A NOVEL APPROACH FOR NETWORK INTRUSION DETECTION USING EXTREME GRADIENT BOOSTING AND LONG SHORT-TERM MEMORY RECURRENT NEURAL NETWORKS

A THESIS PRESENTED BY

R. E. RATNAYAKE

to the

DEPARTMENT OF STATISTICS AND COMPUTER SCIENCE

in partial fulfillment of the requirement

for the award of the degree of

BACHELOR OF SCIENCE (HONORS) IN COMPUTER SCIENCE

of the

UNIVERSITY OF PERADENIYA

SRI LANKA

2019

DECLARATION

I do hereby declare that the work reported in this thesis was exclusively carried out by me under the supervision of Dr. Hakim Usoof. It describes the results of my own independent research except where due reference has been made in the text. No part of this thesis has been submitted earlier or concurrently for the same or any other degree.

Date: ……. ……..

Signature of the Candidate

Certified by:

Supervisor: Dr. Hakim Usoof Date: ……

Signature: ……

A NOVEL APPROACH FOR NETWORK INTRUSION DETECTION USING EXTREME GRADIENT BOOSTING AND LONG SHORT-TERM MEMORY RECURRENT NEURAL NETWORKS

R. E. Ratnayake

Department of Statistics and Computer Science, University of Peradeniya, Peradeniya, Sri Lanka

A novel approach for network intrusion detection was presented using Extreme Gradient Boosting (XGBoost) for feature selection and then using the selected features in a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) classifier. The purpose of this research was to create a number of minimal feature sets using XGBoost and study the effects of using them in a LSTM RNN model for detecting whether or not the network features belong to an attack. NSL-KDD dataset, which is an improved version of KDD Cup '99 dataset was used. After preprocessing all the features in the dataset, they were fed into XGBoost and feature importance was calculated. Based on the scores for importance obtained by the features, minimal feature sets were created. Then, LSTM recurrent neural network classifier was applied using all the features in the dataset and the minimal feature sets, separately. The results of applying the LSTM on all the features and only the feature sets created using XGBoost were finally compared. According to the experimental results, it was observed that XGBoost feature selection could be used to create minimal feature sets with very high feature reduction ratios to use in a LSTM RNN model for network intrusion detection to achieve a good accuracy value close to that achieved using all the features in the dataset. The findings of this study can be used for building better network intrusion detection systems using deep neural networks for real-time network intrusion detection. Also, they can be utilized to develop a first layer of defense for alerting the users about possible threats in real-time.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Hakim Usoof, for his support and guidance for this work throughout the research project. I would also like to acknowledge my gratitude to Prof. Saluka Kodithuwakku and Dr. Ruwan Nawarathna, for their encouragement and concern on our academic requirements. I also convey my gratitude to all the staff members in the Department of Statistics and Computer Science who helped me by giving advice.

Finally, the completion of this research project would not have been possible without the assistance of my parents. Their contribution is sincerely appreciated and gratefully acknowledged.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1)	AE	-Auto-Encoder
2)	ANN	- Artificial Neural Network
3)	BFGS	- Broyden–Fletcher–Goldfarb–Shanno Algorithm
4)	DLMC	- Distributed Machine Learning Community
5)	DR	- Detection Rate
6)	FAR	- False Alarm Rate
7)	GA	- Genetic Algorithm
8)	GPU	- Graphics Processing Unit
9)	ICT	- Information Communication Technology
10)	IDS	- Intrusion Detection System
11)	KDD	- Knowledge Discovery in Databases
12)	KNN	- K-Nearest Neighbor
13)	KPCA	- Kernel Principal Component Analysis
14)	LDA	-Linear Discriminant Analysis
15)	LM	- Levenberg-Marquardt
16)	LSTM	- Long Short-Term Memory
17)	NB	- Naïve Bayesian
18)	NIDS	- Network Intrusion Detection System
19)	NSL-KDD	- Network Socket Layer-Knowledge Discovery in Databases
20)	PCA	- Principal Component Analysis
21)	QDA	- Quadratic Discriminant Analysis
22)	RAM	- Random Access Memory
23)	RF	- Random Forest
24)	RNN	- Recurrent Neural Network
25)	SAE	- Sparse Auto-Encoder
26)	STL	- Self-Taught Learning
27)	SVM	- Support Vector Machine
28)	TCP	- Transmission Control Protocol
29)	XGBoost	- Extreme Gradient Boosting

CHAPTER 1

INTRODUCTION

1.1 Network Intrusion Detection

We are living in the era of computer systems and networks. Network intrusion detection is a very important subject nowadays since increasingly powerful technologies keep emerging and people are so much integrated with networks and computer systems in the current world. A very good example is the internet which connects the world forming a global village. We use the internet for basically everything: social networking, business, education, entertainment etc. This leaves us more vulnerable towards attacks and the various security threats that we are facing are becoming more and more serious day by day. With concepts like virtualization and cloudification coming into play, more and more data are being added to the internet, and "security" remains a key issue. Therefore, security mechanisms we use against threats and attacks need to evolve too.

Many processes like monitoring, identifying, analyzing and managing the events within infrastructure that may affect the security, together make up intrusion detection. In order to ensure the reliability of a system, effective and efficient intrusion detection systems are needed. Attackers who attack computer networks leave small traces of their presence in network traffic. Network traffic is reflected by networking devices like switches, routers, and firewalls, or by network taps that are placed directly on the physical transmission media, such as copper wire or optical fiber. If we analyze this traffic carefully, it is possible to identify these traces of malicious behavior and hence identify attacks.

There are two main types of intrusion detection techniques used in network intrusion detection systems: Misuse-based (Signature-based) detection and Anomaly-based detection. The major difference between an anomaly-based IDS and a signature-based IDS is that the latter is most effective in protecting against attacks and malware that have already been identified and thus are known. Misuse-based detection works on offline data and can catch unknown attacks while anomaly detection can work well with online data and can detect any abnormal behavior. Most IDSs rely on misuse detection technique and since it depends on attack signatures for successfully detecting attacks, if we do not have a signature for an attack, we may not detect it

at all. Hence, they are unable to defend against new attack types and variants of old types of attacks.

1.2 Research Problem Statement

There is a solid necessity for dependable information and communication technology (ICT) functionalities in today's world and this calls for novel security mechanisms. The study is on intrusion detection systems that can detect known attacks as well as new attacks, with a high detection rate. Ideally, the false alarm rate of an IDS should be 0%. However, it is not an easy task to observe huge amounts of network data in real-time. Furthermore, new attacks are being invented, and many forms of old attacks are being modified into many variants, and weaknesses in security in network protocols and software are being used by attackers. All of these make it necessary to scan for unknown patterns in network traffic which is only possible using anomaly-based intrusion detection systems.

As much as it is important to achieve higher accuracies and lower false alarm rates in network intrusion detection, it is also very important that the detections can be done very fast. Presently, the challenge is to construct intrusion detection systems based on deep neural networks; frameworks that require a minimal input feature set with reasonable complexity from network traffic information, and with the most elevated conceivable effectiveness with respect to recognizing novel attacks to computers and networks.

The best way to do this is by carefully selecting the most important features that affect key decisions that lead to finally detecting whether the record belongs to an attack or not, so that using only those features, a comparable accuracy with better performance can be achieved. Therefore, in this study, in investigating the possibilities of data mining, the main focus was in the effects of reduced feature sets on the network intrusion detection performance.

1.3 Research Questions

The main research question addressed in this research project is whether feature selection using Extreme Gradient Boosting (XGBoost) can be used for improved network intrusion detection. This question implies the following sub-questions:

1. Can the feature importance scores calculated using XGBoost be used for feature selection for a long short-term memory recurrent neural network classifier for network intrusion detection?

2. Does the feature selection show an improved performance and an acceptable accuracy compared to using all the features in the dataset?

1.4 Research Objectives

The objectives of the research are to propose a novel approach for anomaly-based network intrusion detection which would give good results with better performance in terms of time and memory consumption. The method chosen in order to achieve this, is feature reduction which decreases the amount of computation to be done than when performing classification using all the features in the dataset. This research is aimed at the study of whether XGBoost can be used as a feature selection method to improve network intrusion detection using a LSTM RNN model on the NSL-KDD dataset.

