Design Approaches of Intrusion Detection Systems using Ensembling Algorithms

MSc Research Project
Cloud Computing

Saurabh Kulkarni
x16104307

School of Computing
National College of Ireland

Supervisor: Dr.Horacio Gonzalez-Velez

National College of Ireland
Project Submission Sheet – 2016/2017
School of Computing

Student Name:	Saurabh Kulkarni
Student ID:	x16104307
Programme:	Cloud Computing
Year:	2017
Module:	MSc Research Project
Lecturer:	Dr.Horacio Gonzalez-Velez
Submission Due Date:	16/08/2017
Project Title:	Design Approaches of Intrusion Detection Systems using Ensembling Algorithms
Word Count:	XXX

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	13th September 2017

PLEASE READ THE FOLLOWING INSTRUCTIONS:
1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Design Approaches of Intrusion Detection Systems using Ensembling Algorithms

Saurabh Kulkarni
x16104307
MSc Research Project in Cloud Computing

13th September 2017

Abstract

Intrusion Detection Systems are very important when it comes to monitoring network traffic, so fast and efficient analysis of these malicious network attacks can be a challenging task especially dealing with sophisticated cyberattacks with large amount of network traffic flowing from one host to another. So proper validation and classification of these intrusions is very important. Many machine learning algorithms are present that can be used in classification of these intrusions but not all of them are good enough, every algorithm has their own limitations and many tools are incapable of handling such large chunks of data. This research is focused on dealing with intrusion attacks by using modern machine learning Ensembling approaches. The study is divided into three approaches first one involves using clustering algorithms, second one is focused on detecting each attack individually and the third approach consists of Ensembling these approaches and compare the results. On top of that, our classifier has been tested using Apache Sparks machine learning libraries with PySpark. All the experiments are carried on NSL-KDD data set which consists of many network intrusions. With our approach, we managed to get accuracy of around 92% and detection rate of 99%.

1 Introduction

Cloud Security has been an important concern when it comes to management of large cloud services provided by cloud service providers, one of the major concern can be intrusion detection, identifying intrusions among large network traffic flow is very difficult task since massive amount of data is passed from one host to another, this adds complexity in dealing with cyber-attacks. Therefore, it is very important for cloud providers to provide proper implementation of Intrusion Detection Systems (IDS). According to NIST intrusion detection can be termed as monitoring and analyzing the possible events occurring in the computer hardware and software for the signs of possible incidents (Scarfone and Mell; 2010). Traditional intrusion detection methodologies can be categorized in two forms first one is signature based intrusions and another one is anomaly based detection.

Signature based intrusion are based on pattern matching, it corresponds to a known patterns or the type it resembles, in this type the intrusions are identified by comparing the signatures against the obscure events to determine the possible attack. On the other hand, anomaly based detection are used to identify unknown attacks it can be done by setting profiles of the behaviour that are normal against the observed events to detect deviations.

Many researchers have come across different methodologies to analyze the meaningful patterns by making the use of machine learning approaches which include classification and regression in which input data is extracted based on certain parameters in a network traffic and are carried forward to detect attacks. Machine learning techniques can be based on supervised or unsupervised learning which can be performed on labelled or unlabelled data. Many machine learning algorithms can be used for data mining which can contribute significantly in intrusion detection. It has been helpful to extract malicious data from the network, but sometimes these tools are too slow and not efficient enough to handle large workloads which can led to less accurate results. So, to fill the gap between this approach, ensemble approaches need to be followed and certain abilities of these algorithms can lead to better prediction.

Apache Spark is a distribution engine for distributed computing on a cluster. Apache Spark is very fast big data processing tool and is the enhanced version of Apache Hadoop, it uses Resilient Distributed Datasets (RDD) which allows applications to store data in-memory (Zaharia et al.; 2012). Spark provides its own machine learning libraries called MLib which provide access to various machine learning algorithms.

So, based on our background study our research project addresses the question of "*Can Ensembling of different clustering algorithms improve the detection rate of intrusion detection systems processed over Apache Spark?*"

To accomplish the aforesaid aim the research study comprised with following objectives:

1. To train our classifier to achieve better prediction accuracy
2. To impose various clustering algorithms on our classifier
3. To compare K-means and Gaussian Mixture and check which is better
4. Apply Ensembling approaches to combine the obtained results and check whether it improves the intrusion detection rate of our trained classifier
5. Compare the results

1.1 Paper Overview

This paper comprises of six sections, Section 1 contains the related information and identified objectives about this project, Section 2 consists the background study related to our topic and also identifies the research gaps among them, Section 3 consists of the methodology which explains our research approach, Section 4 comprise of design approaches used in this project, Section 5 evaluates the comparative study based on results obtained and Section 6 concludes the paper with future work.

2 Related Work

Intrusion Detection is considered as security management system and can be used in exposure of vulnerabilities and assessment of malicious network activities within a network. Huang et al. (1999) presented a novel framework based on attack strategy analysis on a large scale distributed intrusion detection. Deng et al. (2017) has said that it is very important to control and prevent intrusions in accurate and timely manner with proper detection and identification methods. Today cyber world is filled with vulnerabilities and malicious users that can be exploited easily, there are many examples of such kind of attacks like WannaCry Ransomware or Petya. Enache et al. (2017) suggest that as a pool of data is gathered through various monitored devices there are chances that the noise inside the data can negatively influence intrusion detection.So, to overcome these challenges, they made a comparative study based on Feature selection methods wrappers using swarm intelligence algorithms but their results were not satisfying enough as they could not get the desired outcomes.

