International Journal of Scientific & Engineering Research, Volume 11, Issue 3, March-2020
ISSN 2229-5518

Analyzing Network Traffic Data Using a Neural Network in the Statistical Computing Language R

Zachariah Pelletier, Munther Abualkibash

Abstract— The analysis of network traffic is crucial to the design and implementation network detection systems. The R language is a statistical computing environment capable of performing a variety of data analysis techniques. In this paper, we will evaluate the R language for its ability to pre-process network data from the NSL-KDD dataset and use third-party packages to analyze this data. We will build a predictive model to classify testing data using a neural network from one of the widely available R packages.

Index Terms— R Language, Machine Learning, Neural Network, Network Traffic, NSL-KDD, nnet, Boruta

1 Introduction

R is a programming language and development environment used primarily for statistical analysis and graphics. As a widely available tool for statistics, many data scientists and statisticians use R for data processing, data analysis, and displaying visual data metrics. Although popular among data scientists, the R language has also been highly utilized by data miners, healthcare workers, government workers, and professionals in many other fields [1]. One field that has a very low utilization of this language is Information Security. Currently, Python is more favorable among security professionals due to its versatility and ability to be used as a scripting language, data processing language, and programming language.

R packages can be created to expand the initial set of capabilities contained by the language and are able to be processed by other statistics tools. In this paper the NSL-KDD network dataset [2] will be used to evaluate several R packages and libraries to determine the efficiency of the use of R for detecting anomalies in network traffic. The ease of use and readability of the R language will be evaluated as well as the availability of popular libraries for completing common tasks. Our primary benchmark for determining the fit of the R language for completing security related work will be the ability of the language to produce meaningful visuals and perform predictive tasks in machine learning such as detecting malicious network traffic in the context of intrusion detection.

2 Related Work

2.1 Using R For Anomaly Detection in Network Traffic

Dennis Hock and Martin Kappes evaluate the R language for its ability to perform anomaly detection in network traffic [3]. Hock and Kappes use R for data preprocessing, analysis operations, and detection of anomalies in order to create an anomaly-based Intrusion Detection System entirely in R and support their process with experiment driven results. An anomaly-based Intrusion Detection System identifies attacks by finding deviations from normal behavior. In order to evaluate R for use in Intrusion Detection Systems in general, the authors used a wide range of different properties and calculated their metrics using the R programming language.

The authors propose an entirely R-based system of anomaly detection on the premise that the resulting system would allow for easier development of new detection methods [3]. In order to thoroughly test the practicality of the R language in such an application, a wide range of metrics are used to detect anomalies which can generally be divided into two categories: volume-based and feature-based.

Volume-based metrics detect anomalies on an overall network level by counting properties of flows. Feature-based metrics use the information in packet headers to detect anomalies and can be used to detect special attacks like a port scan that are not detectable via a volume-based method. The authors used correlation analysis as their primary test on their metrics, however, they also performed outlier detection and checked these metrics against thresholds on the data in order to explore a wide range of techniques in R. Aside from the use of tshark to extract data from the traffic dumps, no outside software or libraries were used to for the correlation analysis. The authors then derived the metrics from their sample data and performed a correlation analysis between abnormal data and their example traffic. The authors concluded that the R language is fully capable of both data pre-processing and analyzing network traffic without the use of any third part R packages.

2.2 A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms

The NSL-KDD Dataset is a highly refined and improved version of the KDD cup99 data set [4]. The NSL-KDD data set is made up of labeled network connection vectors from the same data that comprises the KDD cup99 data set. In this work, Dhanabal and Shantharajah outline the various

- *Zachariah Pelletier is currently pursuing bachelors degree in Information Assurance and Cyber Defense at Eastern Michigan University, United States. E-mail: zpelleti@emich.edu*
- *Munther Abualkibash is an Assistant Professor at Eastern Michigan University, United States. E-mail: mabauilki@emich.edu*

International Journal of Scientific & Engineering Research, Volume 11, Issue 3, March-2020
ISSN 2229-5518

features of each of the network connection vectors in the data set and relate each attack type to their various category of packet. The authors also briefly reflect on the improvements that were made on the original data set and what issues were addressed by those improvements.