1.5 Thesis Overview

In chapter 2, the relevant details of previous research and studies done on network intrusion detection are summarized. Chapter 3 starts off with brief introductions of the dataset which is NSL-KDD dataset, and the main models used in this research which are XGBoost model and LSTM RNN model. Then, the methodology followed starting from data preprocessing is presented in detail. In Chapter 4, the experimental results obtained during this research project are presented. The results of experiments done with LSTM RNN model using all features and using the minimal feature sets created using results of XGBoost implementation are compared and discussed. In the final chapter, Chapter 5, the conclusions of the research project are stated and some future work is suggested.

CHAPTER 2

LITERATURE REVIEW

2.1 Prior Studies Relevant to Network Intrusion Detection

Network Intrusion Detection (NID) is an Intrusion Detection mechanism by which attempts are made to discover unauthorized access to a computer network. This is done by analyzing network traffic for signs of malicious or suspicious activity. The very first time in which a publication appeared regarding Intrusion Detection (ID) was in the mid 1980's. It was about Denning's work about Intrusion Detection (ID) and where his main assumption has been that intrusive activities are different from the usual activities and that by discovering models for normal behavior, intrusive activities can be identified. Ever since, many approaches have been proposed and prototypes have been developed for commercial as well as military applications [1].

Intrusion detection, nowadays, is a challenging task due to the proliferation of different types of computer networks since the connectivity of computer systems has increased giving greater access to outsiders and making it easier for intruders to avoid identification. Hence, machine learning based methods have become popular in network intrusion detection.

In previous studies related to this regard, a number of traditional machine learning based approaches have been proposed, including Support Vector Machines (SVM) [2], K-Nearest Neighbor (KNN) [3], Artificial Neural Networks (ANN) [4], Random Forest (RF) [5], [6] and others [7], [8]. In [2], a SVM model combined with KPCA and GA has been proposed and the results have shown that the proposed model achieves better generalization performance than when using a SVM classifier by using feature extraction using PCA. In [3], KNN classification algorithm has been used for network intrusion detection in a wireless sensor network. Results using the proposed system has shown that it can prevent flooding attacks efficiently. An ANN has been used in [4] to evaluate the performance of NSL-KDD dataset. It has used Levenberg-Marquardt (LM) and BFGS quasi-Newton Backpropagation algorithm for learning. Results when compared to other reported papers before that, have shown that it has given a higher accuracy. In [5] and [6], Random Forests have been used. Results have shown that the models proposed are efficient and gives high detection rates and low false alarm rates. The authors in [7] have presented a survey on the primary intrusion detection techniques and emphasized the

use of data mining algorithms to implement IDS such as SVM, Kernelized SVM, Extreme Learning Machine and Kernelized Extreme Learning Machine. They have also suggested that combining more than one datamining algorithm may be used in order to reduce disadvantages of one another. In [8], another survey has been presented and the authors of that have focused on machine learning and data processing strategies for cyber intrusion detection in wired networks. They have also proposed an approach to identify the representative system call patterns for users by using data mining and forensic techniques.

Deep learning too has been applied for intrusion detection and has become highly popular since they show results that completely surpass those of traditional methods. In [9], the authors have taken an approach based on a deep neural network for flow-based anomaly detection. The results prove that deep learning can be applied for anomaly detection in software defined networks. In [10], the authors have proposed an approach using self-taught learning (STL), which is also a deep-learning based method, on the benchmark NSL-KDD dataset. The results when compared to previous studies, have shown that this method is much more effective in terms of performance. In [11], the authors have proposed a three-layer RNN architecture misuse-based IDS. However, since it is a reduced-size recurrent neural network, the nodes of layers are partially connected and also it does not show the ability of deep learning to model high-dimensional features. Further, the authors have not studied the performance of the model for binary classification. According to [12], the authors have proposed another deep learning-based approach using recurrent neural networks. They have experimented on the NSL-KDD dataset with separate training and testing set to evaluate their performances in detecting network intrusions in both binary and multiclass classification and results have shown that it gives a better accuracy when compared to J48, ANN, RF, SVM and other machine learning methods proposed by previous researchers. In [13], the author has modeled network intrusions using LSTM RNNs and claimed with proof that modelling network traffic as a time series with a supervised learning approach, using known attack and non-attack type behavior, improves intrusion detection. It is the first time that the usefulness of LSTM networks to intrusion detection was demonstrated and it was in 2012. The author has used KDD Cup ’99 dataset and results have shown that the LSTM RNN classifier provides superior performance when compared to other strong static classifiers trained.

Lately, XGBoost has been used in many researches. Researchers have compared the implementation of XGBoost and the RF classification method on datasets to find the better

classification model. The results do give XGBoost an advantage over the RF classification method [14]. Moreover, XGBoost can perform classification in parallel on multiple operating systems [15]. In 2015, XGBoost was used by every winning team in the top 10 in the KDD Cup competition. In [16], XGBoost model has been used for intrusion detection on the NSL-KDD dataset and the results have been compared with other classification models. Results show that the accuracy achieved by XGBoost model is the best when compared to the other models employed on the same dataset. However, the training and testing dataset of the NSL-KDD dataset have been combined and the experiments have been performed on the combined dataset.

2.2 Feature Reduction-Related Work

For decades, feature reduction methods have been used in experiments for better performance. The authors in [17] used Sparse Auto-Encoder (SAE) for feature learning and dimensionality reduction with SVM on the NSL-KDD dataset [18]. The efficiency of that model has been studied in both binary and multiclass classification and also compared with methods like J48, Naïve Bayesian, Random Forest, and SVM. Results have shown that the effect of SAE has accelerated SVM training and testing times and performed better in terms of performance metrics. The researchers in [19] defined the reduction rate and studied the efficiency of PCA for intrusion detection. The authors have used KDD Cup ’99 dataset and UNB ISCX. The experimental results have shown that the first 10 Principal Components are effective for classification and achieved a classification accuracy nearly same as using all the features in KDD and 28 features in ISCX. In [20], two feature dimensionality reduction approaches: Auto-Encoder (AE) and PCA, have been used. The resulting low-dimensional features from both techniques have then been used to form different classifiers like RF, Bayesian Network, Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). The dataset used is CICIDS2017 and the findings of that research have shown that low-dimensional features used gave better performance in terms of Detection Rate (DR), F-Measure, False Alarm Rate (FAR) and Accuracy. The In [21], the authors have categorized the attributes in the KDD dataset into 4 classes: Basic, Content, Traffic and Host, and generated fifteen variants of the dataset by forming all combinations of four classes. They have then studied the dominance of each class of attributes by simulating on Random Tree algorithm. The author in [13] has ranked all the features using information gain and then used a combined approach using decision tree pruning, biased backward elimination and forward selection, and also heuristic domain

knowledge to detect and remove unnecessary features. Results have proved it as an effective technique for finding small feature sets with 4-8 important features.

However, in previous studies, XGBoost has not been used as a feature selection method for the network intrusion detection problem. Also, most previous studies done using XGBoost on NSL-KDD dataset have been done by combining the train and test datasets and then doing some k-fold cross-validation on the combined dataset. The fact that studying the effects of using XGBoost's ability to calculate feature importance scores and thus act as a good feature selection method for classification problems has not been utilized for studying the effects of it in intrusion detection can be clearly identified as a research gap in this domain. In order to address this, in this research project, XGBoost is used as a feature selection method on the NSL-KDD dataset and the effects of using minimal feature sets created are compared using a LSTM RNN classifier model.

CHAPTER 3

METHODOLOGY

3.1. Overview

In this research, the train and test datasets in the NSL-KDD dataset separately for training and testing respectively. Jupyter Notebook was chosen for the simulation.

