INTEGRATE MODEL AND INSTANCE BASED MACHINE LEARNING FOR NETWORK INTRUSION DETECTION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Lena Ara

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2018

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Brian King, Co-Chair

Electrical and Computer Engineering

Dr. Xiao Luo, Co-Chair

Computer Information Technology

Dr. Mohamed El-Sharkawy

Electrical and Computer Engineering

Approved by:

Dr. Brian King

Head of the Graduate Program

This research work is dedicated to my parents.

ACKNOWLEDGMENTS

This research work is dedicated to my father Dr. D. M. Afzal Khan, mother Mrs. Tabassum Ara for their unbeatable love and support throughout my Masters in the USA. I am thankful to my sisters for their affection and motivation. Heartfelt thanks to my advisor Dr. Xiao Luo for motivating to learn and explore in Intrusion Detection System using Machine Learning. Her guidance has played a crucial role throughout my journey at IUPUI. I would like to thank my committee members Dr. Brian King and Dr. Mohamed El Sharkawy for their valuable feedback. I am thankful to Sherrie Tucker for her timely help during deadlines and with thesis documentation. I thank all my friends for being patient and helpful during the final stage of submission. A special thanks to my Pinnacle Solutions team for their support and understanding.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

SYMBOLS

χ	Chi
ζ	Regularization factor
P	Probability
E	Expectation
$\|\|.\|\|$	Vector norm
α	Threshold

ABBREVIATIONS

IDS	Intrusion Detection System
SVM	Support Vector Machines
KNN	K Nearest Neighbors
DR	Detection Rate
FPR	False Positive Rate
TPR	True Positive Rate
RF	Random Forests
NB	Naive Bayes
LR	Logistic Regression

ABSTRACT

Ara, Lena. M.S.E.C.E, Purdue University, December 2018. Integrate Model and Instance Based Machine Learning for Network Intrusion Detection. Major Professors: Xiao Luo and Brian King.

In computer networks, the convenient internet access facilitates internet services, but at the same time also augments the spread of malicious software which could represent an attack or unauthorized access. Thereby, making the intrusion detection an important area to explore for detecting these unwanted activities. This thesis concentrates on combining the Model and Instance Based Machine Learning for detecting intrusions through a series of algorithms starting from clustering the similar hosts. Similar hosts have been found based on the supervised machine learning techniques like Support Vector Machines, Decision Trees and K Nearest Neighbors using our proposed Data Fusion algorithm. Maximal cliques of Graph Theory has been explored to find the clusters. A recursive way is proposed to merge the decision areas of best features. The idea is to implement a combination of model and instance based machine learning and analyze how it performs as compared to a conventional machine learning algorithm like Random Forest for intrusion detection. The system has been evaluated on three datasets by CTU-13 [1]. The results show that our proposed method gives better detection rate as compared to traditional methods which might overfit the data. The research work done in model merging, instance based learning, random forests, data mining and ensemble learning with regards to intrusion detection have been studied and taken as reference.

1. INTRODUCTION

The convenient and rapid Internet access has been facilitating many Internet services, but also continues to accelerate the spreading of malicious software. This makes the detection of malicious activities difficult. Computer networks have been playing important roles in modern society, however, their security is always at risk. We need efficient ways to protect the system. Intrusion detection [2] is a technology to protect the network from malicious activities.

The detection on network traffic is critical in preventing intrusions. Since intrusion detection is for network data with large traffic size, machine learning algorithms seem to be a good approach for handling this problem.

Utilizing our proposed data fusion along with clustering algorithms, this research work explores an approach to identify intrusions on network following the identification of maximal clusters of hosts. We have found clusters of hosts that we believe has given us a more robust model without retraining. The already learned model of hosts that are in a cluster are merged based on the decision regions obtained from their decision making trees, thus reducing the retraining cost. The decision regions of the trees of the hosts in the cluster have been merged to obtain a model. Then according to our proposed algorithm intrusion detection is done using either the model or the instance based knn approach.

Machine learning algorithms [3] [4] have been investigated for intrusion detection in the past years. Many researches are being attempted to apply machine learning techniques to this area of cybersecurity [5]. Research work by [5] implemented IDS using ML at application layer where they explain that it is challenging to stay fixed with the features. They analyzed the payload and the type of code it contains like Javascript and SQL. At host level, one or more machine learning algorithms can be deployed on the hosts of a network for identifying intrusions. Generating robust learning models from the training data pose to be a critical component for using supervised machine learning algorithms for intrusion detection. The training needs to be efficient that relies on the huge amount of network traffic.

Definitely, there are a lot of challenges involved here. These challenges include the generation of robust learning models, systematic training methods on large size data and the periodic refreshing of detecting models for showing up the latest network traffic behavior and attack patterns. This research work focuses on intrusion detection for similar and non similar hosts on a network and new incoming data. For new incoming traffic, if the network flows are similar with regards to the intrusion detection, it might imply that these are under the attack by a hacker who is using the same hacking mechanism to attack different machines in a network. We have proposed a technique which includes model and instance based approach.

Furthermore, if the network flows are similar enough, these host then can be clustered into the same group. Finding out the similarity among network flows by comparing the flows of each host one by one to those of another host is not a viable solution. Hence, we proposed to employ machine learning for summarizing the network behaviors from the generated learning models. The hypothesis is that if the generated learning model based on the network flows of one host can be used to detect the intrusion flows of the other host, these two hosts should be in the same cluster. Going deeper with the research, these clusters can be merged into a bigger cluster based on the detection rate of the attack traffic. The ways to merge the decision tree models of hosts in such clusters have been studied. Merging the decision tree regions of similar hosts has been found to be the best approach for our work. We have then compared the results obtained using the proposed method with the traditional way of using machine learning methods. By the traditional machine learning methods we mean having a model achieved by training on the training set and testing on the testing set. Our proposed method is different from the traditional way where we have tried to eliminate the retraining process for the new incoming traffic and thus be more robust.

2. BACKGROUND

2.1 Intrusion Detection System

An intrusion detection system (IDS) is a system that issues alerts when malicious or abnormal activities are detected on a network that is being monitored. Intrusion detection primarily detects and reports malicious activity or anomalous traffic, there are IDS that are capable of taking actions like blocking traffic sent from suspicious IP addresses against malicious behavior.

Significant amount of work has been done in past with respect to intrusion detection. As mentioned in paper [6] two general approaches are popular for intrusion detection: Misuse detection, which involves patterns of known attacks to identify intrusions; and anomaly detection which aims to find the outliers from the normal patterns. Anomaly based IDS monitors network traffic and compares it against an established baseline. This baseline identifies normal and attack traffic. Intuitively, this baseline identifies the sort of bandwidth, protocols and ports and devices connected to are normal for that network and alert the administrator or user when traffic is detected which is aberrant, or significantly different than the baseline. In the research work described in [7] misuse detection and anomaly detection have been combined to detect unknown intrusion and achieve better detection and false positive rates. They extracted class association rules from training database and then classified the data as normal, known intrusion or unknown intrusion through the analyses of matching degree of data with the rules.

Machine learning techniques have been explored for detecting intrusions. According to paper [8] signature based approach has high detection rate but is not efficient for novel attacks. Anomaly based methods can detect but give high false positive rate. Using machine learning, a detection model is developed at the training phase.

In this research, a comparative study of SVM, Naive Bayes, J.48 and decision table for anomaly detection on KDD99 dataset [9] has been presented. They found different results for each class when tested with different algorithms and out of the five algorithms no single algorithm gave high TPR for the classes. But they concluded that decision tree based technique J.48 has high overall accuracy with low false positive rate. Decision tree yielded a good result with redundant features. Their future work mentions the use of feature selection with the J.48 for better results. The research work mentioned in the paper [10] has classified attack types for intrusion detection using machine learning. The performance has been analyzed using Random Forest algorithm on Kyoti 2006+ dataset [11]. Going beyond the supervised techniques for intrusion detection, outlier detection using unsupervised method in cloud computing environment has been explored by [12] [13]. Unsupervised techniques appear to be good approach because they do not require any training data set or any kind of previous knowledge for detecting the possible novel attacks.

2.1.1 Host Based vs Network Based

Based on the location in a network, IDS can be categorized into two groups: Host Based and Network Based.

Host based IDS are at the host level as the name suggests. The traffic originating and coming to a particular host are monitored. Apart from monitoring incoming and outgoing traffic, a host based IDS can also analyze the file system of a host, users login activities, running processes, data integrity etc. As described in Figure 2.1, at host level intrusion detection, the hosts on a network can deploy one or more machine learning algorithms for intrusion detection.

Network based IDS are the types of IDS which are strategically positioned in a network for detecting any attack on the hosts of that network. IDS is placed at the entry and exit point of data from the network to the outside world, for capturing all the data passing through the network. IDS can also be strategically positioned,

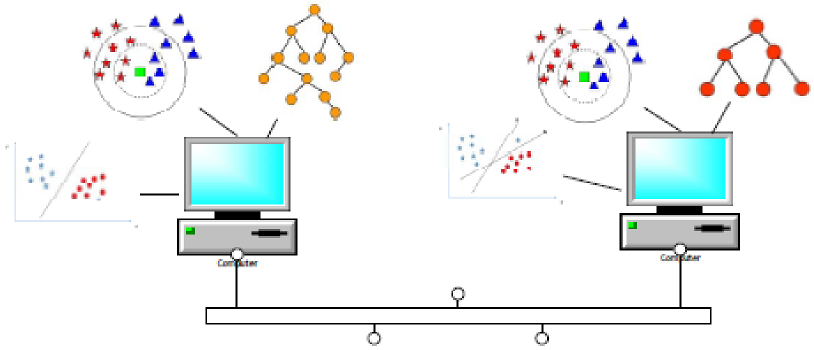


Fig. 2.1. Machine Learning for Host Based Intrusion Detection

depending on the level of security needed in our network. The speed of traffic analysis is important, since the network based IDS need to monitor all the data passing through the network with dropping as much little traffic as possible.

2.2 Machine Learning for Network Traffic Analysis

Machine Learning algorithms are efficient way of tackling problems with huge data. The work done by [14] have used various ML algorithms to compare and find the algorithm which gives the best accuracy. Their work shows that decision tree C4.5 classifier gives the best accuracy among Support Vector Machines, Decision tree C4.5, Bayesian networks and Naive Bayes. Feature scaling before implementing any algorithm has also been done to select the best features that give most accurate classification.

2.2.1 Supervised Learning

In supervised learning an algorithm is employed to learn a mapping function from the input variables (X) to the output variable (Y). The training dataset can be thought of as a teacher supervising the learning process, so it is called supervised learning. The learning continues and stops when the algorithm achieves an acceptable level of performance after the algorithm iteratively makes predictions on the training data. Supervised learning includes regression and classification. Supervised machine learning algorithms like Linear Regression for regression problems, Random Forest for classification and regression problems, Support Vector Machines for classification problems are well known algorithms.

2.2.2 Unsupervised Learning

Ensuring the security of large and complex networks appears to be a challenging task. Unsupervised ML techniques have also been explored in past to study anomaly detection. Research work by [15] focuses on unsupervised learning algorithms for approaching the problem of anomalies in complex systems. They implemented Self organizing maps that overcome the dimensionality problem of network based systems.

2.3 Model Based and Instance Based

Model based approach have been extensively used for solving machine learning problems. As the name suggests there is a model to predict the output either for regression or classification. Instance based learning also called memory based learning is described as the family of learning algorithms that compares new problem instances with instances seen in training that are stored in the memory. So, instead of performing explicit generalization, it uses memory based techniques.

As described in this research [16], the central idea of model-based is to create a custom model tailored specifically to each application. The model can even be associated together with an inference algorithm. ”Typically, model-based machine learning will be implemented using a model specification language in which the model can be defined using compact code, from which the software implementing that model can be generated automatically”. The key goals of model based approach have been mentioned. These goals include the specific nature of models corresponding to the application. The application might include a combination of clustering and classification. Describing the model based approach, in this study they focused on a framework based on Bayesian inference in probabilistic graphical models.

