Anomaly-Based Intrusion Detection*:* Feature Selection *and* Normalization Influence *to the* Machine Learning Models Accuracy

Danijela Protić, MSc

Center for Applied Mathematics and Electronics, Belgrade, Serbia

Miomir Stanković, PhD

Matematical Institute, Serbian Accademy of Science and Arts, Belgrade, Serbia

Abstract

Anomaly-based intrusion detection system detects intrusion to the computer network based on a reference model that has to be able to identify its normal behavior and flag what is not normal. In this process network traffic is classified into two groups by adding different labels to normal and malicious behavior. Main disadvantage of anomaly-based intrusion detection system is necessity to learn the difference between normal and not normal. Another disadvantage is the complexity of datasets which simulate realistic network traffic. Feature selection and normalization can be used to reduce data complexity and decrease processing runtime by selecting a better feature space This paper presents the results of testing the influence of feature selection and instances normalization to the classification performances of k-nearest neighbor, weighted k-nearest neighbor, support vector machines and decision tree models on 10 days records of the Kyoto 2006+ dataset. The data was pre-processed to remove all categorical features from the dataset. The resulting subset contained 17 features. Features containing instances which could not be normalized into the range [-1, 1] have also been removed. The resulting subset consisted of nine features. The feature 'Label' categorized network traffic to two classes: normal (1) and malicious (0). The performance metric to evaluate models was accuracy. Proposed method resulted in very high accuracy values with Decision Tree giving highest values for not-normalized and with k-nearest neighbor giving highest values for normalized data.

Keywords: feature selection, normalization, k-NN, weighted k-NN, SVM, decision tree, Kyoto 2006+

Introduction

Over the past decades the network security has changed with threats becoming far more complex moving from basic attacks against one device to network intrusion attacks against organizations networks. A network intrusion attack is defined as any use of a computer network that compromises network security. Intruders try to gain unauthorized access to files or privileges, modify and destroy the data, or render the computer network unreliable (Aissa & Guerroumi, 2016, 1091). The goal of intrusion detection is to build a system which would scan network activities and generate alerts if either a specific attack occurred or an anomaly in the network behavior detected. Intrusion detection system (IDS) monitors the computer network searching for any suspicious activities that indicate intrusions. In anomaly-based detection base line is what is considered a 'normal' traffic and then flag anything that is not normal as 'abnormal'. The mechanism of anomaly-based IDS depends on the observation to classify input data into classes by adding labels. In a binary classification problem, a single instance can only be divided into two classes. Machine Learning (ML) - based IDS use ML classifiers to learn system normal behavior and build models that help in classifying inputs into the two classes: normal (1) or potentially malicious anomaly (0).

A supervised ML algorithm takes a known set of input data and known responses to generate reasonable predictions for unknown data. In this paper we present four ML algorithms: Gaussian Support Vector Machine (SVM) (Burgess, 1998, 291), Decision Tree (Sebastiani, 2002, 13), k-Nearest Neighbors (k-NN) and weighted k-Nearest Neighbors (wk-NN) (Hechenbichler & Schliep, 2004). k-NN predictions assume that objects near each other are similar. Euclidean distance metric is used to find nearest neighbor. SVM classifies data by finding the linear decision boundary that separates all data

points of one class from those of another class. A decision tree predicts responses to data by following the decisions in the tree from the root down to a leaf node. A tree consists of branching conditions where the value of predictor is compared to a trained weight. However, ML algorithms are computationally expensive if they are trained with the set that has a large number of features. The solution to this problem is to reduce feature space and train classifiers only with the reduced subset. In this paper we present algorithms for feature selection based on preprocessing the Kyoto 2006+ dataset. All categorical features are removed, as well as all features related to the duration of the connection and number of bytes transmitted from source to destination and vice versa. Also, all features containing instances which could not be normalized into the range [-1,1] are cut. Two subsets are generated one consisting of 17 features which contain not-normalized instances and another consisting of nine features of normalized instances. After the training and testing the datasets, accuracy is used to determine performances of the models.

1 Anomaly-Based Intrusion Detection Systems

Anomaly-based IDS monitors computer network to detect intrusion based on irregularities in the pattern with the respect to the normal pattern. It creates a model behavior of the system and then looks for activities that differ from the created model. The anomaly detection approach looks for variations and deviations from an established baseline behavior which might indicate malice. If any anomaly in network activities occurs the IDS warns the system administrator of potentially intrusive action. Anomaly detection can be split into two main methods: machine learning method and rule-based method. ML methods are used to train classifiers in order to recognize what is the notion of normality and then rule-based methods identify abnormal network traffic and flag anomaly.