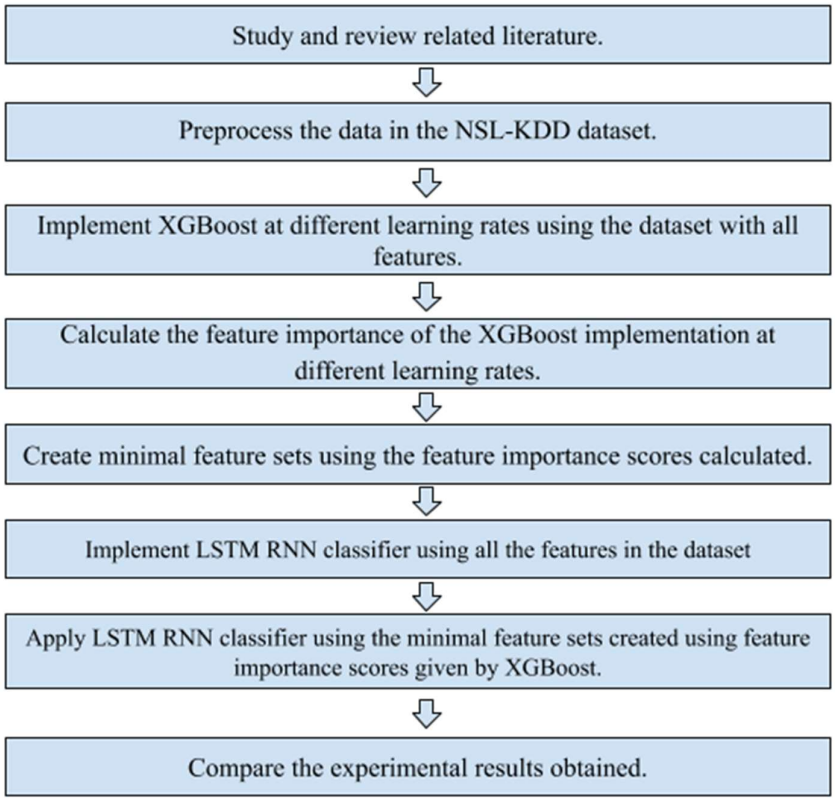

Figure 3.1: Stepwise research procedure.

In an oversimplified overview, first, the data was preprocessed, then fed into XGBoost model. Using XGBoost, feature importance scores were calculated and then minimal feature sets were created. After that, all features in the dataset and the minimal feature sets created were used with LSTM RNN classifier separately. Finally, the results were compared.

3.2 Dataset Description

The KDD Cup ’99 dataset is a standard dataset used for research on intrusion detection systems which was first used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held together with KDD-99, The Fifth International Conference on Knowledge Discovery and Data Mining. The task assigned for the competitors was to build a model capable of distinguishing between connections that belong to attacks or intrusions and normal connections. The database also includes a wide variety of intrusions simulated in a military network environment.

The NSL-KDD dataset used in this research was created in 2009, and is an improvement over the KDD’99 dataset from which many inherent problems like redundant data instances have been removed to avoid the network from giving biased results based on more frequent records. The best version of the KDD Cup ’99 dataset available so far is the NSL-KDD dataset. The dataset contains the KDDTrain+ dataset as the training set and KDDTest+ and KDDTest−21 datasets as the testing set, which has different normal records and four different types of attack records. The KDDTest−21 dataset is a subset of the KDDTest+ and is more difficult for classification. In this research, KDDTrain+ dataset was used as the training set and KDDTest+ dataset was used as the testing set.

KDDTrain+ consists of 125973 records while there is a total number of 22543 instances in the KDDTest+ dataset. There are 42 attributes in each record of which the last attribute is the class or the label of the instance whether it is a normal connection instance or an attack. [21]

Nr	features	
	name	description
1	duration	duration of connection in seconds
2	protocol_type	connection protocol (tcp, udp, icmp)
3	service	dst port mapped to service (e.g., http, ftp, etc.)
4	flag	normal or error status flag of connection
5	src_bytes	number of data bytes from src to dst
6	dst_bytes	bytes from dst to src
7	land	1 if connection is from/to the same host/port; else 0
8	wrong_fragment	number of 'wrong' fragments (values 0,1,3)
9	urgent	number of urgent packets
10	hot	number of 'hot' indicators (bro-ids feature)
11	num_failed_logins	number of failed login attempts
12	logged_in	1 if successfully logged in; else 0
13	num_compromised	number of 'compromised' conditions
14	root_shell	1 if root shell is obtained; else 0
15	su_attempted	1 if 'su root' command attempted; else 0
16	num_root	number of 'root' accesses
17	num_file_creations	number of file creation operations
18	num_shells	number of shell prompts
19	num_access_files	number of operations on access control files
20	num_outbound_cmds	number of outbound commands in an ftp session
21	is_hot_login	1 if login belongs to 'hot' list (e.g., root, adm); else 0
22	is_guest_login	1 if login is 'guest' login (e.g., guest, anonymous); else 0
23	count	number of connections to same host as current connection in past two seconds
24	srv_count	number of connections to same service as current connection in past two seconds
25	serror_rate	% of connections that have 'SYN' errors
26	srv_serror_rate	% of connections that have 'SYN' errors
27	rerror_rate	% of connections that have 'REJ' errors
28	srv_rerror_rate	% of connections that have 'REJ' errors
29	same_srv_rate	% of connections to the same service
30	diff_srv_rate	% of connections to different services
31	srv_diff_host_rate	% of connections to different hosts
32	dst_host_count	count of connections having same dst host
33	dst_host_srv_count	count of connections having same dst host and using same service
34	dst_host_same_srv_rate	% of connections having same dst port and using same service
35	dst_host_diff_srv_rate	% of different services on current host
36	dst_host_same_src_port_rate	% of connections to current host having same src port
37	dst_host_srv_diff_host_rate	% of connections to same service coming from diff. hosts
38	dst_host_serror_rate	% of connections to current host that have an S0 error
39	dst_host_srv_serror_rate	% of connections to current host and specified service that have an S0 error
40	dst_host_rerror_rate	% of connections to current host that have an RST error
41	dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST error

Table 3.1: The 41 features provided by the NSL-KDD dataset. (Source: [13])

The 41 other attributes are related to the network information and can be classified into 4 different classes: Basic features (the attributes of individual TCP connections), Content features (the attributes within a connection suggested by the domain knowledge), Traffic features (the attributes computed using a two-second time window) and Host features (the attributes designed to assess attacks which last for more than two seconds). [21]

Sr. No	Label	Attribute Name	Sr. No	Label	Attribute Name	Sr. No	Label	Attribute Name	Sr. No	Label	Attribute Name
1	B	duration	10	C	hot	23	T	count	32	H	dst_host_count
2	B	protocol_type	11	C	num_failed_logins	24	T	serror_rate	33	H	dst_host_srv_count
3	B	service	12	C	logged_in	25	T	rerror_rate	34	H	dst_host_same_srv_rate
4	B	src_bytes	13	C	num_compromised	26	T	same_srv_rate	35	H	dst_host_diff_srv_rate
5	B	dst_bytes	14	C	root_shell	27	T	diff_srv_rate	36	H	dst_host_same_src_port_rate
6	B	flag	15	C	su_attempted	28	T	srv_count	37	H	dst_host_srv_diff_host_rate
7	B	land	16	C	num_root	29	T	srv_serror_rate	38	H	dst_host_serror_rate
8	B	wrong_fragment	17	C	num_file_creations	30	T	srv_rerror_rate	39	H	dst_host_srv_serror_rate
9	B	urgent	18	C	num_shells	31	T	srv_diff_host_rate	40	H	dst_host_rerror_rate
			19	C	num_access_files				41	H	dst_host_srv_rerror_rate
			20	C	num_outbound_cmds				42	-	class
			21	C	is_hot_login						
			22	C	is_guest_login						

Table 3.2: Classwise details of NSL-KDD dataset attributes. (Source: [21])

3.3 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost), mainly designed for speed and performance, makes use of gradient-boosted decision trees. It is a way of applying boosting to machines and was first done by Tianqi Chen [22]. This tool, which belongs to the Distributed Machine Learning Community (DMLC), helps to exploit every bit of memory and hardware resources for tree boosting algorithms which is beneficial for performance enhancement and model-tuning. There are three main gradient boosting techniques that XGBoost can perform. They are Gradient Boosting, Regularized Boosting, and Stochastic Boosting. XGBoost also stands out from other libraries because it allows for the addition and tuning of regularization parameters. The algorithm is highly effective in reducing the computing time and provides ideal use of memory

assets. It can take care of missing values, supports parallel structure in tree construction, and has the unique ability to perform boosting on added data already on the trained model [23].