Model based approaches are studied in a vast area of applications including medical research areas. For instance model based techniques are explored in this paper [17]. In this research they have explored both model-based like logistic regression and model-free or non-parametric techniques like random forests, support vector machines for investigating falls in patients with Parkinson's disease. They used generalized linear models with linear regression when the outcomes are measured on binary scale and follow Bernoulli distribution. Comparing the two techniques their research depicts scalability for different problem statements. Algorithms like Random Forest, AdaBoost, Support Vector Machines, Neural Network benefit from constant learning or retraining as optimized results are not guaranteed.

On the other hand, instance based approaches have also been used in various applications. The idea behind instance-based learning is that similar examples have similar label. KNN is instance based algorithm where selecting the optimum number of neighbors is important. Depending on the data, usually value of k is selected neither too large nor too small.

3. SYSTEM FRAMEWORK

3.1 Overview

In this research, we have experimented with three different supervised learning algorithms - Decision Tree, k-Nearest Neighbors and Support Vector Machines. A data fusion algorithm and host clustering algorithm are developed to group the hosts into clusters based on their known network flows. The host clustering algorithm is based on the Bron-Kerbosch algorithm [18] which is used to identify the maximal complete subgraphs (Cliques) in an undirected graph. Finally, attack traffic is identified by merged model which is a model and instance based technique. Three subsets of the publicly available botnet intrusion detection data set CTU-13 [1] are used in our work where first step is to identify the clusters of the hosts with regards to intrusion detection and then integrate the final model. Each of the three datasets belong to a different bot category from thirteen of the CTU-datasets.

3.2 System Framework

In this work, we have found a generalized model for our problem, so that retraining efforts are reduced for the hosts. The generalized merged model is found by first finding similar hosts. The assumption is that the network traffic flows (normal vs. attack) can be summarized and inferred by the machine learning model that is derived by the learning algorithm and the training data. If the models based on the same learning algorithm and training data of two hosts are the same or one is a subset of the other, these two hosts can be clustered into one cluster. Otherwise, they are not in the same cluster. However, different learning algorithms make use of different optimization functions and processes to generate models. In this work, we proposed the host clustering algorithm that can make use of m different learning algorithms.

Indeed, some pre-processing steps including feature reconstruction are employed before the network flows are fed into the learning algorithms. The non-numerical values for some features, such as protocol and direction have been replaced by identifiable numbers using one-hot encoding [19]. For example, TCP protocol is replaced by value 12. After pre-processing, all network flows are used to train the supervised learning algorithms to generate the learning models respectively. It is worth noting that the testing process is different from the general training and testing process. The training process is to generate the model based on all network flows of a host, the testing process is to test the correctness of the generated model on classifying the network flows of the other host. We call this testing process as cross-testing. Once the cross-testing matrices are obtained, they are further used in the Data Fusion algorithm described in the chapter Host Clustering.

In this research, we applied three widely used supervised learning algorithms: k-Nearest Neighbors, Decision Tree and Support Vector Machines. These three learning algorithms are briefly described in the following sections.

Decision Tree

Decision tree learning techniques have been used for network intrusion detection in the literature and archived competitive performances [20] [21]. There are different decision tree algorithms, such as ID3, C4.5, C5.0 and CART (Classification and Regression Trees). Decision tree algorithm such as J.48 [22] have been used for detecting intrusions as well. The trained model of the decision tree can be decoded and visualized as a tree structured form. Whereas, other trained models are difficult to interpret. That is one of the important differences between decision tree algorithms and other supervised learning algorithms. In this work, optimized CART which was implemented in python Scikit-Learn package, was employed. CART is very similar to C4.5. The difference is that it supports numerical target variables (regression) and does not compute rule sets. Decision trees algorithm is considered a simple yet effective supervised ML algorithm. The trees often mimic the human level thinking which makes them simple to understand the data and provide good interpretations.

K-Nearest Neighbors

k-Nearest Neighbors (k-NN) learning technique has been used in pattern recognition and classification since the beginning of 1970s as a non-parametric technique. It has been studied extensively for data mining and network flow analysis [23] [24]. k-NN uses a majority vote of its neighbors to classify a data instance. k is a positive integer. If k = 1, then the data instance is simply assigned to the category of its nearest neighbor. Typically, we experiment with a different k to find the optimal k value to classify a given data set. The neighborhood distance function varies from different implementation of k-NN. The typical one is Euclidean distance which was used in this work. The k value was 2 in this work.

Support Vector Machines

Support Vector Machines (SVMs) was one of the classification algorithms evaluated. SVMs is a large margin classifier, and has been widely used in many different data analytic tasks and network intrusion detections [25] [26]. The SVMs classifier aims to separate the input data using hyperplanes. In order to generate a less complex hyperplane function for classification, the maximum margin between the hyperplane and the support vectors which is defined by a fraction of the input data instances is required. Support Vector Machines use a function called kernel function for non-linearly transforming the training features from a two dimensional space to a higher dimensional feature space.

That means the kernel function defines the distance measurement between the data points in the high-dimensional space. For example, if the function is radial basis function (RBF) [27].

$$k(x, y) = exp(-\gamma||x - y||^2) \tag{3.1}$$

where x and y represent two data vectors, γ is training parameter. For making the decision boundary smoother, the parameter γ is used. Another regularization parameter ζ controls the trade-off between low training error and large margin.

The ability to learn from large feature spaces and the dimensionality independence make the support vector machines a universal learner for data classification. In our experiments libsvm [28] has been used for calculating data fusion matrices.

Describing the framework

The intuition behind this research is to identify the known attacks in a faster way without involving a retraining process like in a traditional machine learning approach. Our proposed model uses the data fusion algorithm described in below sections for calculating cross-testing matrices. By cross-testing matrix we mean the matrix obtained by training on one set of hosts and testing on each other. The similar hosts are represented on an undirected graph where an edge between two nodes represent the similarity between them. Going further, the bron kerbosh algorithm implies the output as maximal cliques which are the clusters found. The maximal cliques with largest number of nodes or hosts are combined into a bigger cluster based on their ability to detect attacks. The decision regions of this cluster has been merged according to our proposed decision region merging algorithm after the feature selection. The hosts that are outside the cluster have an instance learning based approach for detecting attacks. The basic overview is described the diagram in Figure 3.1.

The final model shown in this figure is the combination of Merged and Instance Based methods. The approach used in our research work is different than the traditional model which has all data combined and trained for predictions as described in Figure 3.2.

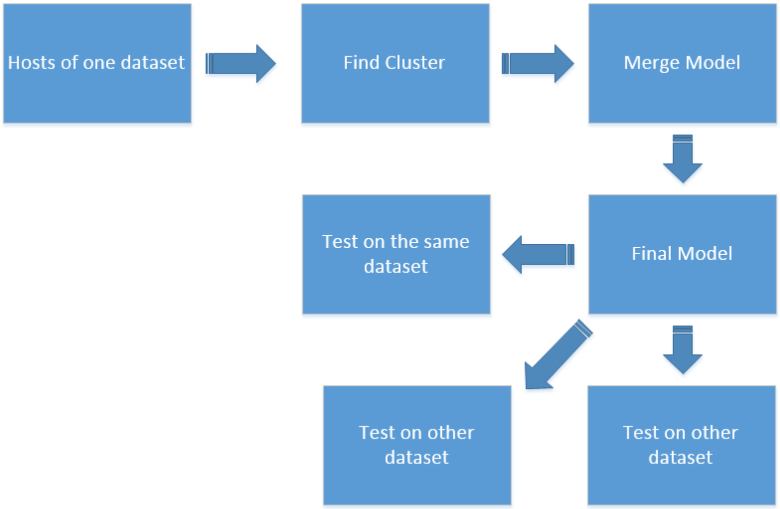


Fig. 3.1. Overview of Our Model

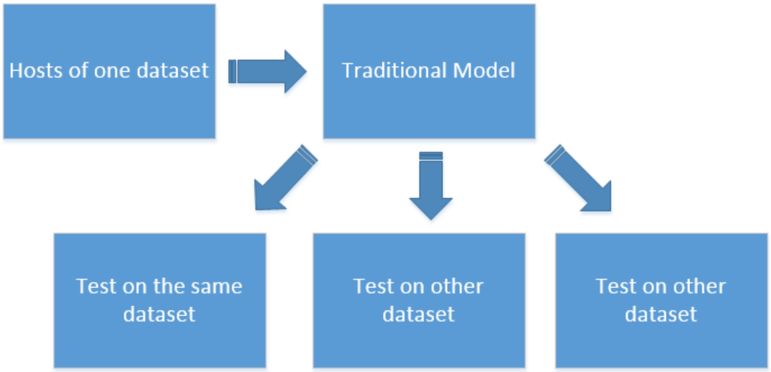


Fig. 3.2. Overview of a Traditional Model

4. DATA SOURCES

We evaluated the proposed system based on the aforementioned supervised learning algorithms using CTU-13 data sets. These were captured at CTU and made publicly available [1].

CTU-13 has 13 different datasets, each one is specified for a botnet, and totally 7 botnet viruses are analyzed [29]. CTU labelled the traffic as: Background, Botnet, C&C Channels and Normal. In our experiments, we used the three subsets of the CTU-13 data sets which contain 6 different characteristics of the botnet attack scenarios, such as ClickFraud, Port Scan, DDoS and so on.

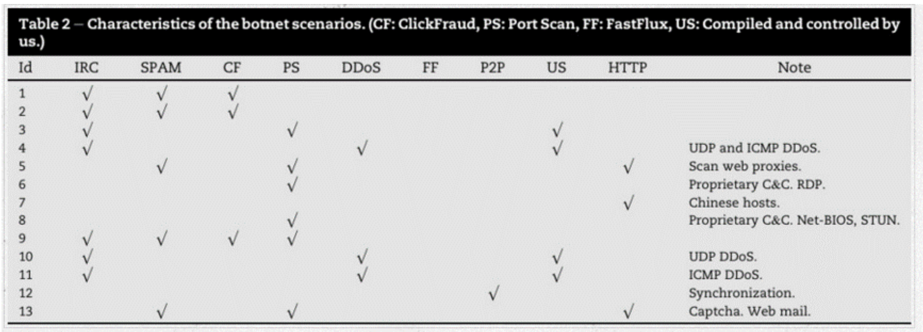


Table 2 – Characteristics of the botnet scenarios. (CF: ClickFraud, PS: Port Scan, FF: FastFlux, US: Compiled and controlled by us.)

Id	IRC	SPAM	CF	PS	DDoS	FF	P2P	US	HTTP	Note
1	√	√	√							
2	√	√	√							
3	√			√				√		
4	√				√			√		UDP and ICMP DDoS.
5		√		√					√	Scan web proxies.
6				√						Proprietary C&C. RDP.
7									√	Chinese hosts.
8				√						Proprietary C&C. Net-BIOS, STUN.
9	√	√	√	√						
10	√				√			√		UDP DDoS.
11	√				√			√		ICMP DDoS.
12							√			Synchronization.
13		√		√					√	Captcha. Web mail.

Fig. 4.1. Characteristics of Botnet Scenarios

There are 13 different sub sets covering different botnet malwares with traffic labels: Background, Botnet, and Normal. The CTU-13 dataset consists of thirteen captures (called scenarios) of different botnet samples. On each scenario a specific malware was executed. Figure 4.1 shows the characteristics of the botnet scenarios as provided in the CTU website. Table 4.1 describes the general types of botnet that are a network security threat.

Table 4.1. Description of Botnets

Botnet	**Description**
IRC Bot	Scripts that connect to Internet Relay Chat as a client [30]
SPAM	Emails about counterfeit goods and security issues
DDOS	Distributed Denial of Service attack targets a system denying services to
P2P	Decentralized botnet without C and C servers

We used 10, 2 and 13 (I, II and III) subsets for our experiments. Since the original data sets have all the network flows of the whole network in one file, the flows are grouped according to the destination IP addresses. Each unique IP address is treated as a host. After grouping the network flows, we found that some of the hosts only contain normal and background network flows. Hence, those hosts are excluded. Only the hosts that contain both normal and botnet attack flows are kept for further analysis.