2.1 Machine Learning Approach:

Machine learning is widely used in classification of intrusions in IDS, many researchers have conducted various experiments that tend to use machine learning algorithms to analyze the patterns of attacks and then classify them as normal or malicious, similar approach was followed by (Aygn and Yavuz; 2017) where they compared the performance of their anomaly detection model which was based on deterministic auto-encoder where they used stochastic approach to discriminate normal and abnormal data, but with their approach they could not improve the accuracy and eventually managed to get 88.28% accuracy which was similar to existing machine learning algorithms. Fast processing of large scale network traffic data, to prevent intrusions is still a real challenge of real time intrusion detection systems. Kulariya et al. (2016) did a comparison of various machine learning algorithms such as Logistic Regression, Support vector machine, Random forest, Nave Bayes and Gradient Boosted decision tree. To process the data, they also used Apache Spark a big data processing tool, they tested the results based on Accuracy, Sensitivity, Specificity, Training time and Prediction time, in their test they found out that overall Random Forest performed better and Nave Bayes was the worst when compared to other machine learning algorithms also it performed worse in terms of Specificity which is also called as True negative rate i.e. it measures the proportion of negative correctly identified.

Similarly Hsieh and Chan (2016) have carried a new way of DDoS detection systems integrated with neural networks where they tested their system on Apache Spark in a distributed mode, they also report that DDoS attacks seems to carry high volume of network traffic, report suggest that in 2015 there were 8 attacks that exceeded 100Gbps which made them very hard to detect. In their experiment they collected packets in the form of Tcpdump and stored them in .pcap file which will be stored in HDFS, after that they the data was trained and partitioned based on source and destination IP addresses and then based on the results neural network was created for prediction of malicious data, with their approach they managed to get 94% accuracy, but since they predefined the features in the packets which can be a limitation since packets can have unexpected features which can led to incorrect results and decrease performance in terms of accuracy.

To achieve further advancements in malicious IDS researchers are developing new algorithmic approaches that can eradicate further enhancements. Pallaprolu et al. (2016) propose a semi-supervised approach with the use of ensemble based label propagation for detection of Remote Access Trojans(RAT) packets in unlabelled data. In their experiment, they compare their approach with the traditional signature based intrusion detection systems. They characterized the data that propagates labels and then classify them based on similarity of features i.e. they extract the unstructured data, then train them and apply label propagation. Based on the above approach comparison is made among two datasets and performance is calculated on basis of that. Their study also shows that in-memory computing is more efficient when compared to MapReduce. Apache Spark is enhanced version of Hadoops MapReduce and it is very efficient tool to perform data analysis due to its wide support of data analysis algorithms and libraries. Kumari et al. (2016) made the use Spark to analyse anomaly detection by the means of K-mean clustering, where they explored the possible outcomes regarding how clustering can be used to solve the malicious network flow data to be identified in a distributed environment, in their experimentation they found that Sparks machine learning libraries like MLib can be efficient tool in terms of getting results quickly and they can be easy to use.

Intrusion Detection can be signature based or anomaly based, IDS are considered as crucial factor when it comes to the security of virtual machines since they can be added and removed dynamically. Therefore, it is important that IDS monitor the physical as well as virtual network traffic. How to make use of algorithms in detecting intrusion is a vast research area Zhang et al. (2017) make use Immune clone algorithm in network intrusion detection where, they compared the immune system to computers immune system and based on adhoc features they determined the results. Bjerkestrand et al. (2015) also did an evaluation on feature selection and reduction algorithms where they came to know that different algorithms chose different attributes and they also trained the dataset to select only those attributes that are selected by FSAs to test the accuracy, by their evaluation they manage to get in faster decision and less of storage. Shah et al. (2016) presents a similar approach having used the SSPLR (Structural Sparse Logistic regression) technique to eliminate the unwanted or irrelevant features in this experiment they have proposed two techniques first one is to select individual features with SSPLR and then compare them with different classifiers by their study we came to know that by removing irrelevant features we can significantly improve the intrusion detection accuracy.

2.2 Research Scope

Following are the research scope that can be derived from the background study:

1. To Handle Large Scale Network Intrusions: Most of the application generate large volume of data that makes it prone to intruders so, proper handling of such malicious attacks needs to be done.
2. Accurate Attack Detection: Network attacks can have many forms but it is very necessary to analyze these attacks and identify them as correctly as possible.
3. Fast Detection: Since there are many systems which are interconnected they can generate heavy network traffic, recent advancements in infrastructure has enabled the systems to receive substantial number of incoming packets so, it is necessary to have framework for faster detection of these intrusions.
4. Scalability: Intrusion Detection System should be scalable enough to monitor heavy and dynamic load.