The fact that the importance scores for each attribute can be retrieved straight away after the boosted trees are constructed is an advantage of using ensembles of decision tree methods like gradient boosting. Importance gives a score which reflects how useful or valuable each feature was in the building up of the boosted decision trees within the model. Thus, a higher value indicates a higher relative importance on its use for making key decisions. Since the scores are calculated explicitly for each feature in the dataset, the attributes can be ranked accordingly and compared with each other. [24]

3.4 Long Short-Term Memory Recurrent Neural Networks

Recurrent Neural Networks are used when it is important to make use of sequential information. In a traditional neural network, we assume that all inputs and outputs are independent of each other. But there are situations where understanding of the previous outputs is important. Prior studies have shown that considering the utilization of deep neural networks for network intrusion detection, recurrent neural networks give very high accuracies. The reason they are said to be "recurrent" is because they perform the same task for every record of a sequence, with the output for each being dependent on the previous computations as they have a memory which captures information about the calculations done so far. [25]

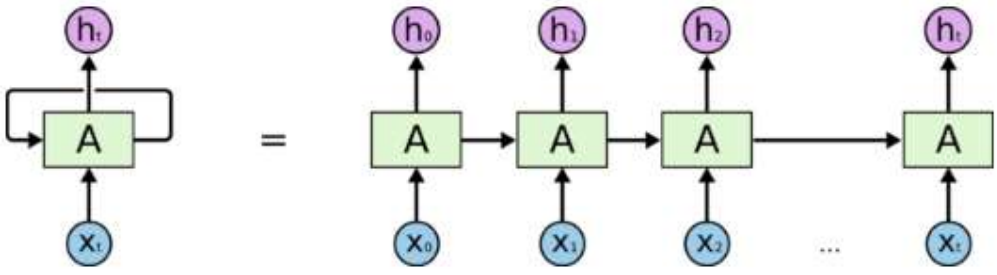

Figure 3.2: An unrolled recurrent neural network. (Source: [25])

However, standard RNN cannot bridge more than 5 to 10 time-steps because error signals tend to either explode or vanish. An exploding error leads to oscillating weights, whereas with a vanishing error, learning takes an unacceptable amount of time, or does not work at

all. As a solution to vanishing gradient problem, LSTM RNNs are used. LSTM is a gradient-based method and can learn how to bridge minimal time lags of more than 1000 discrete time steps. [13]

3.5 Implementation

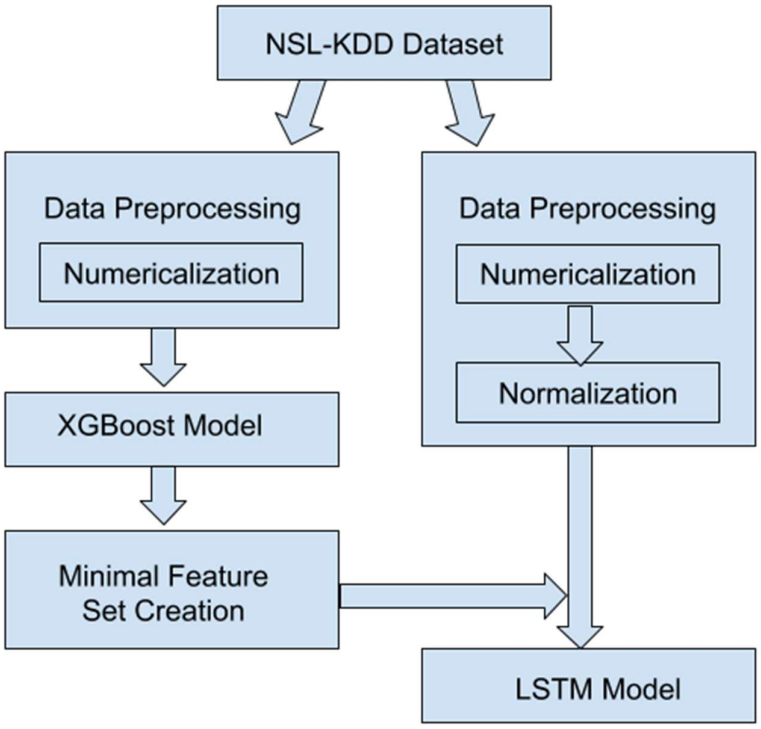

Figure 3.3: Simplified view of the proposed method

3.5.1 Data Preprocessing

Preprocessing of the data was done in two steps: Numericalization and Normalization.

The NSL-KDD dataset consists of records with 38 numeric features and 3 nonnumeric features. Therefore, in numericalization, in order to convert the nonnumeric features: 'protocol type', 'service', and 'flag' into numeric form, they were encoded as binary vectors. For example, the feature 'protocol_type' has three types of attributes, 'tcp', 'udp', and 'icmp'. So, values were assigned as (1,0,0), (0,1,0) and (0,0,1). Similarly, the feature 'service' which has 70 types of attributes, and the feature 'flag' which has 11 types of attributes were encoded and this led to mapping of 41 features into 122 features in total.

Secondly, normalization is important where the difference between the maximum and minimum values in the features has a very large scope. It is often applied as part of data preparation for machine learning, to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. In normalization, each record was sampled individually to unit norm so that those with at least one non-zero component was rescaled independently of other samples so that its norm equaled one.

However, not all models are affected by feature scaling. For example, tree-based algorithms like decision trees and random forests are independent of feature scale. In these algorithms, a node of a tree partitions the data into 2 sets by comparing a feature to a threshold value. Since there's no regularization for the threshold, it's not affected by different scales. Hence, XGBoost, being an ensemble of decision tree methods, is not affected by feature scaling. Due to this, the data used in the XGBoost implementation was the data preprocessed only by numericalization while the data used in the LSTM model implementation was the data preprocessed by both steps: numericalization and normalization.

3.5.2 XGBoost Implementation

The preprocessed data having 122 features in total was fit into a XGBoost model and results were recorded at different learning rates: 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9.

Learning rate	Accuracy (%)	F1 score	Time taken(s)
0.001	79.70	0.8035	37
0.01	77.52	0.78775	33
0.1	79.50	0.8035	32
0.2	79.77	0.8056	32
0.3	79.33	0.8021	32
0.4	79.85	0.8063	32
0.5	79.96	0.80685	31
0.6	79.03	0.79988	31
0.7	79.83	0.8059	31
0.8	81.40	0.81824	35
0.9	78.64	0.79677	30

Table 3.3: Results of XGBoost implementation with all 122 features

Learning rate	True Positives	True Negatives	False Positives	False Negatives
0.001	9356	8611	4222	354
0.01	9405	8070	4763	305
0.1	9453	8469	4364	257
0.2	9450	8533	4300	260
0.3	9447	8437	4396	263
0.4	9454	8547	4286	256
0.5	9435	8591	4242	275
0.6	9449	8366	4467	261
0.7	9445	8550	4283	265
0.8	9438	8912	3921	272
0.9	9439	8289	4544	271

Table 3.4: Results from confusion matrix of XGBoost model implementation at different learning rates

The feature importance scores too were calculated using XGBoost at each learning late and recorded.