The main advantage of anomaly based IDS is its ability to detect new attacks even when there is no complete information about the type of attack (Modi et al., 2013, 46). The second advantage is that profile of normal activity is customized for particular computer network and therefore making it very difficult for attacker to know what is certainly what activities it can carry out without getting detected (Patcha & Park, 2007, 3449). One of the biggest advantages of anomaly-based IDS is its ability to detect zero-day attacks since it does not depend on an established signature dataset. The most fundamental challenge is to identify what is normal. Another issue is that even if everything seems like normal over time there are some legitimate anomalies that can be identified as abnormal. Moreover, triage is complex. If one wants to identify an attack an anomaly-based IDS may be very hard to figure out what caused the trigger happened. Furthermore, anomaly-based IDS generates a large number of false positive alarms, since user or network behavior is not always known in advance (Kajal & Devi, 2013, 16). It also requires time to establish baseline behavior when it is first placed in a new network environment or host device. One of the main problems of anomaly-based technique is the selection of the appropriate set of system features because the activities are mostly ad hoc and experience based. Finally, the drawback is also their expensive nature (Garcia-Teodoro et al., 2009, 21).

2 Feature Selection and Instances Normalization

For the purpose of this study two datasets were generated, both based on feature selection and transformation of the Kyoto 2006+ dataset. The Kyoto 2006+ dataset contains daily records of real network traffic data recorded from 2006 to 2009. Each instance in the dataset is labeled with 14 statistical features derived from the KDD Cup ’99 dataset (KDD CUP ’99 dataset, 1999) and 10 additional features which can be used for further analysis and evaluation of the anomaly-based IDS (Protić, 2018, 587-588). The Kyoto 2006+ dataset is given in the Table 1.

Table 1 goes here

The Kyoto 2006+ dataset was captured using honeypots, darknet sensors, e-mail servers, web crawler and other computer network security systems deployed on five networks inside and outside Kyoto University. During the observation period 50.033.015 sessions of normal traffic, 43.043.225 sessions of known attacks and 425.719 sessions of unknown attack were recorded. The dataset consists of both numerical and categorical features.

Complexity of the Kyoto 2006+ dataset is reduced by elimination of irrelevant features and normalization of instances. In this research normalization executed the following transformation on original instance values

$$\mathrm{tansig(n)} = \frac{2}{1 + \mathrm{e}^{-2\cdot \mathrm{n}}} - 1$$

where n represents the number of instances.

The following preprocessing scheme is proposed:

Cut all categorical features - resulting subset contains 17 features (1, 3, 4-17, 24);

Remove statistical features related to the duration of the connection and the number of Source↔Destination bytes (1, 3, 4, 14),

Cut all the features used for further analysis and evaluation of the models (15-17, 24);

Remove features containing instances which could not be normalized into the range [-1, 1] - resulting subset contains nine features (5-13);

Feature 18 ('Label') is used to categorize network traffic into two categories: normal (1) and anomalous (0).

Experiments were carried out on both generated subsets. Number of features in the first subset is approximately three quarters the size of features in the Kyoto 2006+ dataset. Number of features in the second subset contains less than 40% of the original dataset size and is almost a half the size of features in the first subset.

3 Results

In the experiments Classification Learner is used to train and test Gaussian SVM, Decision Tree, k-NN and wk-NN models. Models are chosen because of the following characteristics: prediction speed, memory usage, interpretability and flexibility of the model. (See Table 2)

Table 2 goes here

Experiments were carried out on 10 daily records from the Kyoto 2006+ dataset. The minimum number of records per day was 57.287, while the maximum number of records per day was 158,572.

After the training and testing accuracy was used to determine performances of the models. Accuracy represents the overall success rate, i.e. the ratio between numbers of correct predictions to the total number of classifications, which can be calculated as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

where:

TP (True Positive) represents the number of correctly classified anomaly as anomaly;

FN (False Negative) occurs when classifier incorrectly classifies anomaly as normal behavior;

FP (False Positive) occurs when classifier incorrectly classifies normal behavior as anomaly;

TN (True Negative) represents the number of correctly classifies normal behavior as normal;

Table 3 shows the results of the accuracy for datasets containing nine and 17 features, respectively.

Table 3 goes here

Results show very high accuracy with decision tree giving high values for not-normalized data. In this case accuracy varies from 99.4% to 99.8%. wk-NN gives the highest value for normalized data (99.5%) followed by decision tree (99.3%), Gaussian SVM (98.3%) and k-NN (98.2%) models (see Table 4).