2.3 Summary of Related Work

Study	**Purpose**	**Main Findings**
Huang et al. (1999)	To gain insights on how IDS works	Novel Framework based on attack strategy analysis on large scale distributed intrusion detection
Tavallaee et al. (2009)	Information about KDD99 data set	All the necessary information related to data set like number of records, types of attacks and features were obtained
Pallaprolu et al. (2016)	Ensembling Methods	Semi-supervised approach with the use of ensemble based label propagation for detection of RAT packets in unlabeled data.
Bjerkestrand et al. (2015)	Feature Selection Algorithm	Evaluation on different feature selection and reduction algorithms to train and extract relevant features from data set
Kumari et al. (2016)	K-means Anomaly Detection	Made use of Spark MLib to efficient analyze anomalies using K-means clustering
Kulariya et al. (2016)	Machine Learning Algorithm	Made use of various machine learning algorithm testing accuracy, sensitivity, training time and prediction time
Zaharia et al. (2012)	Apache Spark Architecture	Functioning of Spark, working with RDD's

Table 1: Literature Review Summary

3 Methodology

Following figure depicts the various approaches which are going to be carried out to test our IDS classifier

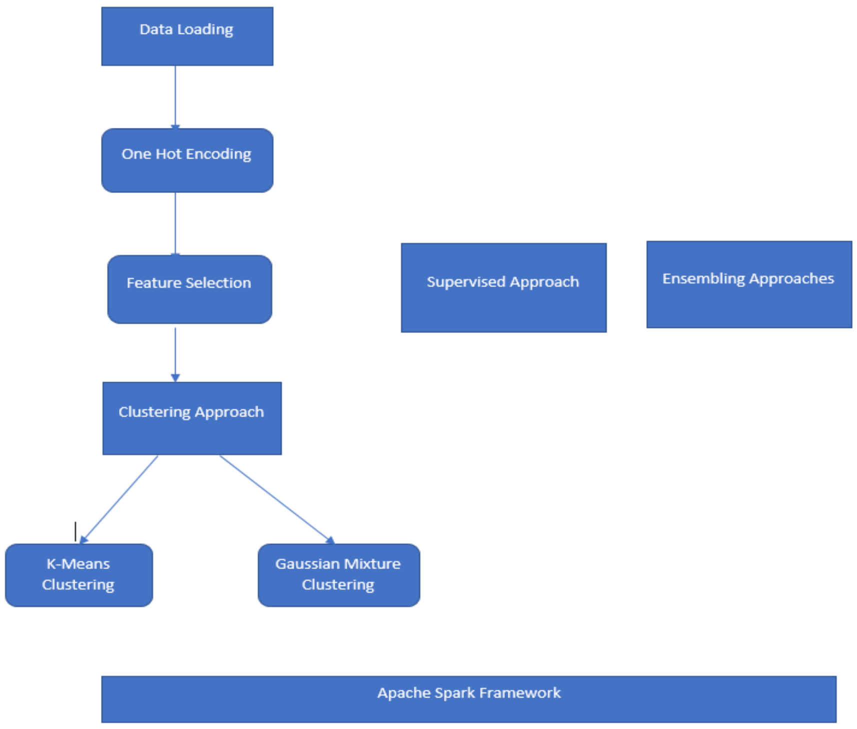

Figure 1: Design Approaches

4 Implementation

Data Loading:

This process will involve taking a dataset in our case the NSL-KDD[1] data set and load it for further processing. For processing the data set we are going to use PySpark with scikit and Mlib as other additional libraries. The dataset is divided into two parts train and test with 80% and 20% respectively.

One Hot Encoding:

One Hot Encoding is a categorical representation of data which is represented in the form of binary vectors. It can be the process of mapping the categorical values to the integer values, where each integer value is marked as a binary vector. We have used this technique just to make our classifier to be more expressive, since many machine learning algorithms are not good at dealing with categorical data directly.

[1]https://github.com/defcom17/NSL-KDD

Feature Selection:

Feature selection is very important when it comes to improving the efficiency of machine learning algorithms, as most of data may be redundant or irrelevant so, it is very crucial to identify which data to work upon. Feature selection algorithm can be used to remove these redundant or irrelevant features from the data and process only those features which are identified as relevant. The NSL-KDD dataset contains three types of features: Numeric, Nominal and Binary features, upon which nominal features are represented with 2, 3 and 4 while binary features are represented with 7, 12, 14,15, 21, 22 respectively and rest of the features are of numeric type.

Type	**Feature**
Nominal Feature	Protocol_type, Service Flag
Binary Features	Land, logged_in, root_shell, su_attempted, is_host_login, is_guest_login
Numeric Features	Duration, src_bytes, dst_bytes, wrong_fragment, urgent, hot, num_failed_logins, num_comprimised, num_root, num_file_creations, num_shells, num_access_files, num_outbounds_cmds, count, srv_count, serror_rate, srv_serror_rate, same_srv_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate. Dst_host_srv_rerror_rate

Table 2: List of Features in NSL-KDD

There are various feature selection algorithm available such Correlation based feature selection (CFS) which is based on feature classification and from that finding the relevant feature correlation, another one is Information Gain (IR) based features selection in which feature evaluation is done on basis of information gained with respect to the appropriate classes. Subset or alternative to information gain ratio is Gain Ratio (GR) which is designed to overcome the difficulties in Information Gain by selecting the features having substantial number of values.