The authors then analyze classification techniques for detecting abnormal traffic and different methods of pre-processing the vector data for classification. A main focus of their efforts to pre-process the data rely on feature selection of the vectors. Many classification techniques are more accurate when applied to normalized data; the authors focused on using the J48, SVM, and Naïve Bayes classification methods on the NSL-KDD dataset. The J48, SVM, and Naïve Bays methods resulted in around a 98%, 95%, and 74% classification accuracy respectively [4].

The authors conclude that the NSL-KDD dataset is the best available dataset to simulate realistic network traffic and test the performance of different IDS methods. The authors also identified that an initial analysis and observation phase improves the ability for the researcher to visualize correlation in different vector features before pre-processing and classifying the data.

2.3 A Pre-Clustering Method to Improve Anomaly Detection

While anomaly detection is a widely accepted method of detecting network misuse, the authors of this paper posit that normal events outside of optimal activity such as unpredictable user actions or application changes could decrease the predictive power of an IDS model. They evaluate the use of pre-clustering or 'pre-sorting' of data in order to mitigate the effects of false positives. The authors test pre-clustering methods based on different features of the sample network traffic they use such as clustering based up IP Address, different network traffic characteristics, and application layer protocols [5].

Since anomaly detection works to detect differing statistical deviations from the normal flow of network traffic, one of the many advantages this method has is to detect previously unseen events over a network. This way, a model does not have to be trained for every possible attack or updated when the attack space changes, the detection system can evolve to match the exterior threats. Coupled with an outlier detection algorithm, pre-clustering can very effectively increase the odds of proper classification when training a predictive model. The authors focus on three primary metrics used to outline the properties of network traffic: IP Addresses, flow characteristics, and Principle Component Analysis/k-means clustering [5].

In order to effectively cluster IP addresses, the authors attempted to separate traffic into subnets associated with the local area network of the sender. Clustering by flow characteristics was mainly achieved by separating data by flow duration, byte size, and number of packets per flow [5]. Since flow characteristics have historically been used to predict application layer information, the authors hoped to use this method to classify all data according to application use. Finally, all of the data was split by application layer protocols using a machine learning algorithm to perform a k-means clustering of the data. Each resulting cluster has the ability to potentially simplify the detection of certain classes of attacks.

Clustering by each of these three methods increased the ability of an anomaly detection system to detect network anomalies by 36%. The authors used ROC curves to calculate the ultimate tradeoff between true positives and false positives. The area under the curve of the detection without clustering was 0.47 while the area under the curve of detection with clustering was 0.83. The authors concluded that clustering by these methods significantly increased the performance of methods of anomaly detection.

2.4 Performance Analysis of NSL-KDD Dataset Using ANN

In this paper, the NSL-KDD dataset is analyzed using ANN or Artificial Neural Network. Using an ANN, the authors construct a model to provide intrusion detection based upon network traffic vectors. They use the neural network to detect both binary data (attack or normal traffic) and classification by attack type (there are five attack types in the NSL-KDD dataset). This paper is focused on intrusion detection surrounding smart and IoT devices, but the methods used to develop the predictive model can be generalized to any network. The authors used a variety of performance metrics and used these metrics to attain better results with increased accuracy.

After gathering the results from the neural network when applied to the NSL-KDD dataset, they found an 81.2% detection rate for binary attack vectors and a 79.9% detection rate for classification of attacks [6]. In order to test the performance of their predictive model against a benchmark, the authors compared their results with that of pre-existing intrusion-detection schemes and found that their system had a higher detection rate and performed well in all categories, both for binary attack data and attack type classification.