Table 4 goes here

Results show the highest accuracy of the decision tree model and 17 features selected. wk-NN gives the highest value for the second subset. Runtime of the decision tree models is significantly shorter than runtime of other models. Runtime of the second subset is up to four times shorter than runtime of the first subset.

4 Conclusions

Feature selection and instances normalization were used to preprocess Kyoto 2006+ dataset. Two subsets were built, one containing 17 features and not-normalized instances and another containing nine features and normalized instances. Classification Learner was used to train k-NN, wk-NN, Gaussian SVM and the decision tree models. Proposed methods resulted in very high accuracy with decision tree giving the highest accuracy value and the shortest runtime for the subsets containing 17 features. wk-NN method resulted in the highest accuracy value and four times shorter runtime for the subsets consisting nine features.

References

[1] Aissa, N.B. & Guerroumi, M. (2016). Semi-Supervised Statistical Approach for Network Anomaly Detection. Procedia Computer Science, 83, 1090-1095.

[2] Burgess, M. (1998). Computer Immunology. 12th USENIX Conference on System Administration, Boston, MA, USA, 06-11 December 1998, 283-298.

[3] Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G. & Vasquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security 28(2009), 8-28.

[4] Kajal, R. & Devi, S. (2013). Intrusion Detection Systems: A Review. Journal of Network and Information Security, 1(2), 15-21.

[5] KDD CUP '99 dataset. (1999, October 28). Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[6] Hechenbichler, K. & Schliep, K. (2004). Weighted k-Nearest-Neighbor Techniques and Ordinal Classification. Sonderforschungsbereich 386, Paper 399 (2004). Retrieved from https://epub.ub.uni-muenchen.de/1769/1/paper_399.pdf

[7] Modi, C., Patel, D., Borisaniya, H., Patel, A. & Rajaran, M. (2013). A survey of intrusion detection techniques in Cloud. J. Netw. Comput. App. 36(1), 42-57.

[8] Patcha, A. & Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks 51(2007), 3448-3470.

[9] Protić, D. (2018). Review of KDD Cup '99, NSL-KDD and Kyoto 2006+ datasets. Vojnotehnički glasnik/Military Technical Courier, 66(3), 580-596. DOI: 10.5937/vojteh-16670.

[10] Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Comput.Surv. 34(1),1-47.

[11] Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D. & Nakao, K. (2011). Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation. In: Proc.1st Work-shop on Building Anal. Datasets and Gathering Experience Returns for Security. Salzburg. April 10-13, 29-36.

Tables

Table 1 The Kyoto 2006+ dataset

No	Feature	Description
1	Duration – basic	The length of the connection (seconds).
2	Service – basic	The connection's server type (dns, ssh, other).
3	Source bytes – basic	The number of data bytes sent by the source IP address.
4	Destination bytes – basic	The number of data bytes sent by the destination IP address.
5	Count	The numbers of connections whose source IP address and destination IP address are the same to those of the current connection in the past two seconds.
6	Same_srv_rate	% of connections to the same service in the Count feature.
7	Serror_rate	% of connections that have 'SYN' errors in Count feature.

8	Srv_serror_rate	% of connections that have 'SYN' errors in Srv_count (% of connections whose service type is the same to that of the current connections in the past two seconds) features.
9	Dst_host_count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose source IP address is also the same to that of the current connection.
10	Dst_host_srv_count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose service type is also the same to that of the current connection.
11	Dst_host_same_src_port_rate	% of connections whose source port is the same to that of the current connection in Dst_host_count feature.
12	Dst_host_serror_rate	% of connections that have 'SYN' errors in Dst_host_count feature.
13	Dst_host_srv_serror_rate	% of connections that have 'SYN' errors in Dst_host_srv_count feature.
14	Flag	The state of the connection at the time of connection was written (tcp, udp).
15	IDS_detection	Reflects if IDS triggered an alert for the connection; '0' means any alerts were not triggered and an Arabic numeral means the different kind of alerts. Parenthesis indicates the number of the same alert.
16	Malware_detection	Indicates if malware, also known as malicious software, was observed at the connection; '0' means no malware was observed, and string indicates the corresponding malware observed at the connection. Parenthesis indicates the number of the same malware.
17	Ashula_detection.	Means if shellcodes and exploit codes were used in the connection; '0' means no shellcode or exploit code was observed, and an Arabic numeral means the different kinds of the shellcodes or exploit codes. Parenthesis indicates the number of the same shellcode or exploit code
18	Label	Indicates whether the session was attack or not; '1' means normal. '-1' means known attack was observed in the session, and '-2' means unknown attack was observed in the session.
19	Source_IP_Address	Means source IP address used in the session. The original IP address on IPv4 was sanitized to one of the Unique Local IPv6 Unicast Addresses. Also, the same private IP addresses are only valid in the same month; if two private IP addresses are the same within the same month, it means their IP addresses on IPv4 were also the same, otherwise are different.
20	Source_Port_Number	Indicates the source port number used in the session.
21	Destination_IP_Address	It was also sanitized.
22	Destination_Port_Number	Indicates the destination port number used in the session.
23	Start_Time	Indicates when the session was started.
24	Duration	Indicates how long the session was being established.