Learning rate	Feature numbers in order of importance scores									
0.001	1	56	65	30	6	2	90	89	88	79
0.01	1	56	65	19	32	6	4	2	9	30
0.1	1	56	65	19	32	6	4	2	9	30
0.2	56	1	65	2	19	30	6	32	8	9
0.3	65	56	1	2	19	6	30	32	90	9
0.4	56	65	1	2	19	9	6	32	30	29
0.5	56	1	65	2	19	90	29	30	32	6
0.6	56	65	1	2	6	19	30	90	26	32
0.7	56	65	1	2	6	19	30	32	90	61
0.8	56	65	1	6	2	30	90	19	32	29
0.9	56	65	1	6	30	2	18	90	19	32

Table 3.5: Feature importance calculated using XGBoost at different learning rates

3.5.3 Minimal Feature Set Creation

Considering the results of the XGBoost implementation with all the 122 features in the preprocessed data, based on the accuracy and F1 score values and the feature importance scores at different learning rates, smaller feature sets of 10 features, 8 features, 6 features and 4 features were created. Since the highest accuracy values and f1-score values were shown at 0.8 and 0.5 learning rates, features for minimal feature sets were decided mainly by taking into account the feature importance orders at 0.8 and 0.5.

	columns	importances
56	ecr_i	0.283210
65	http	0.219024
1	src_bytes	0.158658
6	hot	0.052231
2	dst_bytes	0.047035
30	dst_host_same_srv_rate	0.036526
90	private	0.020444
19	count	0.020093
32	dst_host_same_src_port_rate	0.016791
29	dst_host_srv_count	0.015586
60	ftp	0.013392
55	eco_i	0.010377
35	dst_host_srv_serror_rate	0.008398
112	REJ	0.007230
18	is_guest_login	0.006755
7	num_failed_logins	0.005491
20	srv_count	0.005422
36	dst_host_rerror_rate	0.005376
61	ftp_data	0.005360
34	dst_host_serror_rate	0.004899

Table 3.6: Feature importance scores obtained using XGBoost at learning rate 0.8

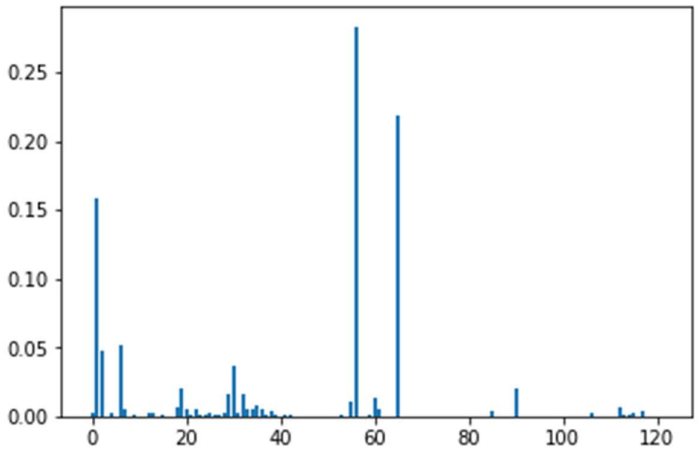

Figure 3.4: Chart showing feature importance scores obtained using XGBoost at learning rate 0.8

	columns	importances
56	ecr_i	0.376655
1	src_bytes	0.144783
65	http	0.140390
2	dst_bytes	0.063815
19	count	0.030076
90	private	0.026545
29	dst_host_srv_count	0.025657
30	dst_host_same_srv_rate	0.024534
32	dst_host_same_src_port_rate	0.023836
6	hot	0.022860
35	dst_host_srv_serror_rate	0.011955
18	is_guest_login	0.011463
38	icmp	0.011293
112	REJ	0.010593
34	dst_host_serror_rate	0.005242
33	dst_host_srv_diff_host_rate	0.004353
39	tcp	0.004278
106	urp_i	0.004198
36	dst_host_rerror_rate	0.003702
59	finger	0.003591

Table 3.7: Feature importance scores calculated using XGBoost at learning rate 0.5

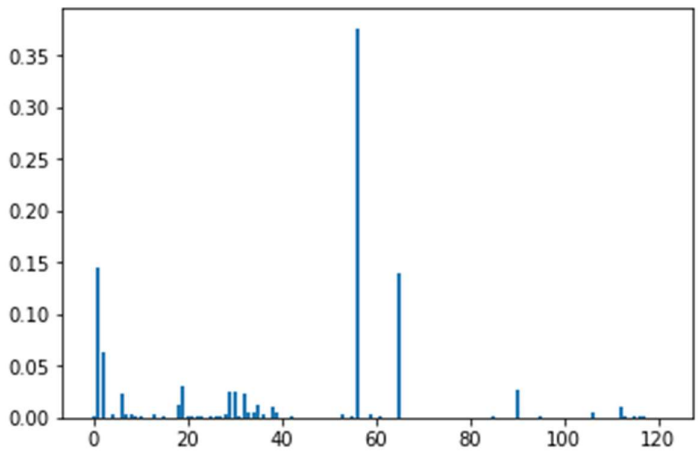

Figure 3.5: Chart showing feature importance scores obtained using XGBoost at learning rate 0.5

Thus, the numbers of the features chosen were as follows.

- 8-feature set: 1, 2, 6, 19, 30, 56, 65, 90
  Feature reduction ratio: 122:8
- 6-feature set: 1, 2, 6, 19, 56, 65
  Feature reduction ratio: 122:6

- 4-feature set: 1, 2, 56, 65
  Feature reduction ratio: 122:4
- 3-feature set: 1, 56, 65
  Feature reduction ratio: 122:3

3.5.4 LSTM RNN Model Implementation with All Features

The preprocessed data having 122 features in total was fit into an LSTM RNN model and results were recorded. The learning rate, the number of hidden nodes and the number of epochs were set to 0.5, 80 and 45 respectively.

3.5.5 LSTM RNN Model Implementation with Minimal Feature Sets

The minimal feature sets were extracted from the preprocessed data and were fit into the LSTM model setting the learning rate to 0.5, the number of hidden nodes to 80 and the number of epochs to 45, and results were recorded.

3.6 Machine Specifications

The specifications of the machine used for running the models are as follows.

Processor – Intel Xeon Gold

RAM – 128 GB

Hard Disk – 3 TB

GPU – P40

GPU RAM – 24 GB

Memory fraction of the GPU utilized – 0.5

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results of LSTM RNN Implementation Without Feature Selection

Feature set	Accuracy	F1 score	Loss	Time taken for training	Time for testing one record (μs)
All 122 features	78.71%	0.7968	1.15	36 mins 45 s	20

Table 4.1: Results of LSTM model using the preprocessed full feature set

Feature Set	True Positives	True Negatives	False Positives	False Negatives
All 122 features	9413	8331	4502	297

Table 4.2: Results from confusion matrix of LSTM model using the preprocessed full feature set

4.2 Results of LSTM RNN Implementation Using Minimal Feature Sets

Feature Set	Accuracy	F1 score	Loss	Time taken for training	Time for testing one record (μs)
8-feature set	73.82%	0.7606	0.86	35 mins 23s	19
6-feature set	75.60 %	0.7715	0.81	36mins 1s	21
4-feature set	76.84%	0.7745	0.82	33mins 48s	22
3-feature set	74.11%	0.7663	0.54	35mins 41s	25

Table 4.3: Results of LSTM model using minimal feature sets created

Feature Set	True Positives	True Negatives	False Positives	False Negatives
8-feature set	9375	7267	5566	335
6-feature set	9287	7755	5078	423
4-feature set	8969	8354	4479	741
3-feature set	9572	7135	5698	138

Table 4.4: Results from confusion matrix of LSTM model using minimal feature sets created

4.3 Results Comparison and Discussion

According to Table 4.2, the minimal feature set created using 4 features has shown the closest accuracy (76.84%) and F1 score (0.7745) to the accuracy (78.71%) and F1 score (0.7968) obtained using all the 122 features in the preprocessed dataset.

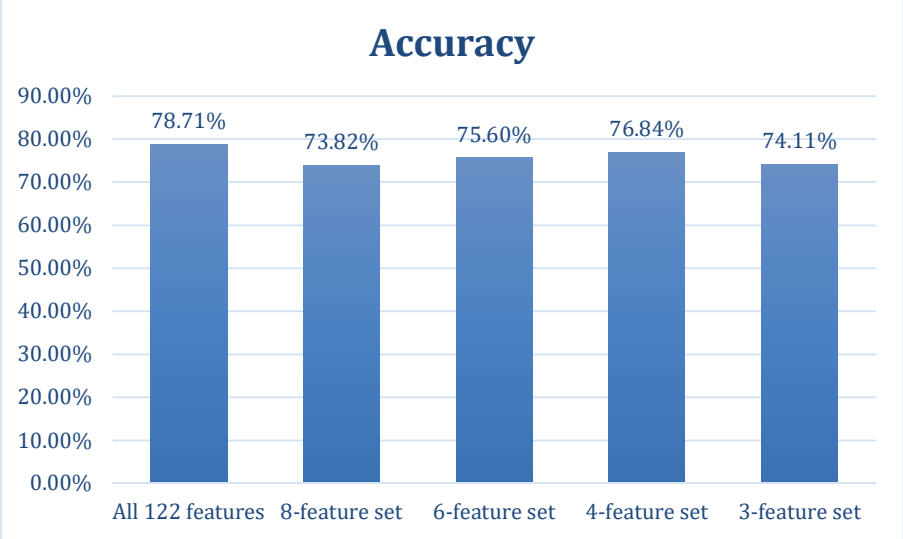

Figure 4.1: Comparison of classification accuracy by models created using different feature sets

The comparison of classification accuracy achieved in each case, as seen in Figure 4.1, shows that all the models created using the minimal feature sets have achieved accuracy values much close to the accuracy value achieved by the model created using all the 122 features.