International Journal of Scientific & Engineering Research, Volume 11, Issue 3, March-2020
ISSN 2229-5518

3 Experimental Methods

3.1 Initial Observation of Raw Vectors

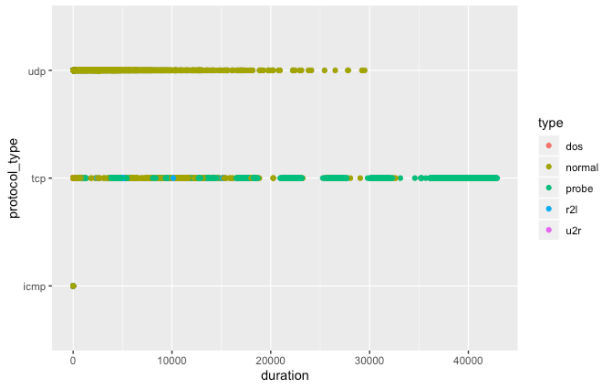

Fig. 1. Duration of the network connection vector plotted against the protocol type of the connection vector. Attack type is color coded.

Before any processing or normalization was performed on the data vectors, we performed an initial observational analysis on the data to determine any pre-existing correlations that may exist between different features. A variety of different feature combinations were checked, only a select few of which were useful. We defined a useful visualization as one that showed a clear correlation of data in a strong enough pattern as to warrant investigation. Visualization was conducted in the form of scatter plots using the qplot function from the ggplot2 R library [7].

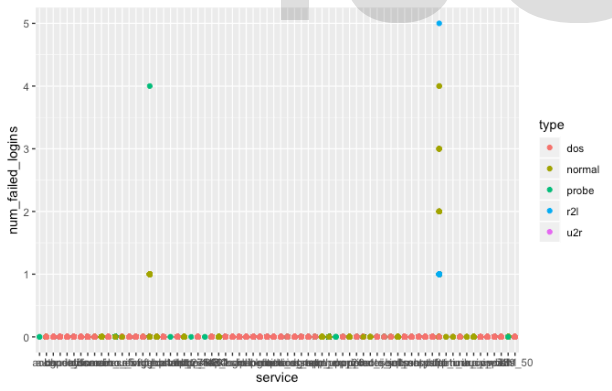

Fig. 3. The number of failed logins to a user given a connection vector plotted against the service used by that connection. Attack type is color coded.

As can be seen in Figure 1, there is a relatively strong correlation between TCP flag and the service that was used in the attack. From here we can see that the S0 flag is a strong predictor for a DoS attack and that RSTR and SH flags are strong predictors for Probe attacks. There is no strong correlation between any other flag and attack type. In figure 2, protocol type was plotted against the duration of the connection. Here we discovered that virtually no malicious packets were sent using UDP or ICMP protocols allowing us to filter out around 18.5% of the traffic when analyzing malicious packets in isolation.

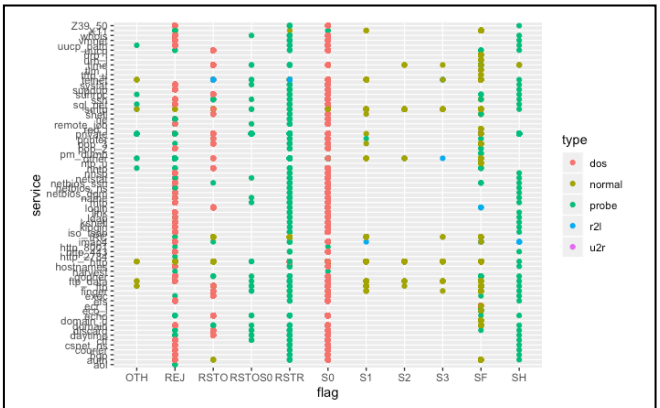

Fig. 2. Service used by the connection vector plotted against TCP flag. Attack type is color coded.

Additionally, it should not be ignored that a majority of the malicious traffic is of the tcp packet type. When looking at the number of failed logins compared to the service that was used for login, it is worth noting that any number of failed logins greater than or equal to five using an FTP service is a
strong indicator of a r2l attack. This relationship can be seen in figure 3. When checking the count of source bytes against the duration of the connection, we can see that for a source byte count over 5e+08, a probe attack type is highly likely. Additionally, any source byte count over 5e+04 also seems to strongly indicate a Probe type attack.