Following the process described above, in the end, there are total of 16 hosts that are selected for identifying the host clustering in Dataset I, 21 hosts for Dataset II and 18 hosts for Dataset III. 10 network flow features out of 13 are used for the proposed host clustering algorithms. The network flow features are listed in following Table 4.2. For Datasets I and III, experiments are done considering all of the data of each host. For Dataset II, 75 percent of host data has been considered and for testing all of the host data has been considered. There are few hosts in Dataset II with either one normal or botnet instance, those hosts are eliminated in the similar hosts calculation step, but are included as unseen data for testing.

Before beginning further analysis, it is always good to have a look at the data. In Figures 4.2 and 4.3, counts of normal and botnet instances are shown for the Datasets I and II respectively. Dataset II has 21 hosts, since for this dataset, results are calculated 75 percent cross validation, so the hosts with either 1 normal or 1 botnet are not considered here, they are included for testing only. The counts of instances for each host of datasets are shown in Tables A.1 to A.3 in the Appendix. Data distribution for Dataset III is shown in Figure 4.4. The purpose of showing data distribution with and without Host A is to give a clear picture of the distribution because the number of instances of this host is very large as compared to others.

Table 4.2. Selected Network Flow Features of CTU-13 Data

Features	**Description**
Duration	Duration of the connection in seconds
Protocol	Type of the protocol (TCP, UDP, ICMP)
DPort	The port of the connection destination
SPort	The port of the connection source
dTos	Type of Service from destination to source
sTos	Type of Service from source to destination
SrcBytes	Total bytes from source to destination
TotBytes	Total bytes of the flow
TotPkts	Total packets of the flow
Dir	Direction of the flow

Viewing the histogram plots are a quick way for getting an idea of the distribution of each attribute. A count of the number of observations in each bin can be calculated by grouping data into bins using histograms. The histogram plot of host C of Dataset I is shown below in Figure 4.5, where we can clearly see that label 3 (botnets) represented by histogram 11 are very less as compared to label 2 (normals). We can also see that the source ports represented by histogram 2 with less values are also very few from its histogram plot.

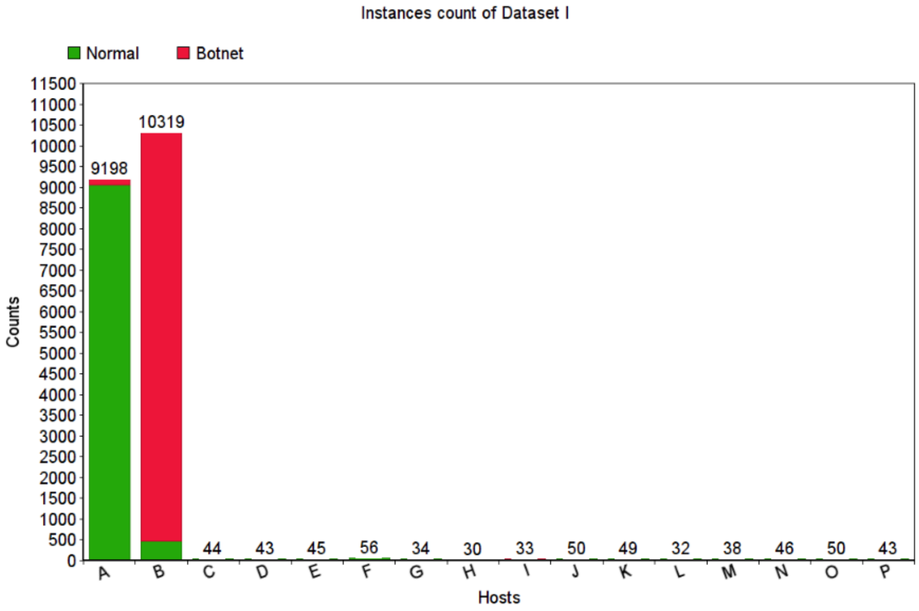


Fig. 4.2. Data Distribution of Dataset I

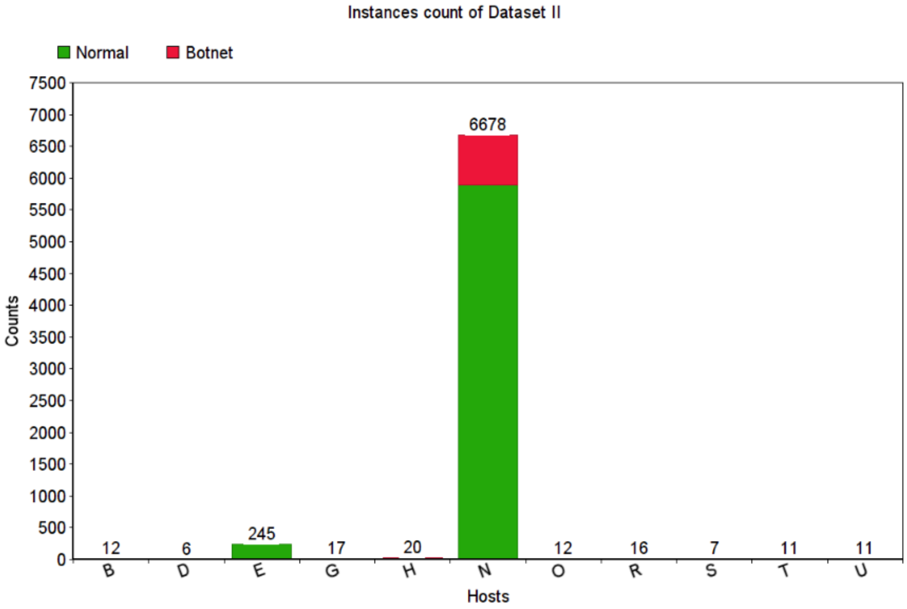
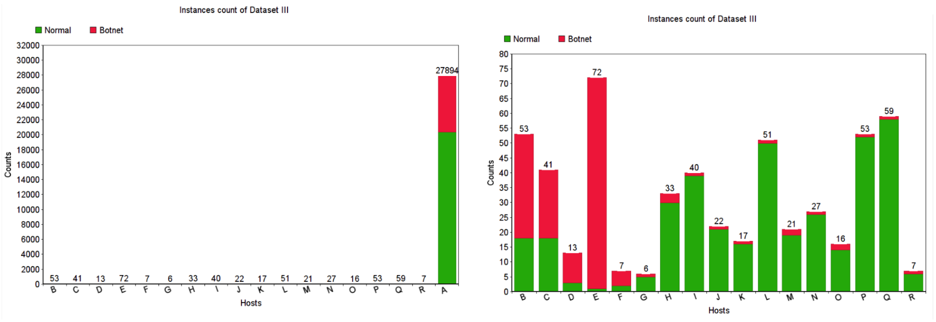


Fig. 4.3. Data Distribution of Dataset II



(a)

(b)

Fig. 4.4. Data Distribution of Dataset III with and without Host A

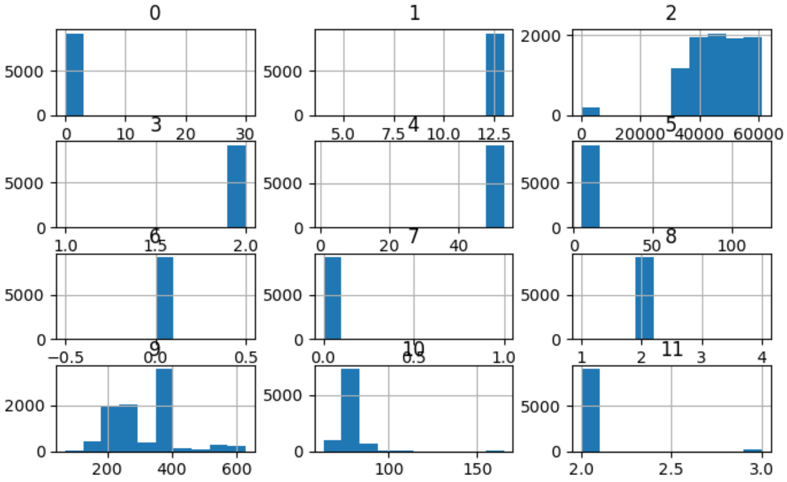


Fig. 4.5. Histogram Plots of One of the Hosts of Dataset I

5. HOST CLUSTERING

5.1 Literature Review

Before implementing our ideas, it is good to study the work that has been done in past. Previous research has been done to cluster or classify hosts based on the traffic behavior or host roles. Wei and colleagues studied host clustering based on traffic behavior profiling [31].

Another research work [32] in host clustering has been done by Xu and colleagues. It includes analyzing behavior of the end hosts in the same network prefixes by considering both host and network level information. A network aware behavior clustering was implemented in the research, where bipartite graphs were implemented to model network traffic. Bipartite graph implementation is obtained based on the communication between source and destination. The bipartite graph was then divided into sub-graphs which represent each source or destination host in the network. The spectral clustering algorithm were used to cluster the end hosts in the same network prefixes.

Some research has used the supervised learning algorithms to classify hosts based on their roles in the network [33]. Some host roles are clients, whereas some are email servers, etc. Because of the huge volume of network traffic and overlap among host roles, modeling host roles based on network flow data is challenging. Derived features based on the network flows, such as number of unique host system ports, standard deviation of ports, number of most often used port, number of unique protocols have been used to identify the roles. However, since host behaviors and roles can change over time, the learning models need to be updated over time.

On the other end, unsupervised and supervised learning algorithms, using flow or packet based data have been investigated for intrusion detection system (IDS) and anomaly detection systems (ADS) since long. Usually the performance metrics of the algorithms are detection rate and false alarm (positive) rate. Many researches [3] [4] [34–38] proposed and explored different machine learning mechanisms for improving the detection rate and reducing the false positive rate and detecting different types of attack behaviors. However, the common challenge is how often to update the generated learning models and whether a generated learning model can be directly applied to a new host without training process. To the best of our knowledge, our research is the first to cluster the hosts based on the cross-testing using the generated models. The following section explains the algorithms that are implemented for host clustering.

5.2 Algorithms

5.2.1 Data Fusion Algorithm

If the model generated by the flows of host A can correctly classify all the network flows of host B and vice versa, that means the network flows patterns of host A and B are very similar to each other. Hence, host A and B can be clustered into the same group. F1-measure (given in Formula 5.1) is used to evaluate the correctness of the cross-testing. Given N hosts within a network, the cross-testing F1-measure values will be stored in a matrix as Formula 5.2.

$$F1 = \frac{2*TruePositive}{(2*TruePositive)+FalseNegative+FalsePositive} \tag{5.1}$$

$$L = \begin{pmatrix} 1 & F1_{1,2} & ... & F1_{1,N} \\ F1_{2,1} & 1 & ... & F1_{2,N} \\ . & . & . & . \\ F1_{N,1} & F1_{N,2} & . & 1 \end{pmatrix} \tag{5.2}$$

In L, F1i;j is the F1-measure of testing the network flows of host j using the machine learning model that is generated on host i based on its network flows. F1i;i represents F1-measure of testing the network flows of host i using the machine learning model that is generated on host i. Essentially, all the values of F1i;i is 1. In this research, a data fusion algorithm is designed to integrate the cross-testing F1-measure matrices (L) of different learning algorithms, then generate host clusters. Given N hosts in the network and m machine learning algorithms, the data fusion algorithm is given in Algorithm 1. Algorithm 1 calls Algorithm 2 to identify the maximal host cluster in a network. Any host in the maximal host cluster are similar to all other hosts within the cluster. In this work, the hosts within the network are treated as nodes in an undirected graph, whereas, the edges between the hosts are the similarities between the hosts. The similarity is reflected by the cross-testing F1-measure. An edge (i;j) exists only when F1(i;j) and F1(j;i) are 1 for all cross-testing F1-measure matrices corresponding to the machine learning algorithms. When considering 75-100 cross-validation threshold may be selected a value less than 1, for example 0.6.

