Comparison of Intrusion Detection Systems based on Machine Learning and Data Mining Algorithms for Low-Powered Devices

Project progress report submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

of

Bachelor of Technology

in

Computer and Communication Engineering

by

Pratyay Amrit
Reg. No. 140953430

Under the guidance of

Ms. Ipsita Upasana
Assistant Professor
Department of I & CT
Manipal Institute of Technology
Manipal, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Jan 2018

1 Introduction

Security in low-powered devices has always been a major concern as their limited computational capacity prohibits the use of sophisticated algorithms. An Intrusion Detection System(IDS) provides a first-line monitoring service for the devices to help them protect themselves against anomalous data. Several methods have been proposed to implement such a system, ranging from Association Rule Mining to complex Machine Learning algorithms. Most of these systems, however, fail to address the issue of low computational capabilities of small devices, which may soon comprise the majority of man-machine interaction with the ever growing popularity of IoT. This work attempts to analyze and compare the performance of some of these methods. UNSW-NB15 dataset has been used for testing these methods. The KDD'99 dataset has been popular for the development of such systems, however, the dataset has a few problems. UNSW-NB15 is a recently curated dataset that has been gaining popularity in recent years. It provides a more modern set of data and features to be used in IDSs for detecting modern attacks. It also addresses some of the issues in KDD'99 [1].

2 Problem Definition

Technology is growing rapidly every day, and so are incidents concerned with cyber security. Intrusion Detection Systems are responsible for monitoring the network and detecting a data packet that may pose a risk for the network. Primitive IDSs were built to detect attacks on the basis of their signature. Every time a new attack was discovered, it's signature was recorded to protect the system from similar future attacks. This, however, could not prevent the system from new attacks. With the advent of Machine Learning and Data Mining techniques, these IDSs could now be taught how an attack looks in the network to perform detection of new attacks. This however, significantly increased the false positives in detection. Another problem came up as IoT began expanding exponentially. Massive networks of small, low-powered devices have taken over the majority of space in todays world. These small devices, however, lack the ability to execute complicated machine learning algorithms. This work attempts to assess the performance of some of these algorithms in such low powered devices.

3 Objective

The objectives of this work can be summarized as follows:

1. To find and choose a dataset that will resemble a network dump as closely as possible, and must conform with modern trends to make sure the IDS is well trained for novel attacks.

2. To develop IDSs based on [2] and [3]. These works propose two very popular classification algorithms, K-Means and Multilayer Perceptron respectively.

3. To analyze and compare the two algorithms based on similar parameters. The accuracy and efficiency of these algorithms must be taken in to account.

4. To test these algorithms in low powered devices and compare the effects of the decreased computational power.
5. To make an attempt to scale down the complexity of these algorithms to make it more efficient for low powered devices.
6. To analyze the optimized algorithms and compare them

4 Scope

This work will benefit cyber security companies or agencies to understand how different types of IDS perform on similar environments and may use this work to influence their decision in choosing one. This work will help individuals working in IoT or cyber security domain in making the right choice for their specific scenarios. This work may also help future research in developing an all round hybrid IDS capable of performing equally well in both, high end and low powered devices, as well as analyzing and comparing other numerous algorithms used for developing Intrustion Detection Systems.

5 Literature Survey

[1] is a comparative study of the two most popular datasets used for testing Intrusion Detection Systems. It explains the shortcomings of the KDD'99 dataset while introducing the UNSW-NB15 dataset. In this paper, the feature characteristics of the UNSW-NB15 and KDD'99 datasets are examined, and the features of the UNSW-NB15 are replicated to the KDD99 data set to measure their effeciency, they apply An Association Rule Mining algorithm as feature selection to generate the strongest features from the two datasets.

[2] proposes the use of k-means algorithm for IDS. The algorithm is tested with various number of clusters and the best result was shown to be when the number of clusters equalled 22 for the static database the algorithm was tested with. In a dynamic network, however, the number of clusters will greatly affect the results for different data sets and thus, identifying the number of clusters becomes an important issue.