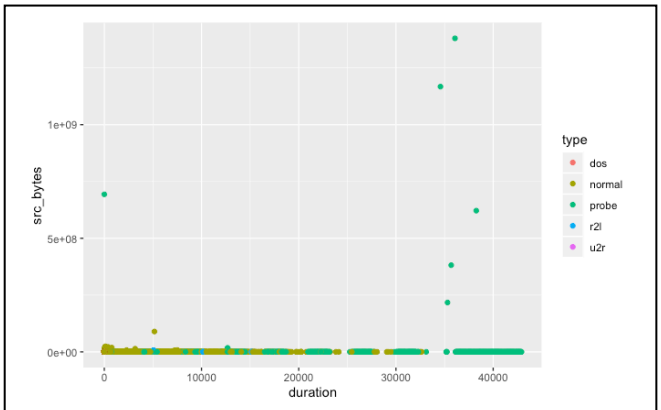

Fig. 4. The number of source bytes transmitted by the connection vector plotted against the duration of the connection. Attack type is color coded.

When duration was plotted against source, but with a third axis of the number of source bytes associated with the packet, we can see a clear correlation between the size of the packet (number of source bytes) and the duration of the connection. A larger duration seems to be predictive of a

International Journal of Scientific & Engineering Research, Volume 11, Issue 3, March-2020
ISSN 2229-5518

packet having an abnormally high number of source bytes. This relationship can be seen in figure 5.

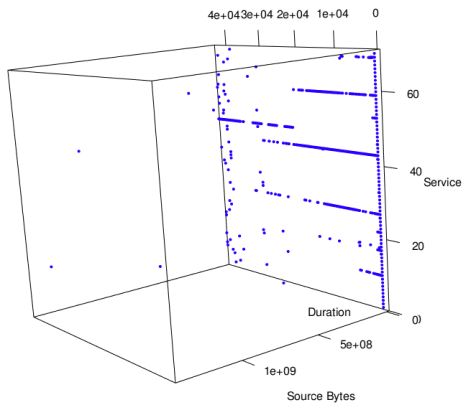

Fig. 5. Duration of the connection plotted against the service used with the number of source bytes on a third axis.

3.2 Automated Pre-Processing with Boruta

Many of the same processing, machine learning, and deep
learning libraries that are available for Python are also available in R. For this project, we have decided to use Boruta to determine the importance of each column in predicting an attack output. Boruta is an open source library that works by searching for relevant features in a dataset to determine importance by comparing selected features correlation against a random selection of predictors [8]. By using Boruta to test features in our data set for importance, we can effectively pre-process our data to eliminate superfluous or non-predictive columns in our data set. In creating the Boruta object, we checked the feature relevant importance of the service against the type of attack that the vector was labeled for with a maximum of 20 iterations through the generated search tree. The relevant library function we used to retrieve the results was getSelectedAttributes().

Boruta classifies a feature as one of three possible levels of predictive qualities; a dataset feature is either important, unimportant, or tentative [8]. The tentative category is reserved for features that were found to be within a certain margin of the importance cutoff, but needed to be run through the more iterations of Boruta's algorithms in order to be strictly classified as either important or unimportant. It is possible for all features to be classified, but what determines how fast (and under how many iterations) a dataset can be fully classified depends on the hardware on which the program is run. For instance, for the NSL-KDD dataset, the Boruta function was run on two different machines with differing hardware specifications. The faster computer was able to classify all features in just 13 iterations in a period of 11 minutes while the slower computer could only classify 38 out of 41 features over 20 iterations in a period of 60 minutes. Both trial results as well as hardware specifications can be found in table A.