Algorithm 1 Data-Fusion-Algorithm

```
Input: Cross-testing F1-measure matrices $(L1, L2, \ldots, Lm)$, $\alpha$
Output: Cluster
  $L = L1 + L2 + \cdots + Lm$
  $Vertices = \emptyset$
  $Edges = \emptyset$
  for i in $(1, \ldots, N)$ do
    for j in $(1, \ldots, N)$ do
      if $(L_{i,j} = L_{j,i})\&\&(L_{j,i} \geq \alpha)\&\&(i \neq j)$ then
        $Vertices = Vertices \cup \{i\} \cup \{j\}$
        $Edges = (i, j)$
      end if
    end for
  end for
  $X = \emptyset$
  $R = \emptyset$
  $P = Vertices$
  $Cluster = Maximal - Cliques(R, P, X, Edges)$
```

Algorithm 2 Maximal-Cliques

```
Input: R, P, X, Edges
Output: Cliques
  if $(P = \emptyset)\&\&(X = \emptyset)$ then
    $Cliques = R$
  end if
  for each vertex $v$ in $P$ do
    $N(v)$ = Neighbor Set of $v$ based on $Edges$
    $R = Maximal - Cliques(R \cup \{v\}, P \cap N(v), X \cap N(v), Edges)$
    $P = P \setminus v$
    $X = X \cup v$
  end for
```

5.2.2 Maximal Cliques Algorithm

The Algorithm 2 works as Bron-Kerbosch algorithm [18] which is used to find the maximal cliques in an undirected graph. Based on the training and cross-testing processes described in section 5.2.1, three cross-testing F1-measure matrices are generated through applying the three supervised machine learning algorithms. Tables 5.1 to 5.3 present the cross-testing F1-measure matrices of the three learning algorithms. The rows of each matrix represent the hosts of which the network flows are used to generate the learning models. The columns represent the hosts of which the network flows are used to cross-test the learning model to gain the F1-measure values. An undirected graph based on integrating the three F1-measure matrices is presented as Figure 5.1. Through applying the data fusion and maximal cliques identifying algorithms, hosts C, D, E, I and M are identified in the same cluster and hosts P, J, F, L, N in another cluster. These two are the largest clusters within the selected 16 hosts. Host A, B and H have no similarity with any of the other hosts and are outside the clusters.

5.3 Merging the Clusters into One

The maximal clusters obtained are merged into one cluster using the detection rate and false positive rate for the attack traffic. That means the tree structures of the hosts in the two clusters are merged so that each of the cluster has its own decision making model. The data of each cluster is tested on each others cluster model. Since the merged model of each cluster detected the hosts of other clusters with detection rate of 1 and very less false positive rate, they are merged into one bigger cluster. Figure 5.1 shows the merged cluster. The goal of this chapter is to provide an overview of the implementation, the exact results are tabulated in the Results chapter.

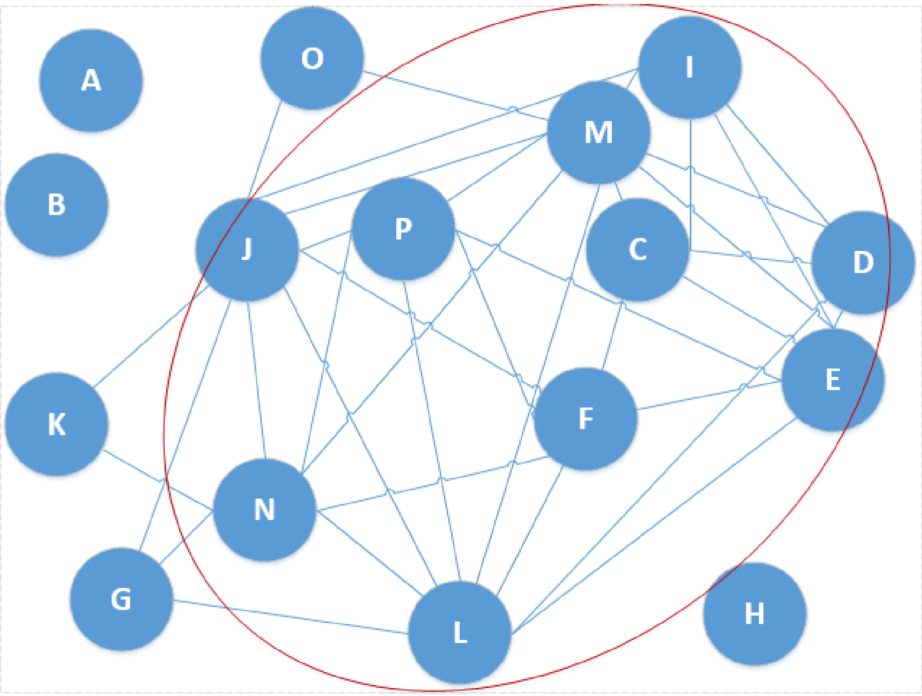


Fig. 5.1. Merged Cluster Illustration

5.4 Merged Model and Instance Based Approach

The merged model explained in the above section works well for most of the hosts. But it is possible that some of the hosts have very different traffic pattern that the merged model is unable to detect the attack traffic. Such hosts have an instance based KNN model to predict the oncoming traffic. This is explained by the flowchart in the Figure 5.2.

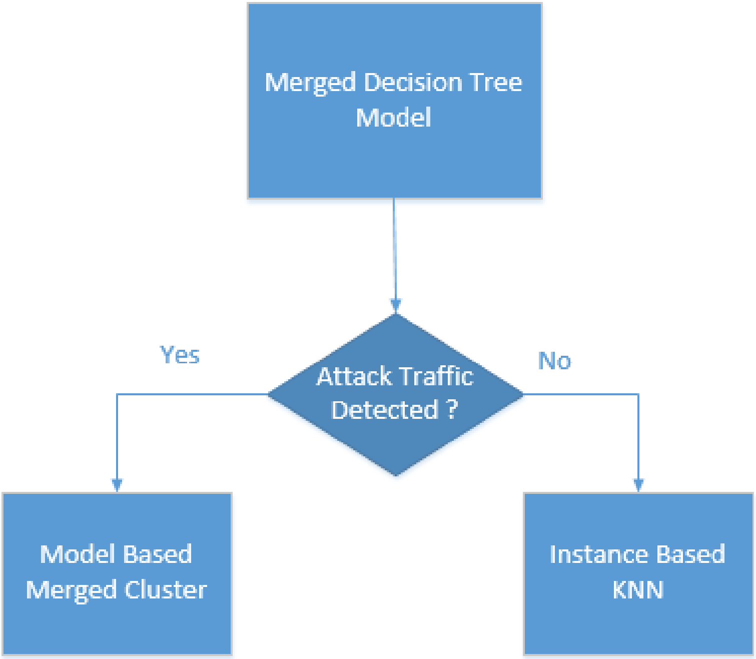

Merged Decision Tree Model
Yes
Attack Traffic Detected ?
No
Model Based Merged Cluster
Instance Based KNN

Fig. 5.2. Model-Instance Based Approach

5.5 Experimental Results

5.5.1 Dataset I

Tables 5.1 to 5.3 present the cross-testing F1-measure matrices of the three learning algorithms. Based on the data fusion algorithm explained above, the following maximal clusters are obtained (shown in Figure 5.3). Figure 5.3 shows the edges that correspond to the similarity between hosts, the maximal clusters with maximum number of hosts are CDEIM, FJLNP and JLMNP. However, for the ease of viewing two of the three clusters are highlighted in this figure.

Table 5.1. Dataset I Cross-testing F1-measure of Decision Tree

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	1	0.06	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	0.5	1	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.5	0.4	0.5	0.4	0.5
C	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
D	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
E	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G	0.1	0.4	0.6	0.6	0.6	1	1	1	0.6	1	0.9	1	0.9	1	0.9	0.9
H	0.4	0.5	0.6	0.8	0.8	0.6	0.7	1	0.7	0.6	0.8	0.6	0.8	0.8	0.7	0.8
I	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
J	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
K	0.1	0.4	0.6	0.6	0.5	1	1	1	0.6	1	1	1	0.9	1	0.9	0.8
L	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
M	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
N	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
O	0.1	0.4	0.6	0.6	0.5	1	0.6	1	0.7	1	0.7	1	1	0.8	1	0.7
P	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 5.2. Dataset I Cross-testing F1-measure of k-Nearest Neighbors

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	1	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	0.0	1	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.1	0.0	0.1	0.0	0.0	0.0
C	0.9	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
D	0.9	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
E	0.9	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F	0.9	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
G	0.9	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
H	0.9	0.3	1	1	1	1	1	1	1	1	1	1	1	1	1	1
I	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
J	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
K	0.9	0.3	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L	0.9	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
M	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
N	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
O	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
P	0.9	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 5.3. Dataset I Cross-testing F1-measure of Support Vector Machines

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	1	0.4	0.4	0.5	0.4	0.5	0.4	0.4	0.4	0.5	0.4	0.4	0.4	0.4	0.4	0.4
B	0.5	1	0.4	0.4	0.4	0.4	0.4	0.3	0.3	0.3	0.3	0.4	0.3	0.3	0.4	0.3
C	0.9	0.3	1	1	1	1	1	1	1	1	1	1	1	1	1	1
D	0.5	0.5	1	1	1	1	1	1	1	1	1	1	1	1	1	1
E	0.9	0.5	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F	0.9	0.4	0.7	0.8	1	1	0.8	1	0.9	1	0.9	1	1	1	0.9	1
G	0.9	0.5	0.8	1	1	1	1	1	1	1	1	1	1	1	1	1
H	0.9	0.4	0.7	0.8	1	1	1	1	1	1	1	1	1	1	1	1
I	0.9	0.4	1	1	1	1	1	1	1	1	1	1	1	1	1	1
J	0.9	0.4	0.8	1	1	1	1	1	1	1	1	1	1	1	1	1
K	0.9	0.4	0.8	1	1	1	1	1	1	1	1	0.8	1	1	1	1
L	0.5	0.5	0.7	1	1	1	1	1	1	1	1	1	1	1	1	1
M	0.9	0.4	1	1	1	1	1	1	1	1	1	1	1	1	1	1
N	0.9	0.3	0.8	1	1	1	1	1	1	1	1	1	1	1	1	1
O	0.9	0.3	0.8	1	1	1	1	1	1	1	1	1	1	1	1	1
P	0.9	0.4	0.8	1	1	1	1	1	1	1	1	1	1	1	1	1

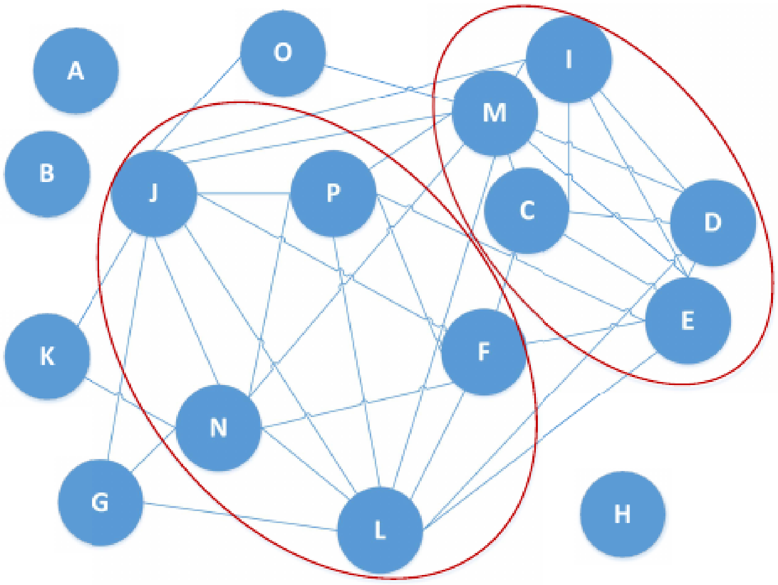


Fig. 5.3. Host Clustering for Dataset I

5.5.2 Dataset II

The cross-testing matrices obtained by using Dataset II and Dataset III are shown in the Appendix Tables A.4 to A.6 and A.7 to A.9 respectively. For Dataset II cross testing matrices are obtained by training on 75 percent and testing on all of the data which also includes the subset (remaining 25 percent). Since there are few hosts that have only one instance of botnet, they are not considered while finding the similar hosts from cross-testing matrices. Leaving such hosts, 11 hosts are selected from 21 of them. Since the cross testing matrices show good F1 scores, the threshold has been taken as 1 for Dataset II as well.