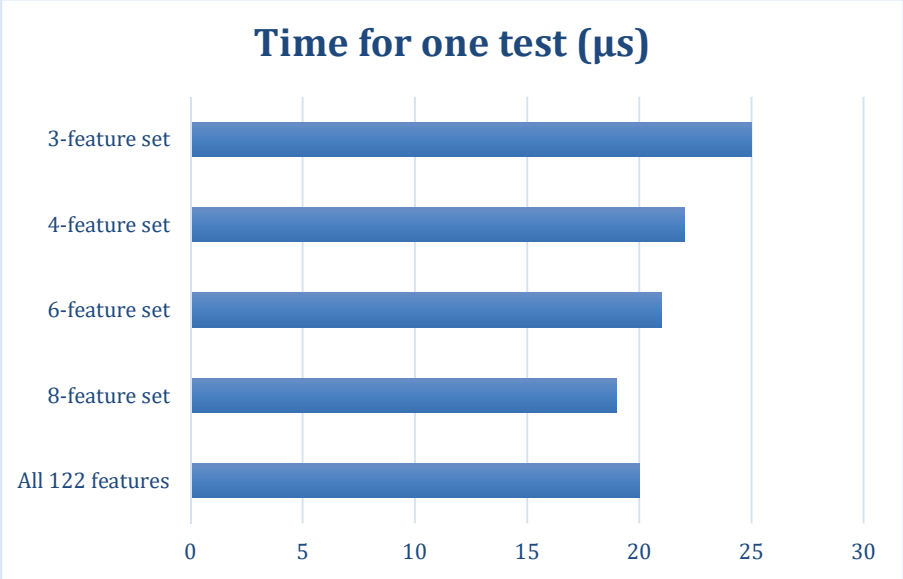

Figure 4.2: Comparison of time taken for a classification by models created using different feature sets

Figure 4.2 shows that according to the experimental results, the values for time taken to perform one classification by the different models show little difference when compared to each other. The reason for this, perhaps, could be the other processes running on the machine in parallel. However, more testing and optimization might give lower testing times for the models trained using minimal feature sets as expected.

Hence, compared to the accuracy obtained by LSTM network trained with all features, it is notable that the values obtained for accuracy by the models trained using minimal feature sets almost match with it and are acceptable. As advantages of using the proposed method, it can be stated that reduction in number of features reduces the amount of processing involved and clearly saves up memory space required given the very high reduction ratio of the number of features.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

Extreme Gradient Boosting (XGBoost) has shown to be very effective for feature selection in network intrusion detection. The classification accuracies achieved by using the minimal feature sets created using XGBoost are very close to that of the original feature set. Using XGBoost for feature selection in designing an intrusion detection system using long short-term memory recurrent neural networks will reduce the complexity of the system whilst achieving an acceptable accuracy and a better performance in terms of memory consumption. This method can also be used in designing a first layer of defense for alerting users about a possible attack, until the complex higher layers complete more accurate classification using more features.

5.2 Future Work

As future work, it can be suggested to perform the same experiments with the KDDTest-21 dataset too and see the effects of using XGBoost feature selection for LSTM model to classify the harder records out of the testing dataset.
It is also possible to do the same experiments for multiclass classification to check the effects of XGBoost feature selection for LSTM model to classify the records into whether normal or which different attack type it belongs to.

In addition, the proposed model has several future scopes:

- Run the model with more combinations of the selected features.
- Optimize the model by tuning parameters further.
- Test with more complex LSTM networks.
- Select more features and test using various combinations of them with the model.
- Perform feature engineering to create new features using the selected features and test the model with them.
- Test the effects of using the minimal feature sets with other classification models.
- Run the model for other datasets.

REFERENCES

[1] Muhammad, K. A., Khan, T. A., Talha, A. T., Umar, N., Yakoob, S., Network Intrusion Detection and its Strategic Importance, IEEE Business Engineering and Industrial Applications Colloquium, 2013.

[2] Kuang, F, Xu, W. and Zhang, S., A novel hybrid KPCA and SVM with GA model for intrusion detection. *Applied Soft Computing*, May 2014, 18, 178–184.

[3] Li, W., Yi, P., Wu, Y., Pan, L. and Li, J., A new intrusion detection system based on KNN classification algorithm in wireless sensor network, Journal of Electrical and Computer Engineering, Jun. 2014.

[4] Ingre, B., and Yadav, A., Performance analysis of NSL-KDD dataset using ANN, International Conference on Signal Processing and Communication Engineering Systems, Jan.2015, 92–96.

[5] Farnaaz, N., and Jabbar, M. A., Random forest modeling for network intrusion detection system, Procedia Computer Science, Jan. 2016, 89, 213–217.

[6] Zhang, J. and Zulkernine, M., A., Network Intrusion Detection using Random Forests, School of Computing, Queen's University, Canada, Sep. 2008.

[7] Khan, J. A. and Jain, N., A survey on intrusion detection systems and classification techniques, International Journal of Scientific Research in Science, Engineering and Technology, 2016, 2(5), 202–208.

[8] Banoth, L., Teja, M. P. S.K., Saicharan, M., and Chandra, N. J., A survey of data mining and machine learning methods for cyber security intrusion detection, International Journal of Research, April 2017, 4(5).

[9] Javaid, A., Niyaz, Q., Sun, W. and Alam, M., A deep learning approach for network intrusion detection system, 9th EAI International Conference on Bio-inspired Information and Communications Technologies, NY, USA, May 2016, 21–26.

[10] Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R. and Ghogho, M., Deep Learning Approach for Network Intrusion Detection in Software Defined Networking, International Conference on Wireless Networks and Mobile Communications (WINCOM), Oct. 2016, 258–263.

[11] Sheikhan, M., Jadidi, Z. and Farrokhi, A., Intrusion detection using reduced-size RNN based on feature grouping, Neural Computing and Applications, Sep.2012, 21(6), 1185–1190.

[12] Yin, C., Zhu, Y., Fei, J. and He, X., A Deep Learning Approach for Intrusion Detection using Recurrent Neural Networks, IEEE Access, Oct. 2017.

[13] Staudemeyer, R. C., The importance of time: Modelling network intrusions with long short-term memory recurrent neural networks. Ph. D. Thesis, University of Western Cape, South Africa, May 2012.

[14] Zygmunt, Z. What is Better: Gradient-Boosted Trees, or a Random Forest?—FastML. 2016. [online] Available at: http://fastml.com/what-is-better-gradient-boosted-trees-or-random-forest/ (Accessed 10 May 2019).

[15] Cross Validated, Gradient Boosting Trees vs Random Forest. Availableonline: https://stats.stackexchange.com/questions/173390/gradient-boosting-tree-vs-random-forest (Accessed 10 May 2019).

[16] S. S. Dhaliwal, A. Nahid and R. Abbas, "Effective Intrusion Detection System Using XGBoost", Information Journal, June 2018.

[17] Al-Qatf, M., Alhabib, M., Al-Sabahi, K. and others. Deep Learning Approach Combining Sparse Autoen-coder with SVM for Network Intrusion Detection. IEEE Access 2018, 6, 52843–52856.

[18] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A. Nsl-Kdd Dataset, 2012, [online] Available at: https://www.unb.ca/cic/datasets/nsl.html (Accessed 28 February 2019).

[19] Vasan, K.K., Surendiran, B., Dimensionality reduction using Principal Component Analysis for network intrusion detection. Perspectives in Science, 2016, 8, 510–512.