[3] proposes the use of multi-layer perceptrons for building an IDS. The model performs better than other methods, however, is computationally expensive. The authors lowered the computational load by binarizing the weights on the neural network. This allowed the use of this algorithm in low powered devices, but also resulted in a drop in performance by 4 times.

6 Methodology

6.1 Dataset

The UNSW-NB15 dataset has recently been released. It was generated using tcpdump tool to capture 100GB of raw data. It has the following 9 types of attacks:

1. Fuzzers: Fuzzing, or Fuzz Testing is a black box testing technique in which aims at finding implementation bugs by injecting malformed or semi-malformed data in an automated fashion. Potential attackers may run fuzz tests on the network to find vulnerabilities that can be exploited.

2. Analysis: Traffic analysis is a special type of inference attack technique that looks at communication patterns between entities in a system. Knowing who's talking to whom, when, and for how long, can sometimes clue an attacker in to information of which you'd rather she not be aware.

3. Backdoor: A backdoor is a malware type that negates normal authentication procedures to access a system.

4. DoS: Denial of Service (DoS) is an attack which aims to make a host machine unavailable to its intended users by temporarily or indefinitely disrupting the host machines connection to the internet.

5. Exploit: An exploit is a piece of software, a chunk of data, or a sequence of commands that takes advantage of a bug or vulnerability to cause unintended or unanticipated behavior to occur on computer software, hardware, or something electronic.

6. Generic: This subset contains any general type of attack such as probe, u2r, r2l etc.

7. Reocinnaissance: Active reconnaissance is a type of computer attack in which an intruder engages with the targeted system to gather information about vulnerabilities.

8. Shellcode: A shellcode is a small piece of code used as the payload in the exploitation of a software vulnerability.

9. Worm: A computer worm is a standalone malware computer program that replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it.

The dataset contains 2,540,044 records with 49 features categorized into 6 categories.

1. **Flow Features:**
   1. srcip: Source IP address.
   2. sport: Source port number.
   3. dstip: Destinations IP address.
   4. dsport: Destination port number.
   5. proto: Protocol type, such as TCP, UDP.

2. **Basic Features:**
   6. state: The states and its dependent protocol e.g., CON.
   7. dur: Row total duration.
   8. sbytes: Source to destination bytes.
   9. dbytes: Destination to source bytes.

10. sttl: Source to destination time to live.
11. dttl: Destination to source time to live.
12. sloss: Source packets retransmitted or dropped.
13. dloss: Destination packets retransmitted or dropped.
14. service: Such as http, ftp, smtp, ssh, dns and ftp-data.
15. sload: Source bits per second.
16. dload: Destination bits per second.
17. spkts: Source to destination packet count.
18. dpkts: Destination to source packet count.

3. **Content Features:**
19. swin: Source TCP window advertisement value.
20. dwin: Destination TCP window advertisement value.
21. Stcpb: Source TCP base sequence number.
22. dtcpb: Destination TCP base sequence number.
23. smeansz: Mean of the packet size transmitted by the srcip.
24. dmeansz: Mean of the packet size transmitted by the dstip.
25. trans_depth: The connection of http request/response transaction.
26. res_bdy_len: The content size of the data transferred from http.

4. **Time Features:**
27. sjit: Source jitter.
28. djit: Destination jitter.
29. stime: Row start time.
30. ltime: Row last time.
31. sintpkt: Source inter-packet arrival time.
32. dintpkt: Destination inter-packet arrival time.
33. tcprtt: Setup round-trip time, the sum of synack and ackdat.
34. synack: The time between the SYN and the SYN_ACK packets.
35. ackdat: The time between the SYN_ACK and the ACK packets.
36. is_sm_ips_ports: If srcip (1) = dstip (3) and sport (2) = dsport (4), assign 1 else 0.