The Figure 5.4 shows the maximal clusters using maximal cluster algorithm. There are 4 maximal clusters with maximum hosts BDOSU, ENO, ENR and HNR.

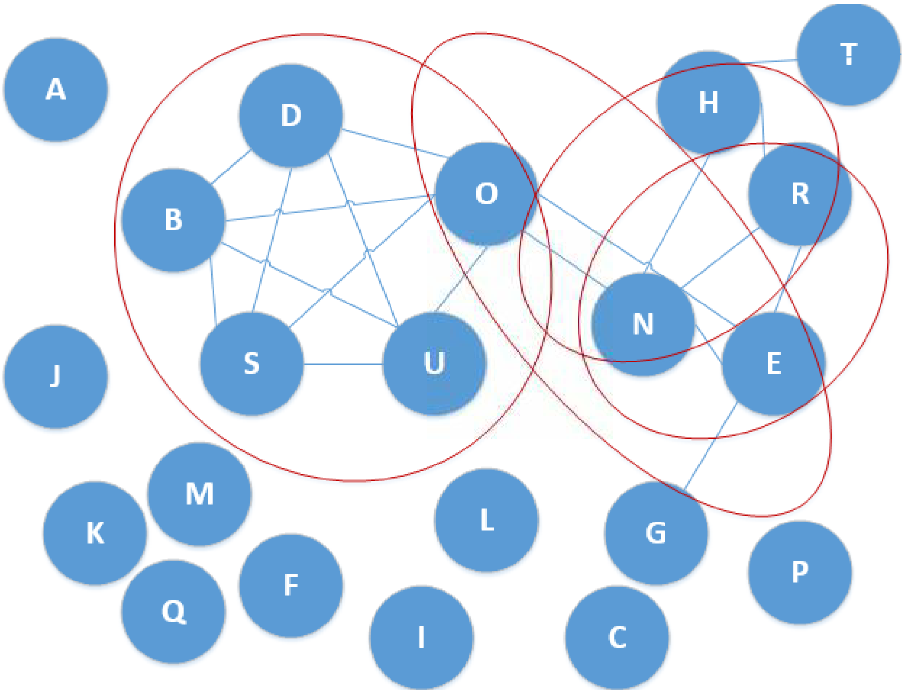


Fig. 5.4. Host Clustering for Dataset II

5.5.3 Dataset III

18 hosts have been found. The cross testing matrices using decision tree, KNN and SVM machine learning algorithms for this dataset are in the Appendix (Tables A.7 to A.9). Figure 5.5 shows the host clustering.

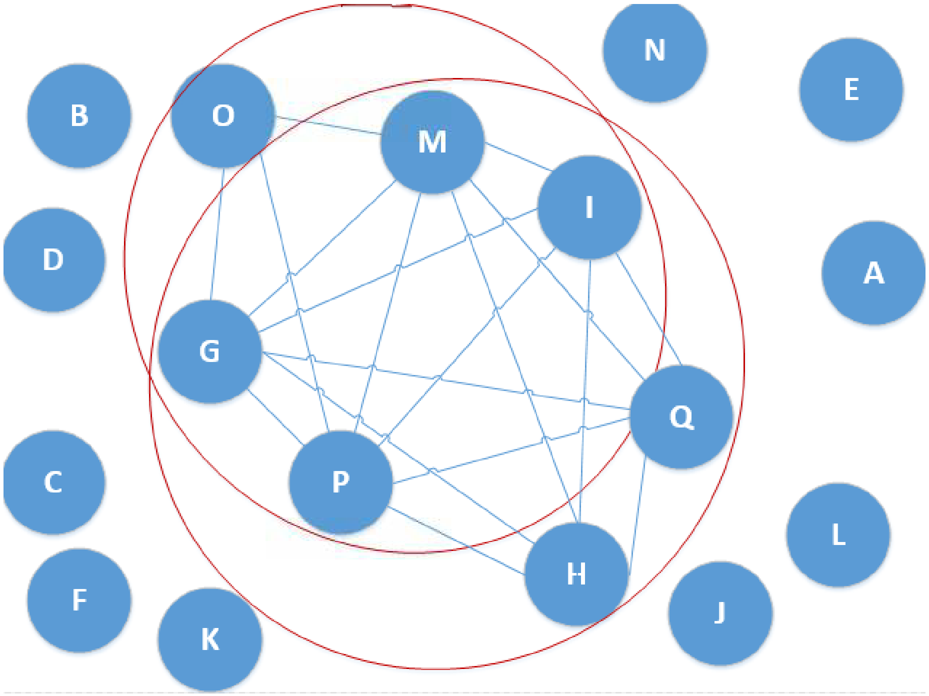


Fig. 5.5. Host Clustering for Dataset III

6. NETWORK INTRUSION DETECTION

6.1 Traditional Methods for Intrusion Detection

Considering intrusion detection as a crucial way of protecting network from malicious activities, researchers have been trying to minimize the threat by various innovative ways. These work include combining anomaly detection and misuse detection [6] using genetic network programming which is a data mining technique. Research work by [39] have enhanced K-Means [40] clustering algorithm to a more robust method underlying the fact that if the clusters are empty in initial iterations over-splitting may be avoided by merging the overlapping clusters, they called it as Y-Means algorithm for intrusion detection. ML algorithms like Multilayer Perceptron, SVM, Multinomial Logistic Regression, Naive Bayes, J.48, Bayesian network have proved to give good detection results [5]. Probability based algorithm like Naive Bayes have been implemented for intrusion detection [41–43]. The Naive Bayes Classifier technique is based on the Bayesian theorem [44] [45] and is believed to be good approach for high dimension data. Known for its simplicity, Naive Bayes can often outperform more sophisticated classification methods. Linear machine learning techniques like logistic regression has also been used for detecting intrusions [46] [47]. Logistic regression is an efficient way for binary classification and uses predictive analysis for assigning observations to a discrete set of classes. Unlike linear regression which gives continuous number values, logistic regression transforms its output using the sigmoid function to return a probability value. This probability value can then be mapped to two or more discrete classes [48] [49].

An extension of the bagged decision trees is Random Forests which aims to reduce overfitting problem. Trees are constructed in a way intended to reduce the correlation between various individual classifiers. These classifiers are constructed using the samples of the training data with replacement. Intuitively, instead of greedily choosing the best split point in the construction of each tree, for each split only a random subset of features are considered. Random Forest has been used in past for obtaining optimal model for intrusion detection [50]. Significant amount of work demonstrates that Random Forest has been used for analyzing network traffic data.

Random forest has been used for feature selection and SVM has been trained on the selected features for intrusion detection on the KDD dataset [9] [51].

Considering the significant amount of work in Intrusion detection using machine learning techniques like Random Forest, we compared the results obtained from our proposed model with the random forests. We have compared the results from our model vs the results obtained by a retrained model using the random forest with 100 trees and split points chosen from a random selection of 3 features. Tables 6.1 to 6.3 show the results using this algorithm trained on Dataset I and tested on each host of itself and Dataset II and III respectively.

Table 6.1. Random Forest Model of Dataset I Tested on Itself

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Avg
DR	1	0.78	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0.98**
FPR	0	0	0.02	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.0012**

Table 6.2. Random Forest Model of Dataset I Tested on each Host of Dataset II

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	Avg
DR	1	0.83	1	1	1	0.94	1	0.83	1	0.34	0.95	1	1	0.33	0.52	0.34	1	1	0.83	0.86	0.89	**0.84**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Similarly, the results using retrained random forest models for Datasets II and III are shown in the Appendix section. Tables A.10 to A.12 show the results using Dataset II and Tables A.13 to A.15 using Dataset III.

Table 6.3. Random Forest Model of Dataset I Tested on each Host of Dataset III

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	Avg
DR	0.81	0.68	0.65	0	0.23	1	0	0.34	0	1	0	1	0	1	0	0	0	0	**0.37**
FPR	0.04	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	**0.03**

6.2 Literature Review of Model Merging

Some work done in the past has been studied and implemented for merging the decision trees. In this research work [52], split point selection approach has been used instead of tree-growing approaches. They exploited the geometry of the attribute space by representing each tree as a set of decision regions or iso-parallel boxes. The merging of k decision trees calls for creating a box set for each of the k trees. As mentioned in their work, computing the label for the two boxes after merging is easy when both have the same labels, it is also assigned the same label. Figure 6.1 describes the merging of decision boxes of two decision trees. One of the strategies used by them to label the intersecting area is averaging the two class probability distributions and selecting the class with the highest probability as the associated label.

Another work by [53] is a survey where they analyzed merging decision trees by data mining approaches. Motivation behind the merging is to generalize the knowledge contained in the models. A larger data set can be split into smaller parts either naturally or artificially. Local models are created using local data sets which provide training examples.

6.2.1 Model Merging Based on Data Mining Approaches

As referred in [52] learning and combining rules on disjoint subsets have been implemented. An algorithm is used to generate the rules on each subset of the training data which are used to construct the merged model. A combination of rules that is

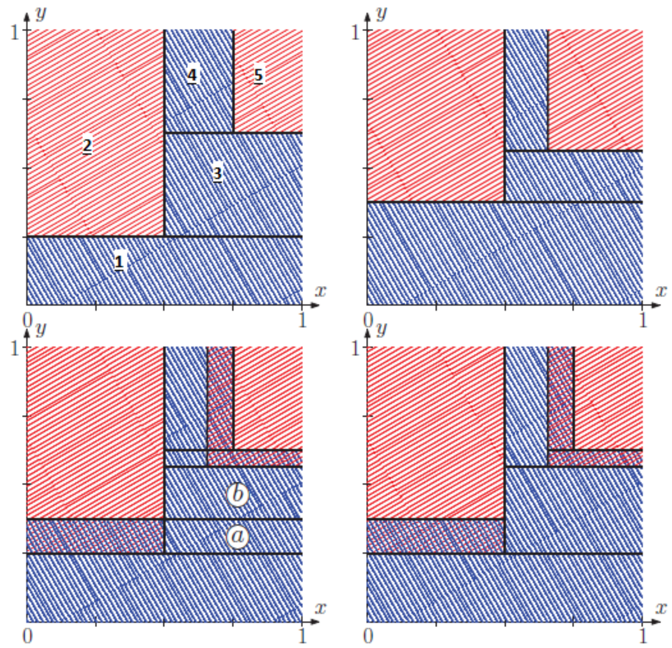


Fig. 6.1. Merging of Decision Boxes Starting from Upper Left [52]

best for the dataset is used. The approach combines the rules into new rules and convert the decision trees from two models into decision rules [52].

6.2.2 Ensemble Learning

Ensemble methods utilize multiple learning algorithms to achieve better predictive performance. An ensemble can be described as a supervised learning algorithm, because it can be trained and then used for making predictions. This method tend to yield better results when the models are significantly diverse. This method consists in combining the predictions of various models into one prediction. The ensembler implements techniques to combine the predictions such as bootstrap aggregating whereas model merging combines the models.