(Source: Song et al., 2011)

Table 2 Classifiers characteristics

Model	Prediction speed	Memory usage	Interpretability	Model Flexibility
Tree	Fast	Low	Easy	Medium. Medium number of leaves for finer distinctions between classes (maximum number of splits is 20).
SVM	Fast	Medium	Hard	Medium. Medium distinctions, with kernel scale set to sqrt(P).
k-NN	Medium	Medium	Hard	Medium. Medium distinctions between classes. The number of neighbors is set to 10.
wk-NN	Medium	Medium	Hard	Medium. Medium distinctions between classes using a distance weight. The number of neighbors is set to 10.

(Source: MathWorks, 2016.)

Table 3 Accuracy of medium k-NN, wk-NN, medium Gaussian SVM and medium decision tree models

No	Size	Model	Accuracy - 9 features	Runtime	Accuracy - 17 features	Runtime
1	158572	k-NN	98.3%	275.72s	99.0%	1000.8s
		wk-NN	98.4%	277.32s	99.1%	1019.15s

		SVM	98.1%	449.35s	98.4%	467.7s
		Tree	97.2%	3.8452s	98.4%	14.241s
2	129651	k-NN	91.8%	175.84s	98.8%	695.88s
		wk-NN	91.8%	173.32s	99.0%	691.08s
		SVM	98.3%	254.32s	98.4%	304.56s
		Tree	97.3%	3.3104s	99.7%	9.4989s
3	128740	k-NN	98.2%	193.82s	98.6%	682.07s
		wk-NN	98.1%	194.81s	98.8%	690.58s
		SVM	97.8%	280.82s	97.9%	379.61s
		Tree	97.2%	3.3033s	99.8%	9.5367s
4	136625	k-NN	99.3%	194.83s	99.5%	782.1s
		wk-NN	99.4%	194.23s	99.7%	788.11s
		SVM	99.1%	217,32s	99.3%	234.59s
		Tree	98.3%	8.3169s	99.7%	10.001s
5	90128	k-NN	99.0%	101.28s	98.5%	731.2s
		wk-NN	99.1%	101.753s	99.6%	744.15s
		SVM	99.0%	86.283s	99.3%	230.33s
		Tree	98.4%	2.2308s	99.7%	10.855s
6	93999	k-NN	96.5%	109.25s	99.4%	354.09s
		wk-NN	96.5%	108.77s	99.5%	351.55s
		SVM	98.0%	111.83s	98.4%	149.03s
		Tree	97.5%	2.2613s	99.5%	6.9921s
7	80807	k-NN	98.8%	91.25s	99.4%	285.77s
		wk-NN	98.8%	91.267s	99.5%	285.25s
		SVM	97.9%	227.28s	98.1%	125.25s
		Tree	98.9%	2..2615s	99.4%	6.2339s
8	57278	k-NN	99.6%	42.704s	99.3%	77.224s
		wk-NN	99.3%	43.235	99.3%	77.121s
		SVM	99.2%	33.754s	99.1%	31.211s
		Tree	99.3%	1.743s	99.4%	3.7448s
9	58317	k-NN	99.1%	31.714s	99.3%	133.92s
		wk-NN	99.2%	31.738s	99.4%	134.4s
		SVM	99.1%	34.234s	99.2%	36.907s
		Tree	98.9%	1.7482s	99.5%	4.4372s
10	57278	k-NN	99.4%	43.734s	99.6%	129.99s
		wk-NN	99.5%	43.272s	99.6%	130.88s
		SVM	99.2%	30.239s	99.3%	37.894s
		Tree	99.4%	1.7489s	99.7%	4.4535s

Table 4 The highest accuracy of k-NN, wk-NN, Gaussian SVM and decision tree models

Model	Accuracy - 9 features	Runtime	Accuracy - 17 features	Runtime
k-NN	98.2%	193.82s	99.6%	129.99
wk-NN	99.5%	43.272s	99.7%	788.11s
SVM	98.3%	254.32s	99.3%	37.894s
Tree	99.3 %	1.743s	99.8%	9.5367s