TABLE I. HARDWARE SPECIFICATIONS

Iterations	Hardware Specifications		
	Processor	*Memory*	*GPU*
11 (11 min)	3.20 GHz Intel Xeon CPU E5-2667 v4	128 GB	NVIDIA NVS 310
19 (60 min)	2.3 GHz Dual-Core Intel Core i5	8 GB 2133 MHz LPDDR3	Intel Iris Plus Graphics 640 1536 MB

After running 19 iterations over the course of 60 minutes, 3 features were determined to be unimportant (is_hot_login, num_outbound_cmds, urgent) and 3 features were tentative (num_access_files, num_shells, su_attempted). Running a second trial on a faster computer confirmed all tentative features to be important, eliminating 3 columns from our data set due to lack of importance. In order to evaluate the performance of all selected features, we used the getSelectedAttributes() function to retrieve scoring information from Boruta. The highest ranking feature was determined to be protocol_type with a median importance level of 34.330594. The top 5 important network features and their stats can be found in table B.

TABLE II. IMPORTANCE BENCHMARKS FOR NETWORK FEATURES

Feature	Importance Benchmarks for Network Features		
	Mean Importance	*Min Importance*	*Max Importance*
Protocol_type	34.330594	32.520757	36.0546544
Wrong_fragment	32.695304	30.513209	33.8031660
Src_bytes	29.944937	28.694210	31.8848235
Srv_count	29.574201	26.431805	31.3077636
Dst_host_same_src_port_rate	28.167884	27.003862	29.3803172

According to Dhanabal and Shantharajah as defined in [4], protocol_type is simply the IP protocol that was used for a specific network connection. Wrong_fragment is the number of IP fragment packets that do not belong to the connection. Src_bytes is considered the number of bytes from the source of the packet. Srv_count is defined as the "Number of connections to the same service (port number) as the current connection in the past two seconds." [4]. Finally, dst_host_same_src_port_rate is defined as "The percentage of connections that were to the same source port, among the connections aggregated by feature 33 (the number of connections having the same port number." [4]. It can be concluded from the results after pre-processing that the aforementioned features are the five most important features to consider when building a predictive model for network attacks.

International Journal of Scientific & Engineering Research, Volume 11, Issue 3, March-2020
ISSN 2229-5518

3.3 Building a Predictive Model with nnet

For each implementation of our predictive model (i.e. one that detects network anomalies), we will try to build our model with and without the features that we determined to be unimportant in order to determine what effect on efficiency of the building and training process it has (if any). Each trial was also conducted with and without pre-clustering to allow for a variety of pre-processing techniques to be utilized. This predictive model was generated using a neural network generated by the R Library nnet [8]. Nnet is a "Software for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models" [9]. In order to factor our data frame such that the data can be processed correctly, all of the imported csv data had to be sorted into categories labeled by attack. Once this pre-processing step was complete, the training of the neural network could begin. We wanted to get a thorough, but accurate sample from our data with which to train our neural network.

In order to accomplish this, we selected a randomized sample of size 5000 from a slice of the first 10000 rows or our data, then a second slice from the second 10000 rows of our data and so on. We then called the nnet subroutine [8] and passed in the following arguments: we trained based upon the attack type using our training data provided in the NSL-KDD data set, used a subset of our randomized sample slices, used a neural network consisting of 5 neurons, a decay of 1.0e-5 and a maximum of 500 training iterations. After training, we imported the test data (also provided in the NSL-KDD dataset) and used our predictive model to predict the outcome of all test network connection vectors. In order to accomplish this, we used nnet's predict subroutine passing in our trained neural network and the test data. Finally, we displayed the resulting data in a table format and calculated the percentage of connection vectors that were correctly predicted as the correct attack type (normal data was also included and predicted with an attack type of N/A). The results from this process are shown in table III below.

TABLE III. RESULTS FROM ATTACK DETECTION VIA NEURAL NETWORK

Attack	Attack Detection Metrics 500 iteration Neural Network (Without Feature Elimination)		
	Number of Connections Correctly Predicted	*Total Connections*	*% Correct*
DoS	5602	5741	97.6
Normal (N/A)	9059	9711	93.3
Probe	1103	1106	99.7
Remote to Local	2135	2199	97.1
User to Root	35	37	94.6

The neural network was first trained using a maximum iteration range of 50 iterations, then 500 iterations to compare the effect of more training sessions on the ability for the model to successfully predict network traffic. Our model trained on 500 iterations had an average success rate of 96.28% over a period of 64.218 seconds while our 50-iteration model's average success rate was only 80.72% over a period of 5.964 seconds. Immediately following the training of a neural network using all network features provided to us in the NSL-KDD data set, we tried training a predictive model using only important features that were established using the Boruta package [8].