6.3 Proposed Approach for Model Merging

6.3.1 Feature Selection

Feature selection is a process that selects those features from the dataset that contribute most to the prediction [54]. Possessing irrelevant features may decrease accuracy of models. The advantages of performing feature selection before modeling include reduced overfitting, better accuracy and reduced training time. Generally feature selection is used to select the features that will give expected good results after training, but we used feature selection not for retraining the model but to see which of the features have the best relation with the labels. Feature selection can be used for selecting best features or for dimensionality reduction, either to improve estimators accuracy scores or to boost their performance on datasets. Main approaches are described below:

1. Univariate Selection: This selection utilizes statistical tests to select those features that have the strongest relationship with the output variable. In our experiments, we have used the chi-squared (chi^2) statistical test for non-negative features to select four of the best features. Chi^2 feature selection [55] is an efficient way of selecting best features in statistics, it is applied for testing the independence of two events, where according to probability concepts, two events A and B are defined to be independent if:

$$P(AB) = P(A)P(B) \tag{6.1}$$

or,

$$P(A|B) = P(A)P \tag{6.2}$$

and

$$P(B|A) = P(B) \tag{6.3}$$

For using χ^2 for feature selection, a value χ^2 is calculated between each feature and the target, and the desired number of features with the best χ^2 scores are selected. The idea behind this is that if a feature is independent to the target, it is uniformative for classifying observations [56].

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{6.4}$$

where, E_i is the expected observations in class i if there was no relationship between the feature and target. O_i represents the number of observations in class i.

2. Recursive Feature Elimination: This technique as the name suggests recursively removes attributes and builds a model on those attributes that remain. Model accuracy is the metrics used to identify which attributes and combination of attributes contribute the most to predicting the target attribute.

3. Principal Component Analysis: Principal Component Analysis (PCA) uses linear algebra for transforming the dataset into a compressed form. This technique is termed as data reduction technique. One of the properties of PCA is that the number of dimensions or principal components can be chosen in the transformed result. This is a good technique if the dataset is highly dimensional.

In our research work, while experimenting with the dataset, we observed that the attack traffic shows lower source port number and lesser duration than the normal traffic. A recursive algorithm has been proposed that does the classification after univariate selection of features. However, the generic algorithm can be applied to any number of features to be merged based on their merging criterion which is explained by The Algorithm 3.

Algorithm 3 Algorithm-for-Decision-Area

```
Input: Features
Output: Decision Regions
  Feature Ranking(Features) ← $Univariate(\text{chi}^2)$
  i ← 1
F ← $FeatureRanking(Features)$
while $(F! = \emptyset) \&\& i \leq count(F)$ do
    $Des\_Reg$ = Create Decision Regions for feature $F_i$ of all datasets
    $F = F \setminus F_i$
    $i = i + 1$
    Algorithm-for-Decision-Area(F)
    return $Des\_Reg$
end while
```

6.3.2 Decision Region Analysis

Maximal cluster algorithm gave us the hosts that are clustered in a group. We have analyzed the tree regions of these hosts. Decision tree region analysis is important for the merging of model into one. The decision tree structures of the hosts that are found in a cluster are converted into decision regions. The best feature that is selected based on information gain is at the root of the tree. For example, for Dataset I four tree structures corresponding to hosts C, D, E and I have source port number as root node. The tree structure of Host M has duration as root node. Although the root node of Host M is different than the others, the cross-testing matrices demonstrate that it can also be used to detect the botnets from the normal for all five hosts. That means the botnets traffic that hit to these five hosts share the same characteristics: lower source port numbers and short duration.

After observing that botnet traffic demonstrates a lower source port number and shorter duration, algorithm 3 has been implemented for merging. The primary idea behind this to obtain the lower bound for the source port and duration in our case.

6.4 Merged Model with Instance Based Approach

The maximal clusters obtained from Bron Kerbosch algorithm are merged into one final cluster on the basis of the high detection rate and low false positive rate for attack traffic which is given by the formula of merged model of bigger maximal clusters, tested on each other.

$$DR = \frac{Number of Detected Attacks}{Total Number of Attack Connections} \tag{6.5}$$

$$FPR = \frac{Normal Detected as Attacks}{Total Number of Normal Connections} \tag{6.6}$$

This technique merges the merged models of these clusters into one model. Maximal cluster identification can help identify the anomaly behaviors and obtain a generic model which can detect anomalies without retraining. After the tree structure analysis of the hosts identified in the cluster, their trees have been converted to decision regions and merged using the Algorithm 3 Recursive-Algorithm defined in Section 6.3.1. The merged model detects the attack and normal traffic from the similar hosts well and most of the other hosts as well. The hosts whose traffic goes undetected by this merged model use the instance based learning approach to have their attack traffic detected. This eliminates the need of refreshing and retraining all of the data. Instance based learning is also called as memory based learning where instead of explicit generalization, new problem instances are compared with the instances seen in training. Hypotheses is constructed directly from the training instances themselves. So, it has the ability to adapt its model to previously unseen data. They may use a new instance and not the old instance. K-nearest neighbor algorithm is one of the instance based learning algorithm. The subset of or the training set is stored.

When predicting the class of the new instance, decision is made based on the computed distance or similarities between this instance and the training instance. The Algorithm 4 proposed combines both the merged model and instance based for detecting the attack. The hosts outside the cluster have a KNN. However, there are few hosts which are not in the cluster but their attack traffic is detected by the model. Since, they are outside the cluster, they should be tested with the KNN of the hosts outside the cluster. For testing with other datasets which is new a new set of data for the model, either the model or instance based detects an attack, that instance is considered attack. This is described in the Algorithm 4 below:

Algorithm 4 Attack Traffic Detecting Algorithm

```
Input: Merged Model of Cluster, Instance Based method of outside Cluster hosts,
  Test Dataset
Output: Optimum DR and FPR
  for each host $h$ in $Dataset$ do
    $DR$ = DR using Merged Model
    $FPR$ = FPR using Merged Model
  end for
  for each host $h$ in $Dataset$ do
    $DR$ = DR using Instance Based Approach
    $FPR$ = FPR using Instance Based Approach
  end for
  Consider the best results
```

6.5 Results

6.5.1 Dataset I

We tested the algorithm on three datasets where all experiments are done on the hosts for obtaining the model and host data is used for testing. Using the algorithm and experiments described in above sections, Dataset I which has 16 hosts, it is found that there are 11 clusters with 3 maximal cliques with maximum number of hosts. These clusters have their models obtained by merging the models of their hosts. That means if hosts C, D, E, I and M are in a cluster, the clusters model will be obtained by merging the model of its hosts. These clusters models are tested on every other cluster based on the detection rate and false positive rate for attack traffic as criterion. The hosts of clusters that give high detection rate and low false positive rate are merged again to obtain a final cluster. Tables 6.4 and 6.5 show the detection rate and false positive rate of merged model of clusters when tested on other maximal clusters.

Table 6.4. Detection Rate when Maximal Clusters Tested on each Other

Clusters	CDEIM	FJLNP	JLMNP
CDEIM	1	1	1
FJLNP	1	1	1
JLMNP	1	1	1

Table 6.5. False Positive Rate Maximal Clusters Tested on each Other

Clusters	CDEIM	FJLNP	JLMNP
CDEIM	0.1	0.03	0.04
FJLNP	0.1	0.004	0.005
JLMNP	0.1	0.004	0.005

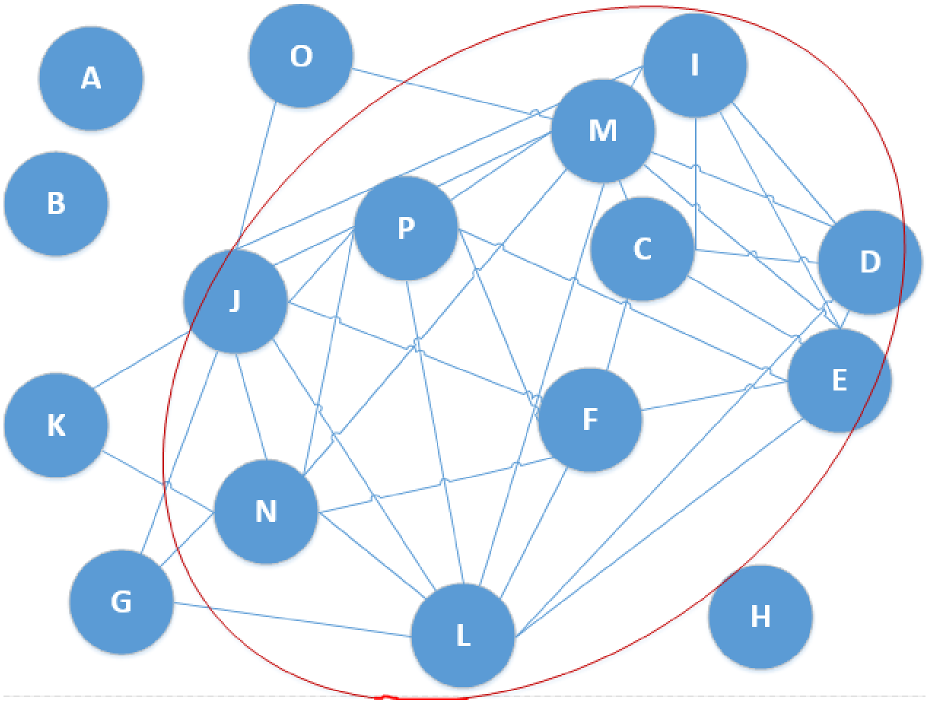


Fig. 6.2. Merged Cluster for Dataset I

Based on high detection rate and low false positive rate for attack traffic, it can be inferred that their performance is good with respect to detecting attack traffic. As seen from these tables, the hosts C, D, E, I, M, F, L, P, J, N can be put into one cluster which is shown in Figure 6.2. The most effective way to integrate the decision tree models of these hosts have been found to be converting the decision trees of each host to decision region and merging recursively. According to Algorithm 4, the hosts outside the cluster will utilize the instance based learning, in our case we used KNN. The average detection rate and false positive rate of all hosts obtained using the model are shown in the column Average which is the macro average. They are calculated using the Formula 6.7 and 6.8. Results are tabulated in Table 6.6. Here n is the number of hosts of the respective dataset. As expected the method performs well on itself.

$$AvgDR = \frac{\sum_{n=1}^{n} DR_n}{n} \tag{6.7}$$

$$AvgFPR = \frac{\sum_{n=1}^{n} FPR_n}{n} \tag{6.8}$$

Table 6.6. Model of Dataset I Tested on Itself

Model Based Approach											Instance Based Approach						Average
Hosts	C	D	E	I	M	F	J	L	N	P	A	B	O	K	G	H	All
DR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Inter-Dataset Testing using Model of Dataset I

In this section, the proposed method for Dataset I has been tested on the other datasets considered. They are Datasets II and III which belong to Neris and Virut bot category respectively. For Dataset II all the hosts were detected by the merged cluster model of Dataset I which shows that our method performed well with the unseen data. For Dataset III, few hosts were not detected by merged model of Dataset I, so according to our Algorithm 4, they are should ideally be detected by the KNN model of Dataset I. The results are shown in Table 6.8.

Table 6.7. Model of Dataset I Tested on Dataset II

Merged Model Based																						
Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	Avg
DR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Table 6.8. Model of Dataset I Tested on Dataset III

Model Based Approach														Instance Based Approach					Average
Hosts	A	B	D	G	H	I	J	L	M	N	P	Q	R	C	E	F	K	O	All
DR	1	0.85	0.6	1	1	1	1	1	1	1	1	1	1	1	0.79	1	1	1	**0.95**
FPR	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	**0.03**

Comparison with Traditional Methods

Comparing the results achieved on unseen data using the proposed method with respect to the traditional methods including Naive Bayes, Logistic Regression and Random Forest demonstrate that our method is more robust. Working with our subsets, a Gaussian distribution [57] is assumed because the probabilities for input variables using the Gaussian Probability Density Function can be estimated easily. NB and LR performance is low as compared to Random Forest. Our approach has results that are comparable to RF. Figures 6.3 and 6.4 show the comparison of our model with NB and LR respectively.

Figure 6.5 shows the comparison with RF. Although the FPR is lower in both the cases, the detection rate using our technique is much better than using the random forest. The random forest model of Dataset III (Figure 6.9, Section 6.5.3) on itself shows good results but it appears to be overfitting the data, that's why the retrained model of Dataset I didn't give acceptable detection rate here. Thus, our method solves the overfitting problem in addition to reducing the retraining efforts.