5. **Additional Generated Features:**
37. ct_state_ttl: No. of each state (6) according to values of sttl (10) and dttl (11).
38. ct_flw_http_mthd: No. of methods such as Get and Post in http service.
39. is_ftp_login: If the ftp session is accessed by user and password then 1 else 0.
40. ct_ftp_cmd: No of flows that has a command in ftp session.
41. ct_srv_src: No. of rows of the same service (14) and srcip (1) in 100 rows.
42. ct_srv_dst: No. of rows of the same service (14) and dstip (3) in 100 rows.
43. ct_dst_ltm: No. of rows of the same dstip (3) in 100 rows.
44. ct_src_: ltm No. of rows of the srcip (1) in 100 rows.
45. ct_src_dport_ltm: No of rows of the same srcip (1) and the dsport (4) in 100 rows.
46. ct_dst_sport_ltm: No of rows of the same dstip (3) and the sport (2) in 100 rows.
47. ct_dst_src_ltm: No of rows of the same srcip (1) and the dstip (3) in 100 records.

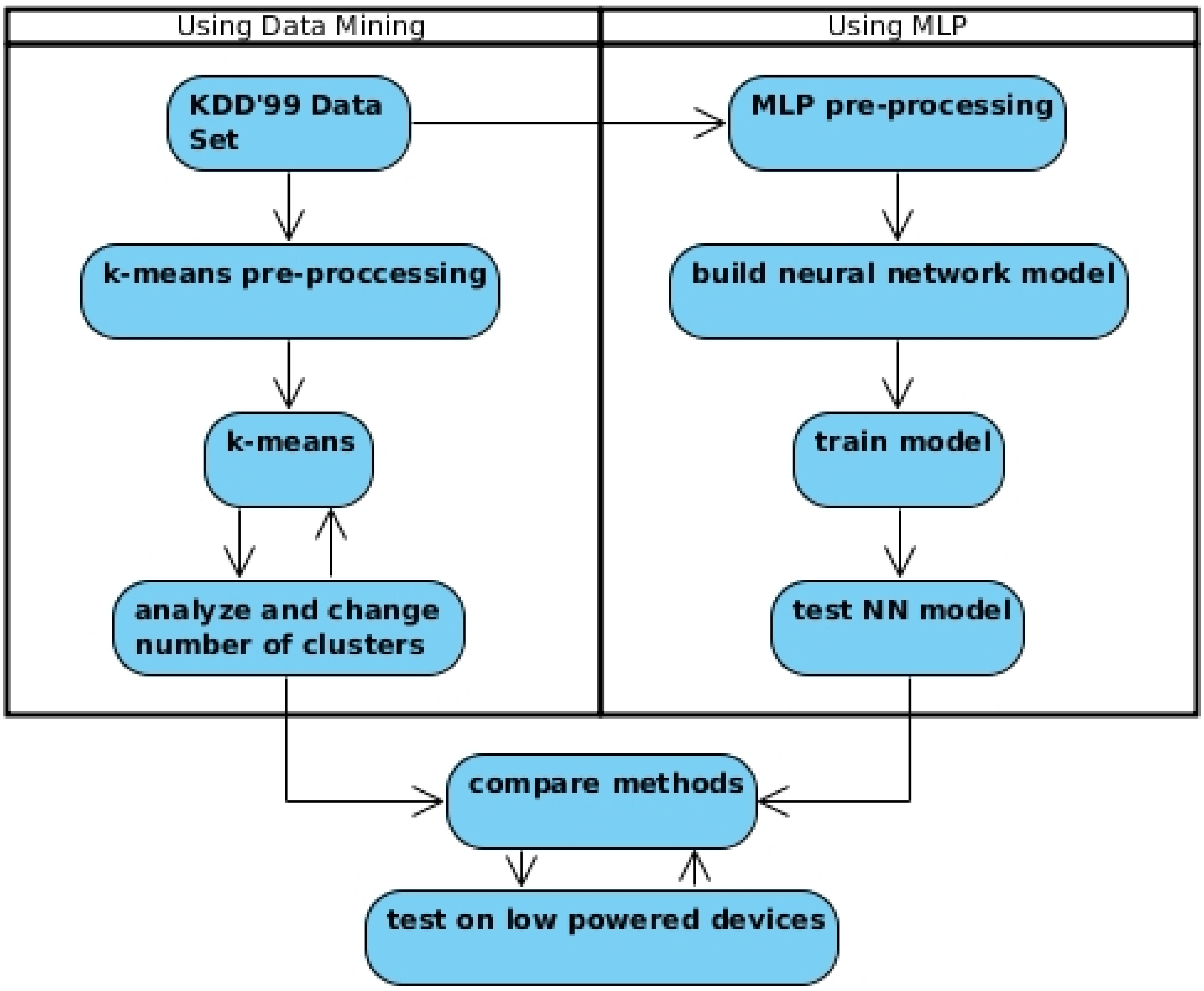


Figure 1: Flow diagram of actions to be performed

6. **Labelled Features:**
   48. attack_cat: The name of each attack category.
   49. label: 0 for normal and 1 for attack records.

6.2 Pre-processing

The following steps were followed to process the data for the algorithms:

1. Import dataset using python pandas library
2. Fill empty values (attack_cat = '' for noraml data) with 0
3. Transform nominal data values to numeric
4. Add feature names to the pandas dataframe (only for readability while debugging)

6.3 K-Means

The goal of this work is to minimize computational load as much as possible. Therefore, while training and testing the datasets, it has been assumed that classifying an attack as an attack has more priority than classifying which type of attack the data packet resembles. This allowed the use of a minimum of 2 clusters for K-Means, one for attacks, the other for normal data. Luckily, the dataset has already been labeled in this manner as its 49th feature (label), which may be used for training and testing.

6.4 Multilayer Perceptron

It has been seen that, to minimize computational load in a neural network, binarizing the weights is possible, at the cost of accuracy of prediction [3]. This attempt shall be made and analyzed with different configurations of hidden layers.

6.5 Performance Paramters

4 generic paramters will be used for comparing the two algorithms:

- True Positive (TP): Number of correctly classified attack records
- True Negative (TN): Number of correctly classified normal records
- False Positive (FP): Number of misclassified attack records
- False Negative (FN): Number of misclassified normal records

Using these 4 paramters, the accuracy may be calculated as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