Based on our study we are going to use Feature selection based on Attribute Ratio (AR) as proposed by (Sang-Hyun and Hee-Su; 2014). The NSL-KDD dataset which has three classes i.e. Nominal, Binary and Numeric which will be used to take the average of attribute and frequency for each class and then calculate the attribute ratio from numeric and binary type. Attribute Ratio is calculated as follows

$$AR(i) = MAX(CR(j)) \tag{1}$$

In which Class Ratio (CR) is an attribute, which is taken as the ratio of each class for the Attribute i. Class Ratio is calculated for both numeric and binary as follows

$$CR(j) = \frac{AVG(C(j))}{AVG(Total)} \tag{2}$$

Clustering Approaches:

Clustering can be stated as the task in which a set of objects having similar set of attributes can be grouped together and form a cluster with those objects. Clusters can be classified as Hard and Soft Clusters.

(i) *Hard Clusters*: Hard Clustering refers to that type of clustering in which every data point either completely is a part of cluster or not.

(ii) *Soft Clusters*: In Soft clustering, the data points are assigned based on the probability of them existing in that cluster

In this research, we are going to use K-means and Gaussian Mixture Clustering and then train them using Random Forest Classifiers

- **Random Forest Classifier**: This approach implements random forest classifiers on both sets of clustering i.e. K-means and Gaussian Mixture Clustering, since based on background history this algorithm is highly effective and computationally efficient (McElwee; 2017).
- **K-means Clustering**: K-means is an iterative and unsupervised clustering algorithm in which data is clustered into k number of clusters having a fixed priority, the process involves defining the k centroids for each cluster the placement of these centroids should be done carefully as different results can be obtained based on various locations. The algorithm is composed of following steps:

  (i) Specify the desired number of clusters

  (ii) Assigning the object to every cluster randomly

  (iii) Then compute cluster centroids

  (iv) When all objects are assigned then re-compute the assigned K centroids

  (v) Repeat the steps 3 and 4 until centroids no longer can improve, this will lead to separation of data points into groups from which we can calculate the minimized metric.

- **Gaussian Mixture Clustering**: Gaussian Mixture is model based on probability that works on assumption that all the data objects are result of the mixture of finite number of Gaussian distribution usually having unknown parameters. It is also continuous model based approach, where every cluster is represented as a Gaussian and the entire data set can be combination of mixture and component distribution.

Supervised Approach:

Since we used K-means and Gaussian Mixture for unlabelled data and trained random forest classifier on them, the next approach involves training random forest classifier on labelled data. In this approach, we are going to train our classifier on each of the four major attack categories i.e. DoS, Probe, R2l and U2L attacks. After classification of these attacks we are going to implement feature selection algorithm based on attribute ratio.

Ensembling Approach:

This approach involves combining all the trained classifiers and compare them and find out which classifier performs better. Ensembling techniques consists of bagging, boosting and stacking which contributes significantly in merging of underlying classifiers. Ensemble approaches are known for efficient performance compared to single classifiers.

Bagging involves building multiple data models of same type from different subsamples of data, while boosting involves building multiple but same type of data models which can be trained for better prediction of errors and stacking consists of building multiple data models mainly of diverse types and train them so that they can learn to combine the best predictions of data models.

4.1 Big Data Tools and Libraries:

In this project, we have used Apache Spark which is an in-memory bigdata processing tool, it is also considered as the extended version of Apache Hadoop. Spark makes use of Resilient Distributed Datasets (RDD) to perform in memory operations in fault tolerant manner (Zaharia et al.; 2012), to add further Spark makes use of transformations and actions with operations like (map, filter, join) to be performed on data. To do data analysis on the dataset we have used PySpark which is a python version of Apache Spark and to perform machine learning we have use Mlib and scikit, which are machine learning libraries available open source.

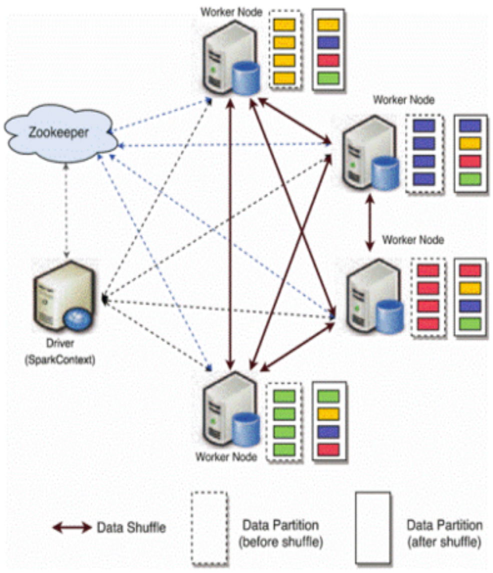

Figure 2: Apache Spark Architecture

4.2 Overall Process

To conduct our experiments, we are going to use NSL-KDD dataset as our test dataset, we are going to load the dataset and apply various approaches and then gather the result of those approaches and compare them with one another.