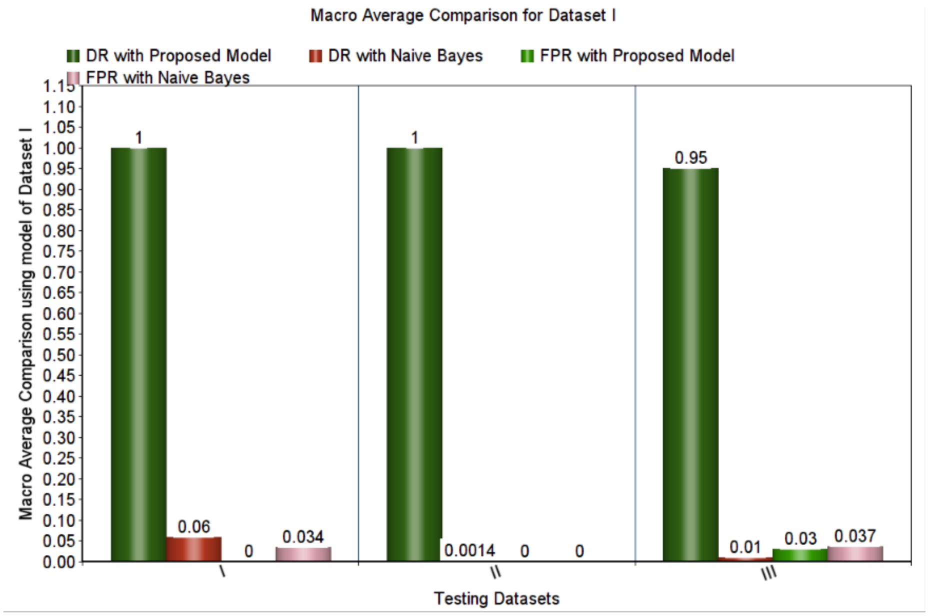


Fig. 6.3. Macro Average Comparison for Dataset I with NB

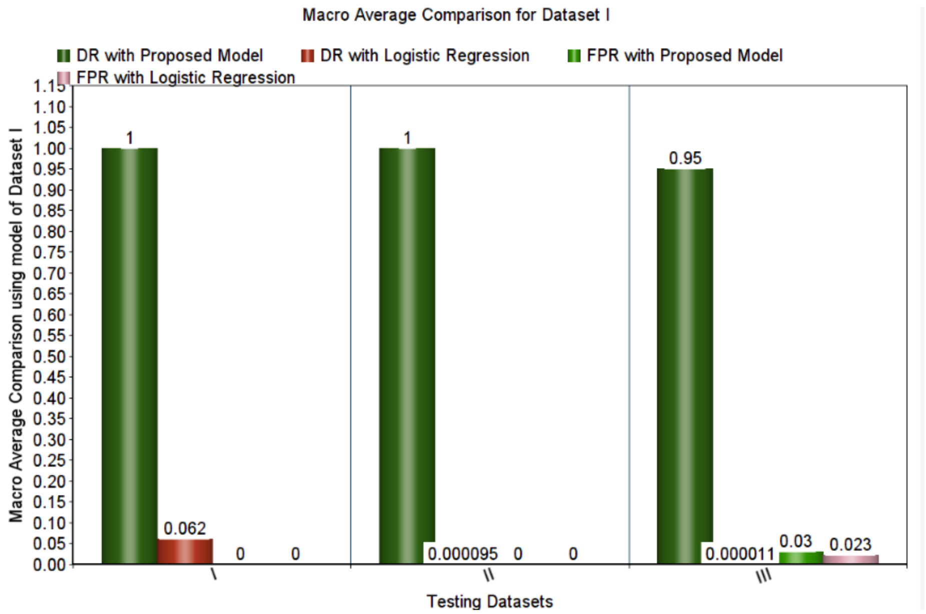


Fig. 6.4. Macro Average Comparison for Dataset I with LR

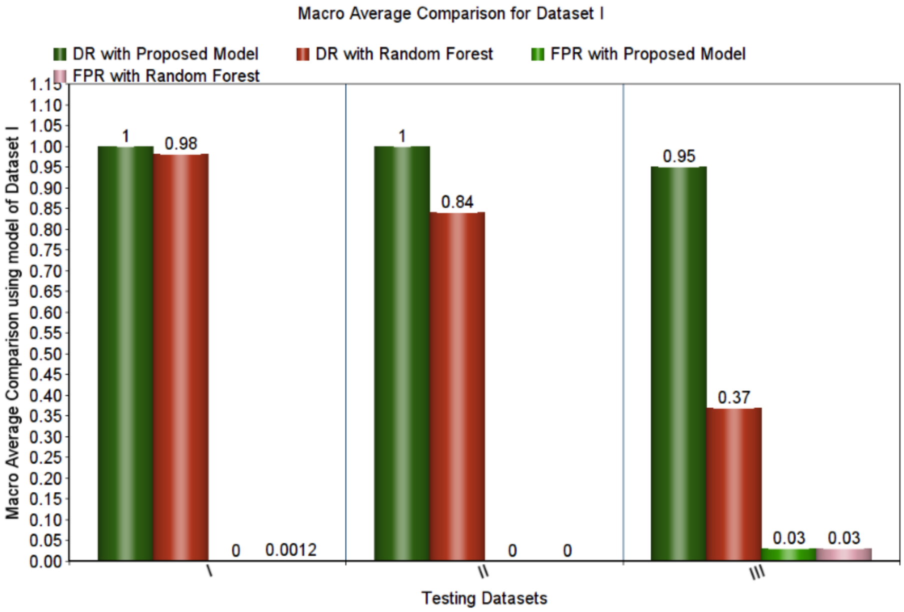


Fig. 6.5. Macro Average Comparison for Dataset I with RF

6.5.2 Dataset II

Similar approach has been tested on Dataset II. From the previous results section for Dataset II, there are four maximal clusters with maximum hosts, BDOSU, ENO, HNR and RNE. Each of these clusters have a merged model that is tested on each other for detection rate and false positive rate. The Tables 6.9 and 6.10 demonstrate these results.

Table 6.9. Dataset II - DR when Maximal Clusters Tested on each Other

Clusters	BDOSU	ENO	HNR	ENR
BDOSU	1	1	1	1
ENO	1	1	1	1
HNR	1	1	1	1
ENT	1	1	1	1

Table 6.10. Dataset II - FPR when Maximal Clusters Tested on each Other

Clusters	BDOSU	ENO	HNR	ENR
BDOSU	0	0.9	0.9	0.9
ENO	0	0	0	0
HNR	0	0	0	0
ENR	0	0	0	0

Based on the results tabulated these tables, the big cluster obtained is by merging ENO, HNR and ENR. The final cluster E, N, O, H, R is highlighted in Figure 6.6.

As described by our method, the hosts E, N, O, H, R have an integrated model. However, the hosts outside have instance based learning methods. The results are tabulated in Table 6.11. The point to be noted down here is that although the integrated model of cluster could detect all the traffic for its own hosts (even outside

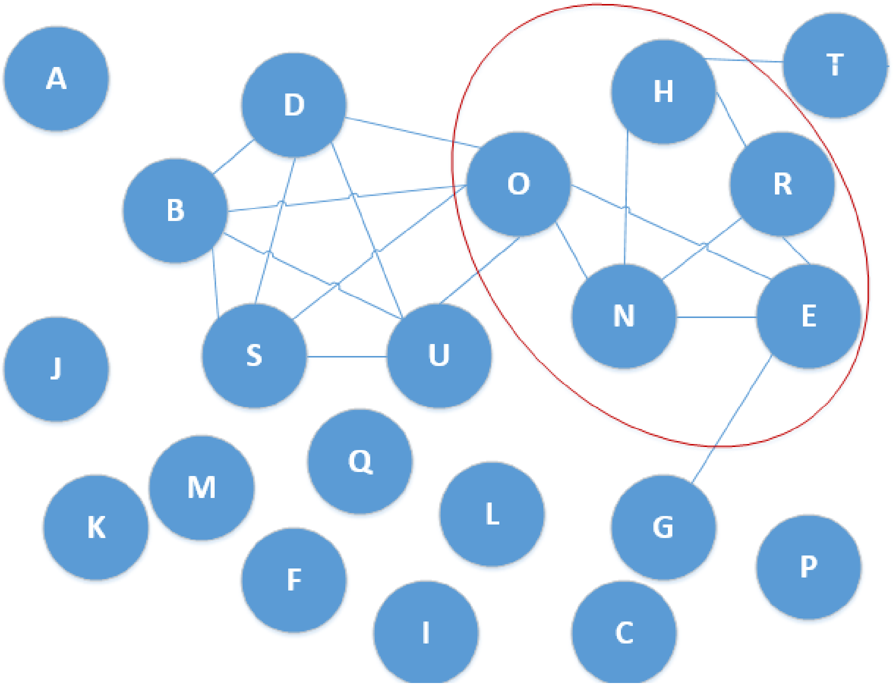


Fig. 6.6. Merged Cluster for Dataset II

the cluster) but according to our proposed algorithm, the hosts outside the cluster should have an KNN towards detecting the attack.

Table 6.11. Model of Dataset II Tested on Itself

Merged Model Based						Instance Based Approach																Average
Hosts	E	N	O	H	R	A	B	C	D	F	G	I	J	K	L	M	P	Q	S	T	U	All
DR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Inter-Dataset Testing using Model of Dataset II

The merged model of Dataset II is tested on each host of other datasets. This is demonstrated in Table 6.12 when tested on Dataset I.

Table 6.12. Model of Dataset II Tested on Dataset I

Merged Model Based Approach																Instance Based Approach	Average
Hosts	A	C	D	E	F	G	H	I	J	K	L	M	N	O	P	B	All
DR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.7	**0.97**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9	**0.05**

The merged model of Dataset II couldn't detect the attack traffic in one of the hosts of Dataset I. This host when tested with the instance based of Dataset I gave a little higher DR of 0.7 but unfortunately the false positive rate is still high. During analysis of this host, it is shown that this host data has a different pattern for its features. It might be possible that Host B of Dataset I is under an attack that belongs to a different category than the hosts whose attack traffic are detected. It is inferred that, if a more robust merging criterion is applied, our method may become more efficient and detect attacks from such eccentric data. Interesting point to note down here is as shown in the Appendix Table A.11 even the retrained random forest model of Dataset II couldn't detect the attack traffic of Dataset I Host B.

Similarly, the merged model of Dataset II is tested on the Dataset III. Table 6.13 shows the result using the model based and instance based approach.

Comparison with Traditional Methods

Results comparisons between our proposed model and three aforementioned approaches for Dataset II are done. Since the results are pretty much comparable to RF technique, comparison with other two are shown in Figures A.1 and A.2 in the

Table 6.13. Model of Dataset II Tested on Dataset III

Merged Model Based Approach													Instance Based Approach						Average
Hosts	A	B	G	H	I	J	L	M	N	P	Q	R	C	D	E	F	K	O	All
DR	1	0.89	1	1	1	1	1	1	1	1	1	1	0.74	1	0.92	1	0	0.5	**0.89**
FPR	0.004	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0.5	0	0	**0.03**

Appendix. Our model performed better on the unseen Dataset III giving DR of 0.89 as compared to 0.81 using traditional method of random forest.