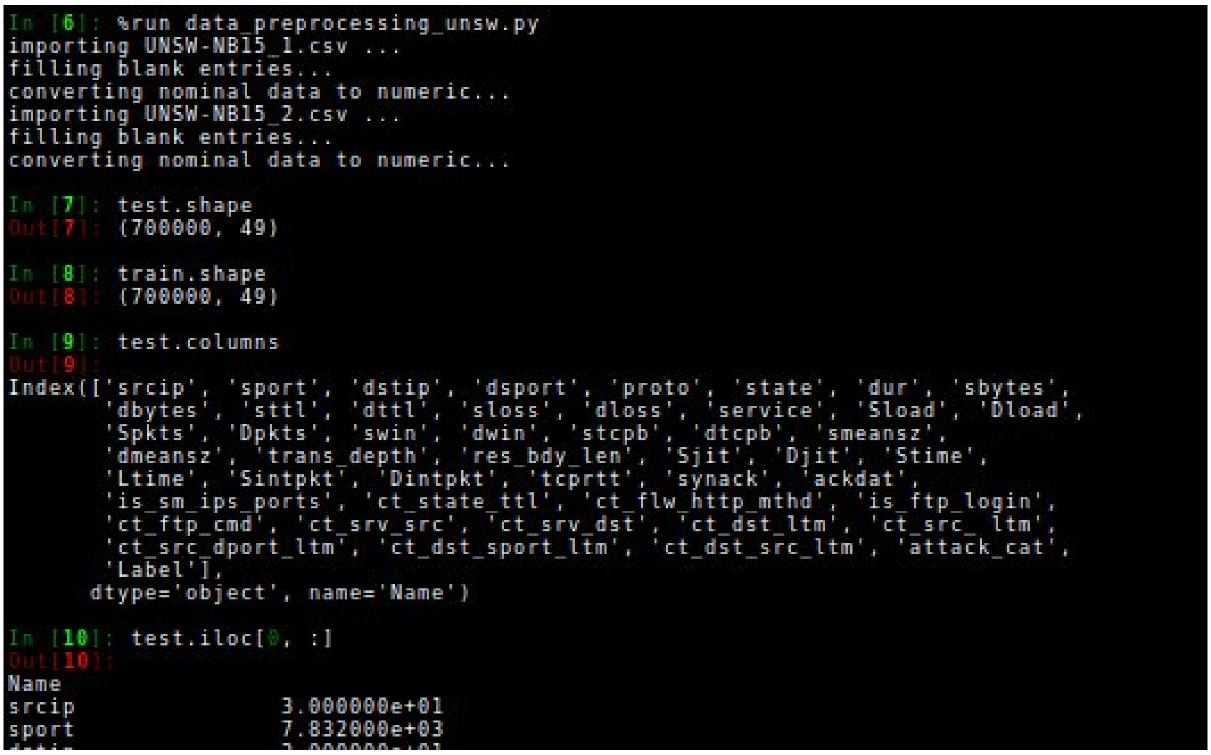
7 Work done so far

7.1 Dataset

The choice of dataset was between 2 very popular ones used for testing IDSs, the KDD'99 and UNSW-NB15. Initial tests were run on the KDD'99, however, it had many problems as stated in [1]. As a result, another, more modern data set called UNSW-NB15, also used for IDS performance testing, was used. The dataset is much larger and carefully chosen training and testing data sets are available at their website. Te data set is modern and goes well with the present standards and tackles most of the issues of KDD99, and has proven to be a beter performer according to [1].

7.2 Pre-processing

The pre-processing of the dataset has been done using pandas library for python, following the steps mentioned in section 6.2. Figure 2 shows the results of running the script for preprocessing the data in an ipython terminal.

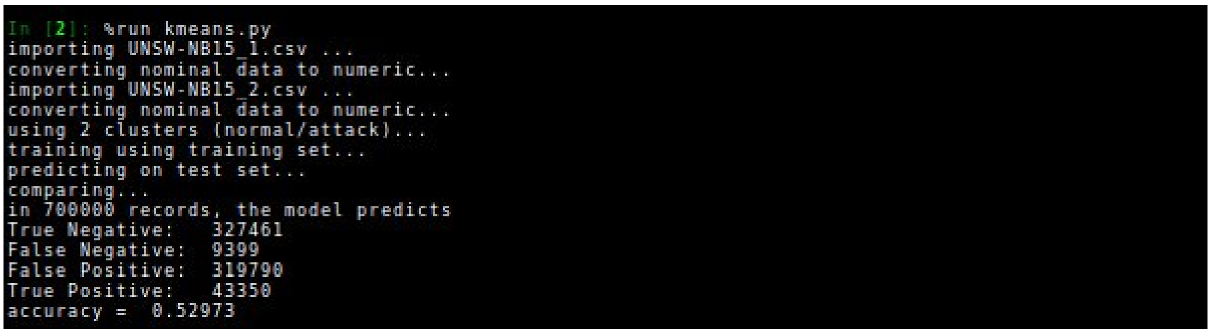


```
In [6]: %run data_preprocessing_unsw.py
importing UNSW-NB15_1.csv ...
filling blank entries...