The first step involves loading of dataset, after loading the data we are going to split the data in two parts, one part will be test data and second part will be used for training the classifier. As NSL-KDD contains several types of attacks but all of them are unlabelled so it is very important to label them to analyse them in future. For that purpose, the connections are divided into classes i.e. normal and attack we have categorized these attacks into the types of attacks they specify then all the attacks are grouped under four categories DDoS, Probe, R2L, U2R.

After training the data we will apply One Hot Encoding (OHE) to the dataset just to make the categorical representation to be more expressive, it is also necessary because many machine learning algorithms cannot work properly with categorical data directly. Then we are going to apply feature selection using attribute ratio. Sang-Hyun and Hee-Su (2014) has proposed using attribute ratio for large scale data, based on their study they found out that other feature selection algorithms like Correlation based, Information based or Gain based are inefficient for large scale data. After this the data is splitted into train and cross validation sets making it 80% and 20% respectively.

The first approach involves the clustering of data, this will be carried out by using K-Means and Gaussian mixture clustering. In K-Means clustering we plan to make clusters of data and then train Random Forest classifier for every clustered data. The next idea is to cluster data using Gaussian Mixture clustering and train them using Random Forest Classifiers, these two classifiers can be combined to improve performance.

The second approach involves using Supervised approach for detecting all types of attack individually here we will apply Random Forest Classifier for four of the attack categories i.e. DDoS, Probe, R2L, U2R after that all the obtained results will combined and then we can test how many intrusions have been classified as an attack.

The third approach involves Ensembling and Stacking all the results obtained from previous approaches, this process involves Linear combination of all models and using Logistic Regression and Random Forest Classifier we will make the predictions, to add further we will stack all the obtained results and see which classifier performs best.Below diagram represents the stages of mining the data, the Spark RDD's are converted into Pandas Dataframe and then we have used machine learning libraries on those dataframes.

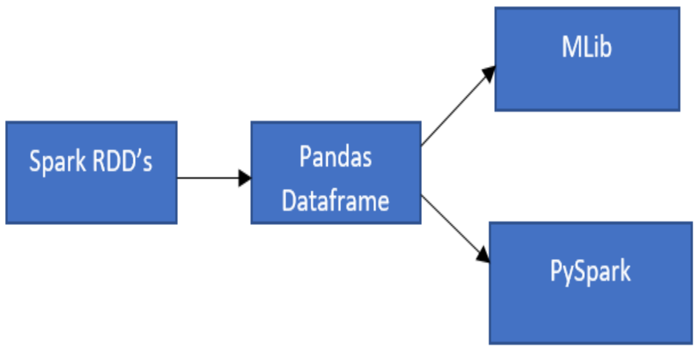

Figure 3: Data Mining Stages

5 Evaluation

5.1 Experiment Setup

To conduct our experiments, we have used Amazon AWS EC2 t2.xlarge instance running on 64-bit Ubuntu Server 16.04 LTS with 4 vCPUs and 16 GB of memory. For data processing, we used Standalone Apache Spark 2.1.1 with PySpark, for machine learning libraries we have used scikit and MLib. We ran our classifier on 8 different threads running on AWS instance.

5.2 Data Set:

To test our approaches, we have used NSL-KDD[2] and KDD99 dataset. KDD99 data set contains total of 4 million record counts, the data set is divided into train and test data containing of 4898431 record count in training and 311029 record counts in testing data set. The dataset is splitted into test data which consists of 22544 records and train data consists of 125973 records which makes it affordable to train the IDS classifier and conduct experiments The records in the dataset are categorized into four types of intrusion attacks like DDoS, R2L (Remote to Local), U2R (Unauthorized Access to Root) and probing attacks (Tavallaee et al.; 2009). Each connection is identified as normal or attack.

5.3 Advantages of NSL-KDD over KDD99

The NSL-KDD data set is considered as the enhanced version of KDD99 dataset [34]. The NSL-KDD dataset does not include any redundant or duplicate records in train data making it less biased towards more frequent records. There are no duplicate records in test set either so the overall performance of classifier are not biased on methods having good detection rate on frequent records. The number of records in train and test data are reasonable to perform tests thus making it consistent and comparable.

5.4 Approach Evaluations:

Since the data set is unlabelled we labelled the data so that it will be easy for processing, we grouped the data into two types normal and attack, out of which attacks are categorized as DoS, Probe, R2L and U2L

The below tables specify the number of attack and normal count that are present in the data set.