[20] Abdulhammed, R., Musafer, H., Alessa, A., Faezipour, M. and Abuzneid, A., Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection, Electronics, March 2019, 8, 322.

[21] Aggarwal, P., and Sharma, S. K., "Analysis of KDD Dataset Attributes - Class wise For Intrusion Detection," 3rd International Conference on Recent Trends in Computing 2015, [online] Available at: https://www.sciencedirect.com/science/article/pii/S1877050915020190/pdf [Accessed 12 April 2019]

[22] Brownlee, J., A Gentle Introduction to XGBoost for Applied Machine Learning. Machine Learning Mastery. [online] Available at: http://machinelearningmastery.com/gentle-introduction-xgboost-appliedmachine-learning/ [Accessed 2 July 2019].

[23] Reinstein, I. XGBoost a Top Machine Learning Method on Kaggle, Explained. [online] Available at: http: //www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html [Accessed 2 July 2019].

[23b] Brownlee, J., Feature Importance and Feature Selection with XGBoost in Python. Machine Learning Mastery. [online] Available at: https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/ [Accessed 23 August 2019]

[24] Gudikandula, P., Recurrent Neural Networks and LSTM explained. [online] Available at: https://medium.com/@purnasaigudikandula/recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9 [Accessed 8 July 2019]

APPENDIX

	columns	importances
1	src_bytes	0.640106
56	ecr_i	0.134497
65	http	0.100245
30	dst_host_same_srv_rate	0.048948
6	hot	0.038426
2	dst_bytes	0.037778
90	private	0.000000
89	printer	0.000000
88	pop_3	0.000000
79	netbios_ns	0.000000
87	pop_2	0.000000
86	pm_dump	0.000000
85	other	0.000000
84	ntp_u	0.000000
83	nntp	0.000000
82	nnsp	0.000000
81	netstat	0.000000
80	netbios_ssn	0.000000
76	mtp	0.000000
78	netbios_dgm	0.000000

Feature importance scores obtained using XGBoost at learning rate 0.001

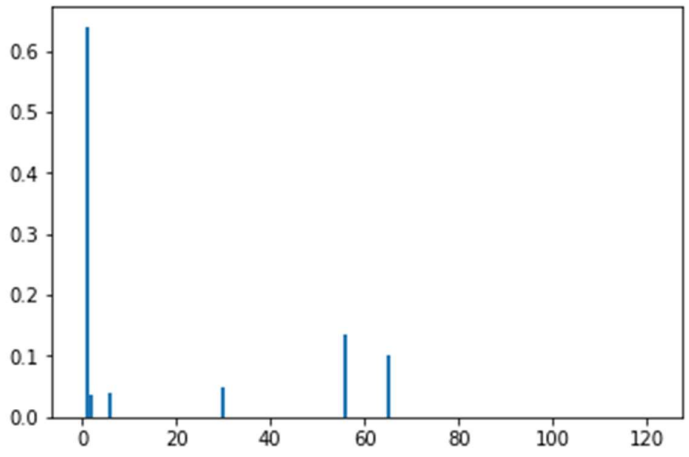

Chart showing feature importance scores obtained using XGBoost at learning rate 0.001

	columns	importances
1	src_bytes	0.458220
56	ecr_i	0.107478
65	http	0.093166
19	count	0.051088
32	dst_host_same_src_port_rate	0.049372
6	hot	0.040510
4	wrong_fragment	0.038893
2	dst_bytes	0.038536
9	num_compromised	0.036466
30	dst_host_same_srv_rate	0.035048
8	logged_in	0.028978
35	dst_host_srv_serror_rate	0.020153
106	urp_i	0.002092
89	printer	0.000000
81	netstat	0.000000
88	pop_3	0.000000
90	private	0.000000
87	pop_2	0.000000
86	pm_dump	0.000000
85	other	0.000000

Feature importance scores calculated using XGBoost at learning rate 0.01

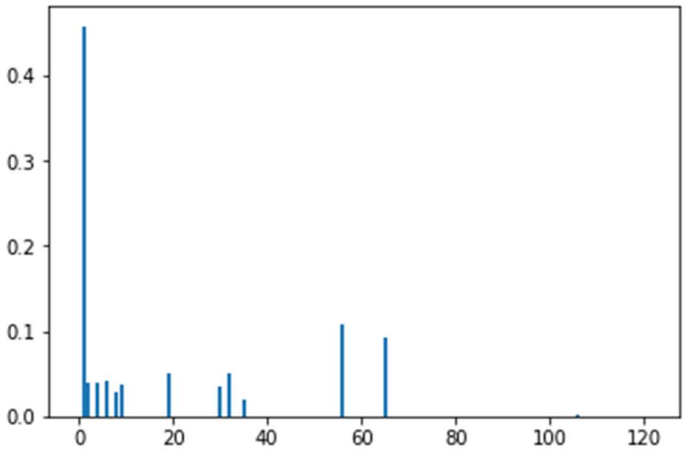

Chart showing feature importance scores obtained using XGBoost at learning rate 0.01

	columns	importances
1	src_bytes	0.231085
56	ecr_i	0.198816
65	http	0.112550
2	dst_bytes	0.089018
19	count	0.041349
30	dst_host_same_srv_rate	0.033058
26	diff_srv_rate	0.032462
6	hot	0.024856
32	dst_host_same_src_port_rate	0.024822
29	dst_host_srv_count	0.022429
9	num_compromised	0.015869
90	private	0.015643
20	srv_count	0.015227
35	dst_host_srv_serror_rate	0.015201
8	logged_in	0.012818
0	duration	0.012330
39	tcp	0.010986
38	icmp	0.009445
95	smtp	0.008043
61	ftp_data	0.007127

Feature importance scores calculated using XGBoost at learning rate 0.1

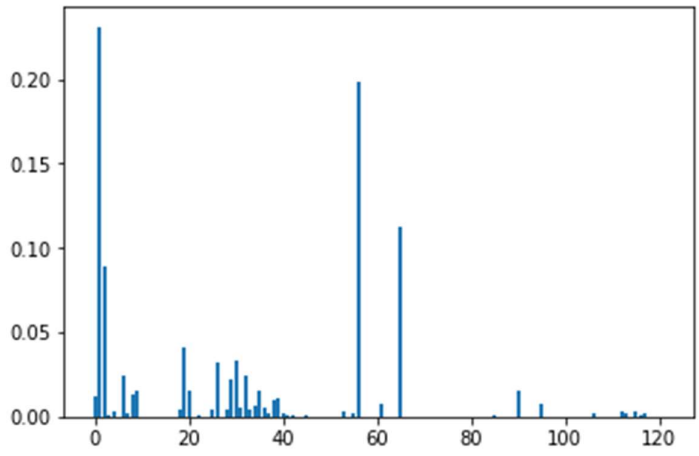

Chart showing feature importance scores obtained using XGBoost at learning rate 0.1

	columns	importances
56	ecr_i	0.269441
1	src_bytes	0.181318
65	http	0.137696
2	dst_bytes	0.078464
19	count	0.041860
30	dst_host_same_srv_rate	0.030983
6	hot	0.024454
32	dst_host_same_src_port_rate	0.022381
8	logged_in	0.019428
9	num_compromised	0.018443
29	dst_host_srv_count	0.017820
90	private	0.017299
35	dst_host_srv_serror_rate	0.015407
61	ftp_data	0.010342
18	is_guest_login	0.008544
26	diff_srv_rate	0.007872
95	smtp	0.006459
39	tcp	0.006294
34	dst_host_serror_rate	0.006252
38	icmp	0.006181