converting nominal data to numeric...
importing UNSW-NB15_2.csv ...
filling blank entries...
converting nominal data to numeric...

In [7]: test.shape
Out[7]: (700000, 49)

In [8]: train.shape
Out[8]: (700000, 49)

In [9]: test.columns
Out[9]:
Index(['srcip', 'sport', 'dstip', 'dsport', 'proto', 'state', 'dur', 'sbytes',
       'dbytes', 'sttl', 'dttl', 'sloss', 'dloss', 'service', 'Sload', 'Dload',
       'Spkts', 'Dpkts', 'swin', 'dwin', 'stcpb', 'dtcpb', 'smeansz',
       'dmeansz', 'trans_depth', 'res_bdy_len', 'Sjit', 'Djit', 'Stime',
       'Ltime', 'Sintpkt', 'Dintpkt', 'tcprtt', 'synack', 'ackdat',
       'is_sm_ips_ports', 'ct_state_ttl', 'ct_flw_http_mthd', 'is_ftp_login',
       'ct_ftp_cmd', 'ct_srv_src', 'ct_srv_dst', 'ct_dst_ltm', 'ct_src_ ltm',
       'ct_src_dport_ltm', 'ct_dst_sport_ltm', 'ct_dst_src_ltm', 'attack_cat',
       'Label'],
      dtype='object', name='Name')

In [10]: test.iloc[0, :]
Out[10]:
Name
srcip                3.000000e+01
sport                7.832000e+03
```