[2]https://github.com/defcom17/NSL-KDD
[3]http://www.unb.ca/cic/research/datasets/nsl.html
[4]http://kdd.ics.uci.edu/databases/kddcup99/task.html

Labels	**Count**
Normal	67343
Attack	58630

Labels	**Counts**
Normal	67343
DoS	45927
Probe	11656
R2L	995
U2R	52

Table 3: Train Data

Labels	**Count**
Normal	9711
Attack	12833

Labels	**Counts**
Normal	9711
DoS	7458
Probe	2754
R2L	2421
U2R	200

Table 4: Test data

For proper demonstration we have visualized the data by using Principle Component Analysis (PCA) following are the results. The first plotted graph shows the normal vs attack counts, the red one specifies the normal connection while grey ones are attacks, while second one represents the four different types of attack vs normal connections

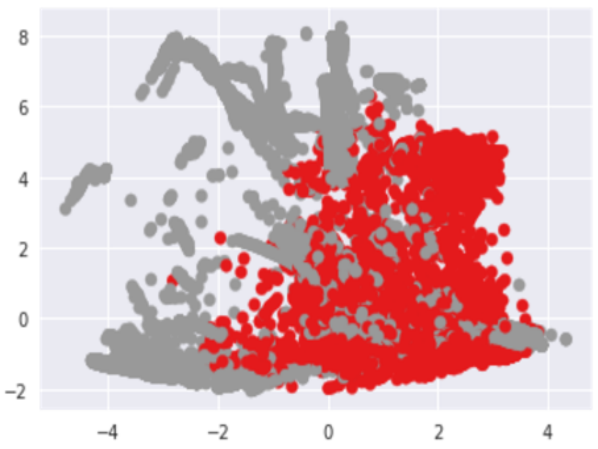

Figure 4: Normal vs Attack Connections

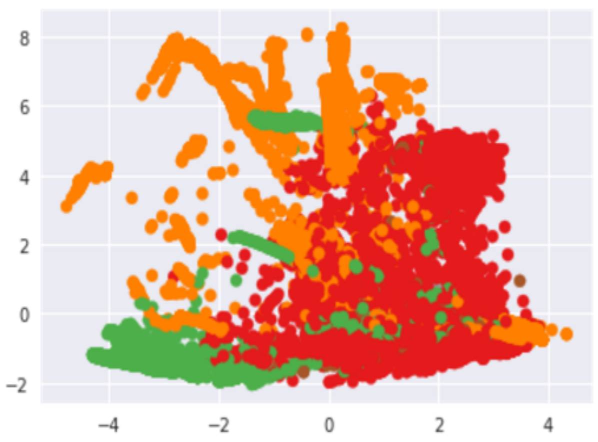

Figure 5: Types of Attacks vs Normal Connections

5.5 Results of K-means and Gaussian Mixture Clustering using Random Forest Classifier:

In this approach, we applied K-means clustering and Gaussian Mixture using Random Forest Classifier, to do this we have divided the clusters into 8 centroids and the first category contains the attack and normal connection having more than 25 connections, while second category contains all the clusters and mapping of attack and normal connections is done based on majority. Then we train Random Forest Classifier on each of the clusters.

Clusters	**Time Taken**
Cluster 0	15.46 sec
Cluster 1	2.681 sec
Cluster 2	15.72 sec
Cluster 3	153.587 sec
Cluster 4	4.69 sec
Cluster 5	16.21 sec
Cluster 6	1.31 sec
Total	**209.680 sec**

Table 5: K-means Cluster Time

Clusters	**Time Taken**
Cluster 1	3.317 sec
Cluster 4	118.179 sec
Cluster 5	51.958 sec
Cluster 6	9.87 sec
Cluster 7	7.95 sec
Total	**191.289 sec**

Table 6: Gaussian Mixture Cluster Time

Table 3 and 4 represents the time taken to train each cluster, by looking at the table we come to know that the training time taken by Gaussian Mixture clusters is less than K-means clusters

5.6 Performance Matrices Classification:

After clustering the data, we have measured the accuracy of both algorithms, accuracy is calculated based on certain parameters like True Positive (TP) and True Negative (TN) in which the attacks or normal connections are identified correctly and another one is False Positive (FP) and False Negative (FN) which specifies the attacks and normal connections identified incorrectly.

a. Prediction Time specifies the time taken by algorithm to make prediction of whole data set classified as normal or attack

b. Accuracy specifies that how accurately the classifier has identified the connection as normal or attack. It can be measured as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

c. Where detection rate is calculated as:

$$\frac{TP}{TP + FN} \tag{4}$$

Below are the results that we calculated based on above mentioned parameters

	False Alarm Rate	**Detection Rate**	**F1 score**	**Accuracy**	**Prediction Time**
K-means	14%	97%	0.93	92%	16.67 sec
Gaussian Mixture	14%	94%	0.92	90%	23.39 sec

Table 7: K-means and Gaussian Mixture Test Data

	False Alarm Rate	**Detection Rate**	**F1 score**	**Accuracy**	**Prediction Time**
K-means	0.0011	99%	0.99	99%	17.96 sec
Gaussian Mixture	0.00045	99%	0.99	99%	22.40 sec

