Positive (FP)*	*True Negative (TN)*

Table 9.0: The Model Confusion Matrix

				Actual			
	Class	***0***	***1***	***2***	***3***	***4***	***%***
Predictions	***0***	*972,273*	*16*	*428*	*18*	*46*	*99.95*
	1	*31*	*3,883,338*	*1*	*0*	*0*	*99.99*
	2	*178*	*0*	*979,291*	*0*	*0*	*99.98*
	3	*0*	*0*	*0*	*135,600*	*5,004*	*96.44*
	4	*0*	*0*	*0*	*0*	*596,802*	*100.00*
	%	*99.98*	*99.99*	*99.96*	*99.99*	*99.16*	

Table 10.0: Classifier's Overall Performance Details.

Classifier	Accuracy	Precision	Recall	F-Measure	TP Rate	TN Rate	FP Rate
H2O Random Forests	*99.91%*	*0.999*	*0.999*	*0.999*	*0.999*	*0.999*	*0.001*

6.0 CONCLUSION AND FUTURE WORK

In this paper, we have been able to demonstrate that big data analytical tools are capable of scaling any data size irrespective of the number of rows and columns. It also provides leverages points for implementing classical and modern machine learning algorithms to cope with large data thereby overcoming any computational and complexity challenges. 99.91% accuracy was obtained from the RF implemented. Classification error were minimized with larger trees. Feature selection was also efficiently computed. This is considerably a better result and it shows that RF is effective in providing predictive mechanism for up-coming IDS. Meanwhile the model was implemented on localhost cluster machine with minimal resources for big data analytics. Future work could be carried out by investigating the model on distributed-mode cluster machine with better resources in order to minimize training time. Other big data tools with integrated machines learning techniques could also be implemented for IDS analysis. This would further help to confirm the viability of the result obtained in this work.

Big data analytics has come to stay as more and more data are being hyper-generated in and around us. Big data analytics promises to yield enormous benefits to life, information systems, businesses and National security at large, as it is being embraced.

7.0 REFERENCES

[1] Aladesote, I.O., Johnson, O.V., and Agbelusi, O. (2016) Comparative Analysis of Machine Learning Algorithms Toward Intrusion Detection System. Journal of Science, Food and Hospitality, Rufus Giwa Polytechnic, Owo, Ondo State, Nigeria. Vol.3/4 No 1. pp. 111-117.

[2] Satyendra, R., and Nitesh, T. (2015) Machine Learning Techniques in Intrusion Detection: A Comprehensive Review. International Journal of Advanced Engineering and Global Technology. Vol 03, Issue 11, ISSN: 2309-4893.

[3] Yasir, H., Sugurmaran, M., and Journaux, L. (2016). Machine Learning Techniques for Intrusion Detection: A comparative Analysis. Proceeding of the International Conference on Informatics and Analytics (ICIA-16), Articles No 53.

[4] Jiong, Z., and Mohammad, Z. (2005) Network Intrusion Detection Using Random Forestss. Proceeding of 3rd Annual Conference on Privacy, Security Trust (PST), St. Andrews, NB, Canada, Oct. 2005, pp. 53–61.

[5] Testfahun, A., and Bhaskari, D.L., (2013) Intrusion Detection Using Random Forests Classifier with SMOTE and Feature Reduction. 2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies. 978-0-4799-2235-2/13 $31.00, IEEE, doi:10.1109/CUBE.2013.31.

[6] Saurabh, M., and Neelam, S. (2012) Intrusion Detection Using Naive Bayes Classifier with Feature Reduction. 2nd International Conference on Computer, Communication, Control and Information Technology (C3IT). Procedia Technology 4 (2012), pp. 119-128, doi:10.1016/j.protcy.2012.05.017.

[7] Zamani, M., and Movahedi, M. (2013). Machine Learning Techniques for Intrusion Detection. *arXiv preprint arXiv:1312.2177*.

[8] Asghar, A.S., Malik, S. H., and Muhmmad, D. A. (2015) Analysis of Machine Learning Techniques for Intrusion Detection System: A Review. International Journal of Computer Applications (0975-8887), Vol. 119, No 3.

[9] Amor, N., Benferhat, S., and Elouedi, Z. (2003) Naive Bayesian Networks in Intrusion Detection Systems. Workshop on Probabilistic Graphical Models for Classification, 14th European Conference on Machine Learning/7th European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD'2003).

[10] Truong, S.P., Tuan, H.H., and Van, C.V. (2016). Machine Learning Techniques for Web Intrusion Detection- A Comparison. 8th International Conference on Knowledge and Systems Engineering (KSE), 978-1-4673-8929-7/16/$31.00, IEEE.

[11] Fanaaz, N., and Jabbar, M.A. (2016) Random Forests Modeling for Network Intrusion Detection System. 12th International Multi-Conference on Information Processing (IMCIP). Procedia Computer Science 89 (2016), pp. 213-217, doi: 10.1016/j.procs.2016.06.047

[12] Jayveer, S., and Manisha, J.N. (2013) A survey on Machine Learning Techniques for Intrusion Detection Systems. International Journal of Advanced Research in Computer and Communication Engineering. Vol. 2, Issue 11, ISSN (Print): 2319-5940, On-line: 2278-1021.

[13] Xiaoyan, H., Liancheng, X., Min, R., and Weiping, G. (2015). A Naive Bayesian Network Intrusion Detection Algorithm Based on Principal Component Analysis. 7th International Conference on Information

Technology in Medicine and Education, 978-1-4673-8302-8/15 $31.00, IEEE. doi: 10.1109/ITME.2015.29.
[14] Gartner IT Glossary (n.d.) Retrieved from http://www.gartner.com/it-glossary/big-data/.
[15] KDD (1999) KDD Cup 1999 Dataset. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[16] Overview of Machine Learning. http://en.proft.me/2015/12/24/types-machine-learning-algorithms.
[17] Breiman, L., (2001) Random Forestss. Statistics Department University of California, Berkeley.
[18] H_2O Datasheet (n.d) Introducing H2O: Fast, Scalable Machine Learning for Better Predictions. http://h2o-release.s3.amazonaws.com/h2o/rel-lambert/5/docs-website/resources/h2odatasheet.html. Retrieved 28th Aug. 2017.
[19] R Core Team (2017) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/