Figure 2: Preprocessing the UNSW-NB15 dataset using part 1 as training set and part 2 as test set.

7.3 K-Means

The execution of K-Means algorithm was done using 2 clusters to achieve an accuracy of 52.97%. The algorithm needs ot be further optimized further to achieve more acceptable results. In 700,000 records, the model detected 327,461 TN, 9,399 FN, 319,790 FP and 43,350 FN. The results are shown in Figure 3. It is clear from the results that in anomaly based IDSs, the false positive rate is very high that has a major contribution in reducing the accuracy of the system.



```
In [2]: %run kmeans.py
importing UNSW-NB15_1.csv ...
converting nominal data to numeric...
importing UNSW-NB15_2.csv ...
converting nominal data to numeric...
using 2 clusters (normal/attack)...
training using training set...
predicting on test set...
comparing...
in 700000 records, the model predicts
True Negative:   327461
False Negative:  9399
False Positive:  319790
True Positive:   43350
accuracy =  0.52973
```

Figure 3: Results of running k-means on UNSW-NB15

8 Remaining work

The following work has to done with their target dates:

- 20 Mar 2018: Implement MLP on the basis of [3] on UNSW-NB-15. More reading on this subject is required to understand the basis on which the number of layers are chosen.
- 25 Mar 2018: Analyze and compare the two algorithms on regular devices. This is important to set a benchmark score for the two algorithms for when tested on low powered devices. A correct parameter for deciding the efficiency of the algorithm must be defined on the basis of compute power and time spent on performing these algorithm, while taking their accuracy into account.
- 30 Mar 2018: Test on low powered devices to understand the feasibility of these systems on such devices.
- 15 Apr 2018: Make an attempt to scale down algorithms for low powered devices.
- 01 May 2018: Re-run tests and understand pros and cons of scaling down.

References

[1] N. Moustafa and J. Slay, "The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems," in *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, Nov 2015, pp. 25–31.

[2] S. Duque and M. N. bin Omar, "Using data mining algorithms for developing a model for intrusion detection system (ids)," *Procedia Computer Science*, vol. 61, pp. 46 – 51, 2015, complex Adaptive Systems San Jose, CA November 2-4, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915029750

[3] P. V. S. Alpao, J. R. I. Pedrasa, and R. Atienza, "Multilayer perceptron with binary weights and activations for intrusion detection of cyber-physical systems," in *TENCON 2017 - 2017 IEEE Region 10 Conference*, Nov 2017, pp. 2825–2829.