Table 8: K-means and Gaussian Mixture Train Data

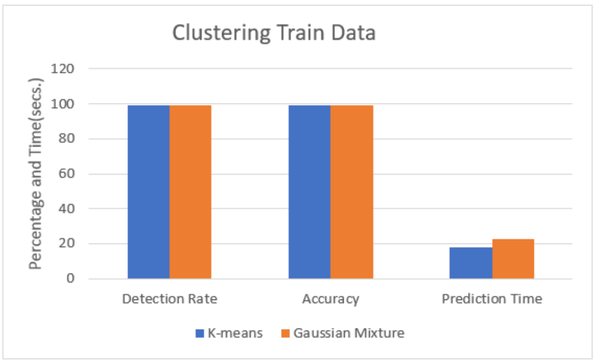

Figure 6: Clustering on Train Data

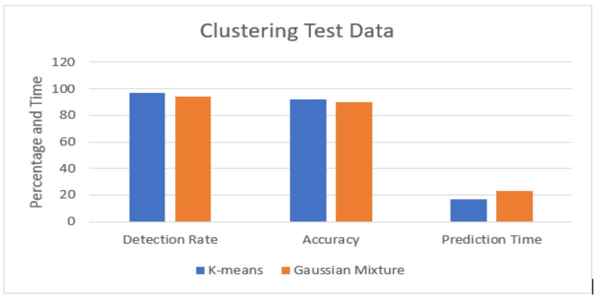

Figure 7: Clustering on Test Data

In our experiments, we see that K-means achieves the overall accuracy of 92% on test data while achieving 98-99% on train data which is better than that of Gaussian Mixture which manage to achieve 90% on test data and around 98-99% on train data with 14-15% False Alarm Rate, however when it comes to prediction time we observed that K-means performed better than Gaussian Mixture in both training and test data set, taking less time for making predictions.

5.7 Analysis of Supervised Approach:

Our first approach involved testing two clustering algorithms together, in second approach we are going to train Random Forest Classifier on each of the attacks separately and then combine them and check the accuracy. The below table portrays the total count of normal connections and number of different attack counts.

Attack Type	Counts
Normal	54015
DoS	36735
Probe	9271
R2L	782
U2R	37

Table 9: Attack Counts

	False Alarm Rate	Detection Rate	F1 score	Accuracy	Prediction Time
DoS	0.0088	77%	0.92	89%	11.69 sec
Probe	0.00224	60%	0.71	90%	6.29 sec
R2L& U2R	0.00020	76%	0.14	78%	6.20 sec
Combined Results	13%	91%	0.93	90%	

Table 10: Results on Different Attack Types

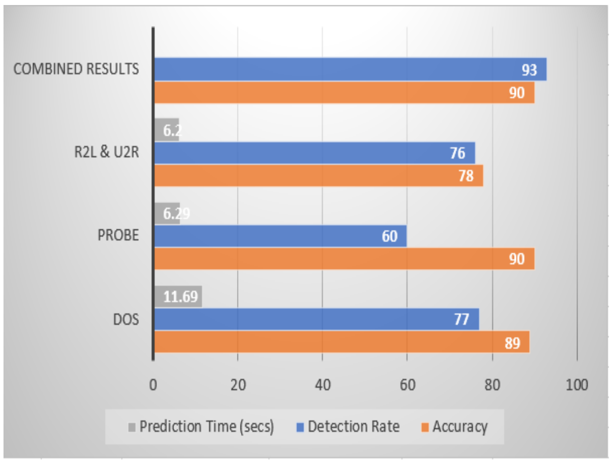

Figure 8: Clustering on Test Data

In this approach we identify each attack separately by doing so the accuracy achieved in identifying the DoS attacks on test data set is around 89-90% with detection rate of 77% and with Probe type of attacks we get an accuracy of 90-91% and detection rate of around 60% while testing R2L and U2L type of attacks the classifier achieves the accuracy of 78-79% and detection rate of 76% but when it comes to prediction time DoS takes more time compared to others, the reason behind this could be more number of DoS attacks present in dataset compared to Probe, R2L and U2L. However, when we combine all attacks together the classifier achieves accuracy of around 90-91%

5.8 Analysis of Ensembling Approach:

In this approach, we have tested our classifier using Logistic Regression and Random Forest Classifiers for this purpose we have made use of scikit libraries which has in build functions to perform these operations. Following are results obtained from our tests

	False Alarm Rate	**Detection Rate**	**F1 score**	**Accuracy**	**Prediction Time**
Logistic Regression	0.15	99%	0.93	92%	32.77 sec
Random Forest Classification	0.16	99%	0.93	92%	28.34 sec

Table 11: Logistic Regression and Random Forest on Test Data

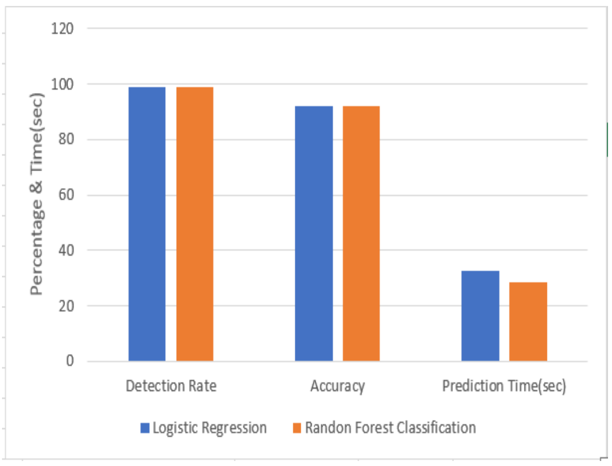

Figure 9: Logistic Regression and Random Forest Test Data

	Detection Rate	**Accuracy**	**Prediction Time**
Logistic Regression	99%	91%	174.77 sec
Random Forest Classification	99%	91%	138.34 sec

Table 12: Logistic Regression and Random Forest on KDD99

Our classifier manages to achieve same accuracy of 92% among both algorithms while detection rate is also similar,while we get False Alarm rate of 15-16% for both algorithm the only difference we see is in terms of prediction time, Random Forest took less time approximately 29 seconds compared to Logistic Regression which took around 10 seconds more. When we ran our test on entire KDD99 data set both algorithm performed well just having difference in prediction time, this comparison was made to justify our selection of Random Forest, as Logistic Regression are good with linear data and cannot handle categorical (binary) data properly and they are not good enough with large amount of data.