According to our the results from our importance test using Boruta, the three least important network features (is_hot_login, num_outbound_cmds, urgent) should be able to be safely eliminated from our training (and testing data) with a minimal resulting effect on the efficiency or effectiveness of our predictive model. We were able to eliminate all three features from both of our data sets and train our neural network using the modified data. A summary of the results from testing our neural network on our testing data with feature elimination is shown in table IV below.

TABLE IV. RESULTS FROM ATTACK DETECTION VIA NEURAL NETWORK

Attack	Attack Detection Metrics 500 iteration Neural Network (With Feature Elimination)		
	Number of Connections Correctly Predicted	*Total Connections*	*% Correct*
DoS	5360	5741	93.4
Normal (N/A)	9093	9711	93.6
Probe	1102	1106	99.6
Remote to Local	2163	2199	98.7
User to Root	37	37	100

As our results have shown, using feature elimination before training our neural network has increased the efficiency of our predictive model by 9 seconds (training took 55.2 seconds) and increased effectiveness by 0.78%. While the increase in predictive ability is marginal at best, the resulting speed reduction was significant with a 17% increase in efficiency. Pre-processing our data by eliminating unimportant data features has thus been determined to be useful when applied to training a predictive model using a neural network.

4 CONCLUSION

The R programming language has been used to complete a variety of standard network security analysis tasks. The ability of the language and development environment to create and display statistical visuals with ease is a strength that adds value to its implementation. In addition to the many built-in options for the language, we have been able to utilize several third-party libraries with ease. Using the widely available Boruta package [7], we have successfully determined the contextual importance of each of our

International Journal of Scientific & Engineering Research, Volume 11, Issue 3, March-2020
ISSN 2229-5518

data features and were able to determine which features could be eliminated from our training set without affecting our average classification accuracy. We were able to implement a neural network using the nnet package [8]. We trained a predictive model using that network with and without the data features that were established to be unimportant while increasing our training algorithm's efficiency by 17% overall.

REFERENCES

[1] "*Visits to R by Industry*," Accessed on Jan. 8, 2020. [Online]. Available: https://www.guru99.com/images/r_programming/032918_1002_WhatisRProg1.png

[2] "*NSL-KDD Dataset*," Accessed on Jan. 8, 2020. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[3] D. Hock, M Kappes, "Using R for anomaly detection in network traffic," ResearchGate, 2013. Frankfurt University of Applied Sciences.

[4] L. Dhanabal, Dr. S.P. Shantharajah., "A Study on NSL-KDD Dataset For Intrusion Detection System Based on Classification Algorithms," International Journal of Advanced Research in Computer and Communication Engineering, 2015. Kumaraguru College of Technology, Coimbatore, India and Sona College of Technology, Salem, India.

[5] D. Hock, B.V. Ghita, M. Kappes, "A Pre-Clustering Method to Improve Anomaly Detection," ReserachGate, 2016. Frankfurt University of Applied Sciences and University of Plymouth.

[6] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," *2015 International Conference on Signal Processing and Communication Engineering Systems*, Guntur, 2015, pp. 92-96.

[7] I. Keka, B. Çiço, "Data Visualization as Helping Technique for Data Analysis, Trend Detection and Correlation of Variables Using R Programming Language," Mediterrnean Conference on Embedded Computing, 2019. AAB College, Prishtina, Kosovo and Epoka University, Tirana, Albania.

[8] "*Package 'Boruta'*," Jul. 17, 2018. Accessed on Feb. 5, 2020. [Online]. Available: https://cran.r-project.org/web/packages/Boruta/Boruta.pdf

[9] "*Package 'nnet'*," Feb. 25, 2020. Accessed on Feb. 20, 2020. [Online]. Available: https://cran.r-project.org/web/packages/nnet/nnet.pd