Feature importance scores calculated using XGBoost at learning rate 0.2

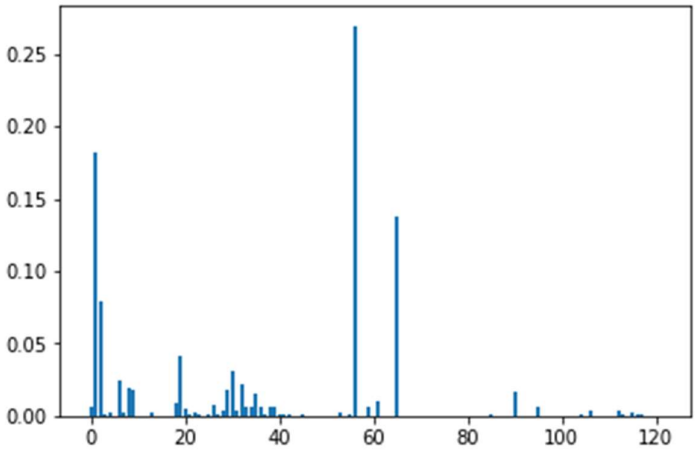

Chart showing feature importance scores obtained using XGBoost at learning rate 0.2

	columns	importances
65	http	0.235086
56	ecr_i	0.223971
1	src_bytes	0.161011
2	dst_bytes	0.052577
19	count	0.046449
6	hot	0.029241
30	dst_host_same_srv_rate	0.028716
32	dst_host_same_src_port_rate	0.027174
90	private	0.021470
9	num_compromised	0.020203
35	dst_host_srv_serror_rate	0.017336
29	dst_host_srv_count	0.013400
95	smtp	0.013214
61	ftp_data	0.008596
38	icmp	0.008291
8	logged_in	0.007042
36	dst_host_rerror_rate	0.006329
33	dst_host_srv_diff_host_rate	0.005982
20	srv_count	0.005891
39	tcp	0.005559

Feature importance scores calculated using XGBoost at learning rate 0.3

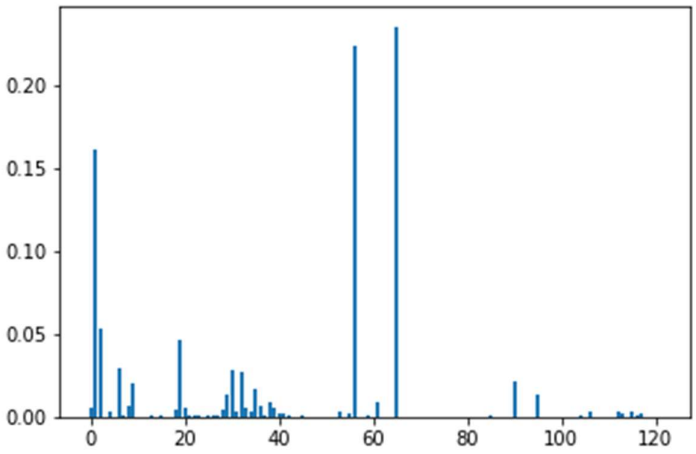

Chart showing feature importance scores obtained using XGBoost at learning rate 0.3

	columns	importances
56	ecr_i	0.307702
65	http	0.162693
1	src_bytes	0.151876
2	dst_bytes	0.054132
19	count	0.035241
9	num_compromised	0.032452
6	hot	0.028879
32	dst_host_same_src_port_rate	0.027909
30	dst_host_same_srv_rate	0.025813
29	dst_host_srv_count	0.023063
90	private	0.015591
35	dst_host_srv_serror_rate	0.014781
60	ftp	0.012482
8	logged_in	0.012067
38	icmp	0.010341
22	srv_serror_rate	0.007665
18	is_guest_login	0.007617
33	dst_host_srv_diff_host_rate	0.005113
34	dst_host_serror_rate	0.004652
0	duration	0.004645

Feature importance scores calculated using XGBoost at learning rate 0.4

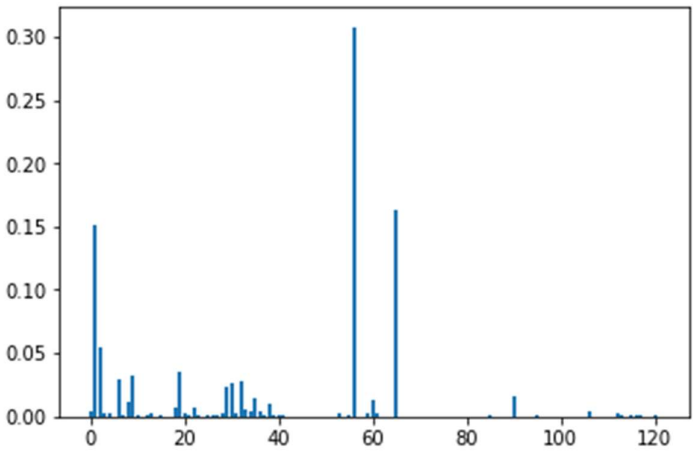

Chart showing feature importance scores obtained using XGBoost at learning rate 0.4

	columns	importances
56	ecr_i	0.281871
65	http	0.273361
1	src_bytes	0.132675
2	dst_bytes	0.057333
6	hot	0.029305
19	count	0.027603
30	dst_host_same_srv_rate	0.023810
90	private	0.020807
26	diff_srv_rate	0.020105
32	dst_host_same_src_port_rate	0.018784
38	icmp	0.015292
29	dst_host_srv_count	0.010532
106	urp_i	0.006443
35	dst_host_srv_serror_rate	0.004875
60	ftp	0.004771
36	dst_host_rerror_rate	0.004461
18	is_guest_login	0.004361
22	srv_serror_rate	0.004218
112	REJ	0.003648
31	dst_host_diff_srv_rate	0.003624

Feature importance scores calculated using XGBoost at learning rate 0.6

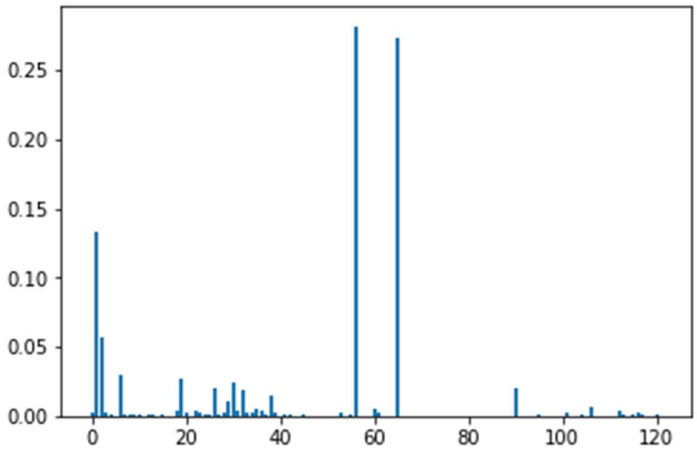

Chart showing feature importance scores obtained using XGBoost at learning rate 0.6

	columns	importances
56	ecr_i	0.309210
65	http	0.244861
1	src_bytes	0.150566
2	dst_bytes	0.041412
6	hot	0.035183
19	count	0.031663
30	dst_host_same_srv_rate	0.023259
32	dst_host_same_src_port_rate	0.019675
90	private	0.015629
61	ftp_data	0.012246
35	dst_host_srv_serror_rate	0.010592
38	icmp	0.010450
95	smtp	0.009906
29	dst_host_srv_count	0.008987
33	dst_host_srv_diff_host_rate	0.007438
112	REJ	0.005812
34	dst_host_serror_rate	0.005001
18	is_guest_login	0.004374
0	duration	0.003597
4	wrong_fragment	0.003570

Feature importance scores calculated using XGBoost at learning rate 0.7

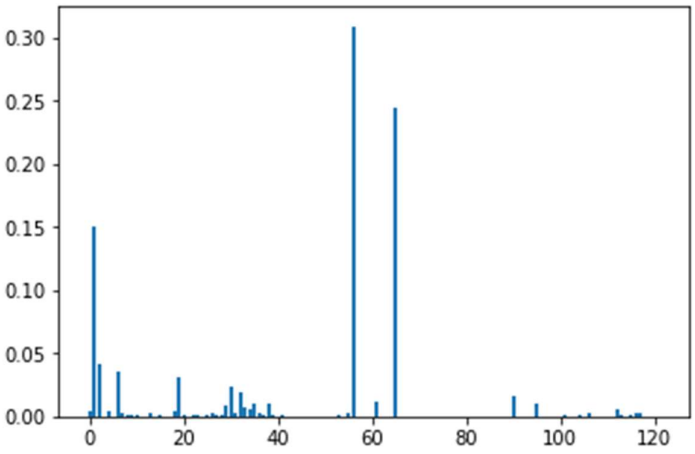

Chart showing feature importance scores obtained using XGBoost at learning rate 0.7