	False Alarm Rate	**Detection Rate**	**F1 score**	**Accuracy**
Overall	0.16	99%	0.93	92%

Table 13: Overall Performance

The overall performance of the classifier has been tested by combining all the approaches together and then determining which performs the best among them. So, with ensembling approach our classifier manages to achieve 92% accuracy and around 99% detection rate which is higher compared to previous approaches.

6 Conclusion and Future Work

In this paper, we have successfully tested various intrusion detection approaches in which K-means and Gaussian Mixture model demonstrate a stochastic approach when it comes to detecting network traffic based anomalies, apart from that we also trained our classifier to detect individual types of attack using feature selection algorithm. To add further, we ensembled all the approaches and compared the results. All these things are processed on Apache Spark which provides fast data processing, all the experimentation was carried on NSL-KDD dataset.

Based on our experimentation we found out that in first approach K-means performed better than Gaussian Mixture achieving accuracy of 92% on test data and detection rate of 97%, in second approach our classifier managed to achieve 90% accuracy and in ensembling approach the overall accuracy was around 92% but when it comes to detection rate we managed to achieve 99% which was higher than both the approaches.

So in conclusion, we can say say that ensemble approaches are more accurate in detecting network intrusions compared to individual approaches.

Future works involves testing intrusion detection with different clustering algorithm in a distributed environment, some researchers have also suggested various techniques for intrusion detections based on neural networks. So, testing some of these techniques on self-collected TCP dumps can be a point of interest.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Horacio Gonzalez-Velez for his guidance throughout my entire project. His valuable suggestions played key part in successfully completion of this project, I would also like to thank my external supervisor Dr. Jason Roche for some important insights and all the staff members of National College of Ireland for all the facilities provided during this research project. Also, Special thanks to God, my Parents and my beloved ones who's constant love and support kept me motivated.

References

Aygn, R. C. and Yavuz, A. G. (2017). A stochastic data discrimination based autoencoder approach for network anomaly detection, *2017 25th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4.

Bjerkestrand, T., Tsaptsinos, D. and Pfluegel, E. (2015). An evaluation of feature selection and reduction algorithms for network ids data, *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pp. 1–2.

Deng, S., Zhou, A., Yue, D., Hu, B. and Zhu, L. (2017). Distributed intrusion detection based on hybrid gene expression programming and cloud computing in a cyber physical power system, *The Institute of Engineering and Technology* **11**: 1822–1829.

Enache, A. C., Sgrciu, V. and Togan, M. (2017). Comparative study on feature selection methods rooted in swarm intelligence for intrusion detection, *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, pp. 239–244.

Hsieh, C. J. and Chan, T. Y. (2016). Detection ddos attacks based on neural-network using apache spark, *2016 International Conference on Applied System Innovation (ICASI)*, pp. 1–4.

Huang, M.-Y., Jasper, R. J. and Wicks, T. M. (1999). A large scale distributed intrusion detection framework based on attack strategy analysis, *Comput. Netw.* **31**(23-24): 2465–2475.

Kulariya, M., Saraf, P., Ranjan, R. and Gupta, G. P. (2016). Performance analysis of network intrusion detection schemes using apache spark, *2016 International Conference on Communication and Signal Processing (ICCSP)*, pp. 1973–1977.

Kumari, R., Sheetanshu, Singh, M. K., Jha, R. and Singh, N. K. (2016). Anomaly detection in network traffic using k-mean clustering, *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pp. 387–393.

McElwee, S. (2017). Active learning intrusion detection using k-means clustering selection, *SoutheastCon 2017*, pp. 1–7.

Pallaprolu, S. C., Namayanja, J. M., Janeja, V. P. and Adithya, C. T. S. (2016). Label propagation in big data to detect remote access trojans, *2016 IEEE International Conference on Big Data (Big Data)*, pp. 3539–3547.

Sang-Hyun, C. and Hee-Su, C. (2014). Feature selection using attribute ratio in nsl-kdd data, *International Conference Data Mining, Civil and Mechanical Engineering (ICDMCME2014)*.

Scarfone, K. and Mell, P. (2010). *Intrusion Detection and Prevention Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 177–192.

Shah, R. A., Qian, Y. and Mahdi, G. (2016). Group feature selection via structural sparse logistic regression for ids, *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 594–600.

Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set, *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S. and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, USENIX Association, Berkeley, CA, USA, pp. 2–2.

Zhang, X., An, J., Wang, Y. and Liu, W. (2017). The application of immune clone algorithm in network intrusion detection, *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pp. 619–622.