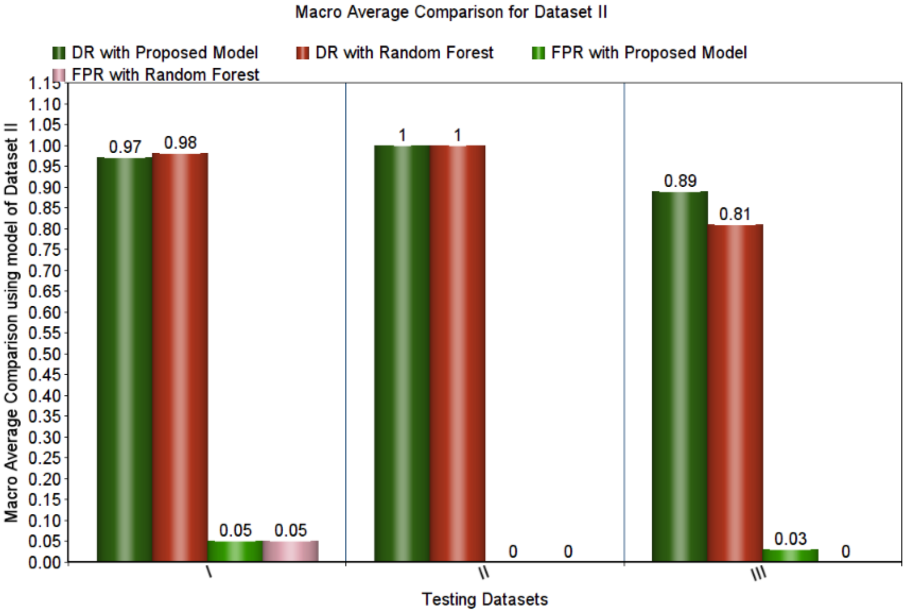


Fig. 6.7. Macro Average Comparison for Dataset II with RF

6.5.3 Dataset III

Biggest maximal clusters are merged to obtain models for them. They are tested for detection rate and false positive rate as described in above section.

Table 6.14. Dataset III - DR when Maximal Clusters Tested on each Other

Clusters	GHIMPQ	GIMOP
GHIMPQ	1	1
GIMOP	1	1

Table 6.15. Dataset III - FPR when Maximal Clusters Tested on each Other

Clusters	GHIMPQ	GIMOP
GHIMPQ	0	0
GIMOP	0	0

The Tables 6.14 and 6.15 show the detection rate and false positive rates respectively with two clusters, GHIMPQ and GIMOP. Based on the results, the merged cluster is using hosts G, H, I, M, P, O, Q which is circled in Figure 6.8. The merged model is tested on each of the host of Dataset III shown in Table 6.16.

Table 6.16. Model of Dataset III Tested on Itself

Model Based Approach								Instance Based Approach											Average
Hosts	G	H	I	M	O	P	Q	A	B	C	D	E	F	J	K	L	N	R	All
DR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

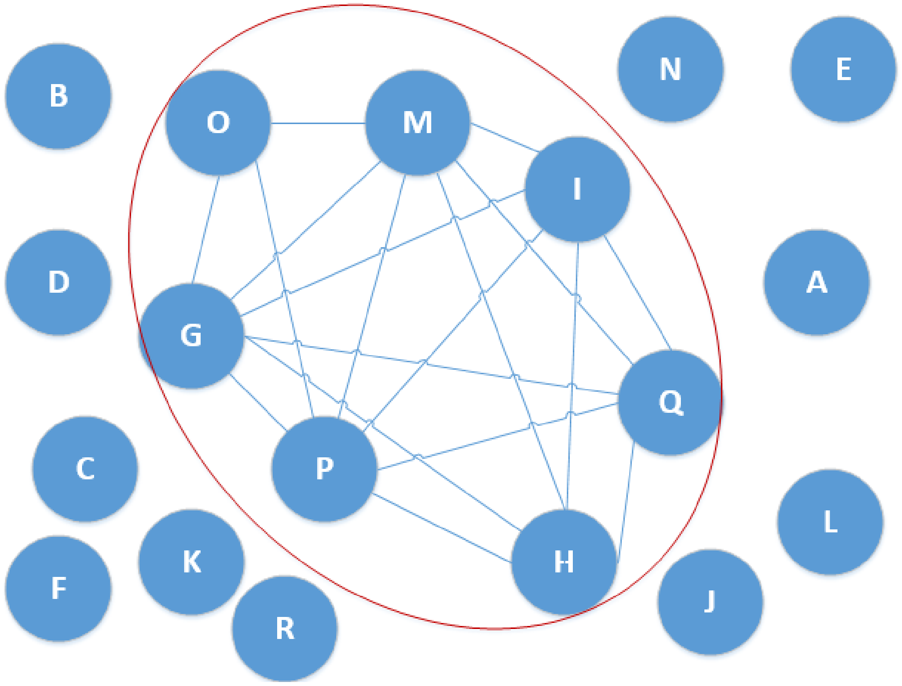


Fig. 6.8. Merged Cluster for Dataset III

Inter-Dataset Testing using Model of Dataset III

Table 6.17 shows the model of Dataset III tested on each host of Dataset I using the aforementioned techniques. The explanations given in subsection Inter-Dataset Testing using model of Dataset II is applicable here contributing to the high false positive rate for Host B of Dataset I.

Table 6.17. Model of Dataset III Tested on Dataset I

Merged Model Approach																Instance Based Approach	Average
Hosts	A	C	D	E	F	G	H	I	J	K	L	M	N	O	P	B	All
DR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.6	**0.97**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9	**0.05**

Table 6.18 shows the results when the merged model of Dataset III is tested on the hosts of Dataset II.

Table 6.18. Model of Dataset III Tested on Dataset II

Merged Model Based Approach																						Average
Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	All
DR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Comparison with Traditional Methods

The results achieved for both the techniques i.e our method and random forests are pretty much comparable, shown in Figure 6.9. Results comparison with Naive Bayes and Logistic Regression are shown in Figures A.3 and A.4 in the Appendix.

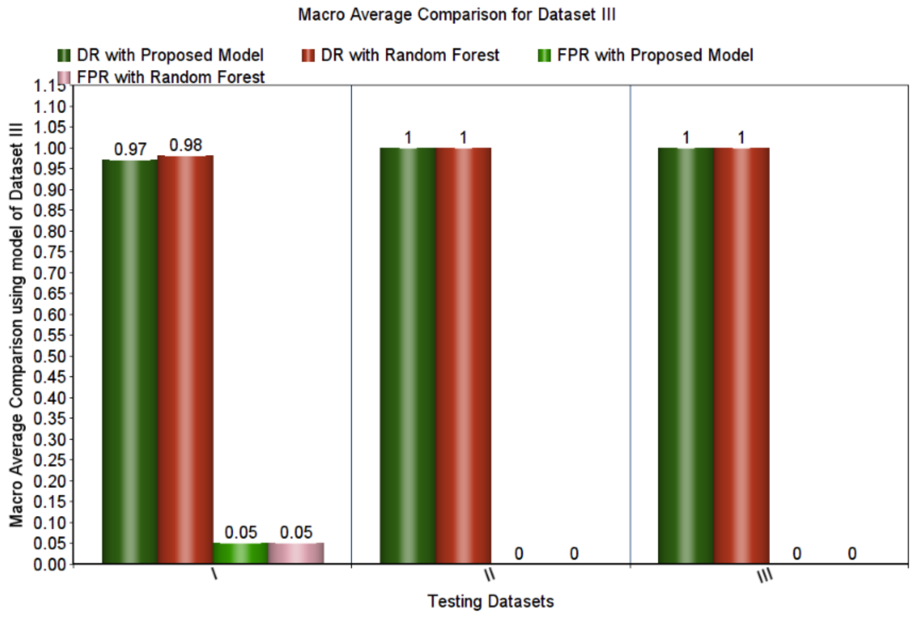


Fig. 6.9. Macro Average Comparison for Dataset III with RF

6.5.4 Results Analysis

The macro average comparison among various models for the three datasets have been shown in above sections. Although the macro average results are acceptable, if considered on individual hosts level there are few hosts where attack traffic is not detected by any of the inter-dataset models. These are the special cases discussed here.

Dataset I: Host B

The attack pattern of Dataset I Host B appears to be different than the other detected attacks. The attack traffic of this host is not detected by most of the inter-dataset models except the Naive Bayes model of Dataset II but with lower detection rate. The performance of various models is shown in Figure 6.10. As seen in this figure, the FPR is very high in most of the cases.

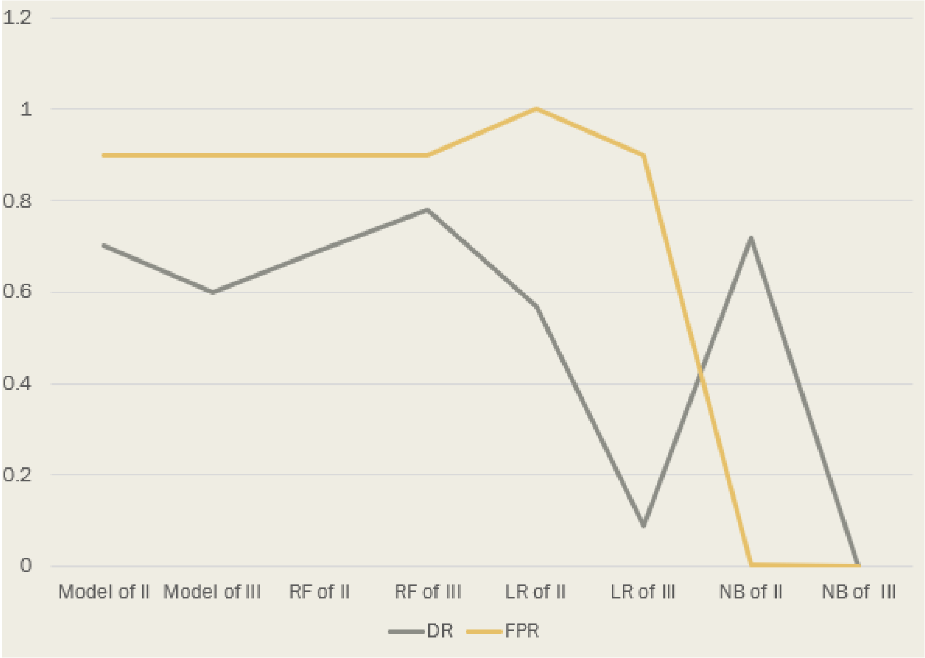


Fig. 6.10. Various Inter-Dataset Models Tested on Host B of Dataset I

The scatter plots for this host has been drawn which shows the relationship between two variables in two dimensions. The plots show the pairwise relationships from different perspectives and can help in analyzing and visualizing the bot pattern. Here diagonal represents histogram of each attribute.

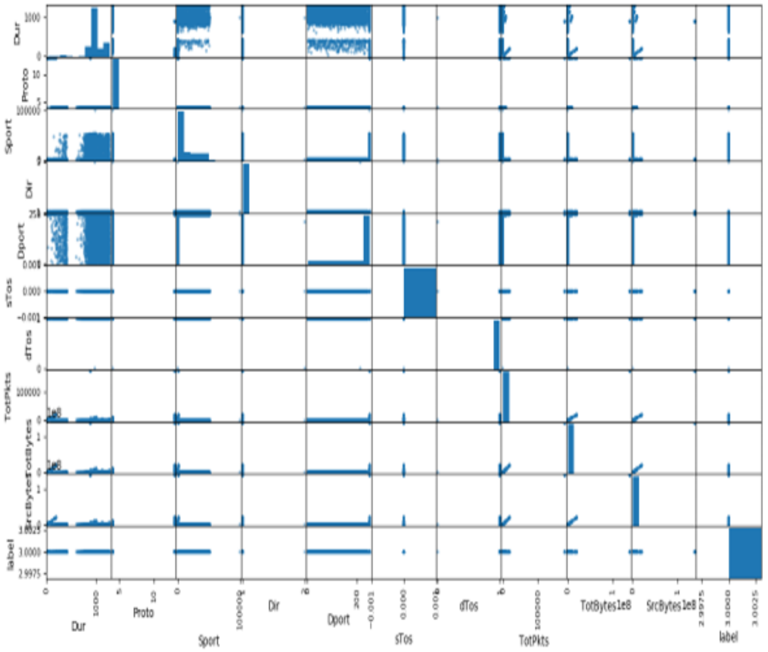


Fig. 6.11. Scatter Plot of Host B Dataset I

Dataset II: Host N

Various models give varying results for Host N of Dataset II. This is shown in Figure 6.12 where using the proposed approaches of Datasets I and III give high DR and low FPR, whereas traditional retrained inter-dataset models give very low DR. This host has higher number of normal and botnet instances, the inter-dataset models using our approach is robust enough to detect attack traffic of unseen data.

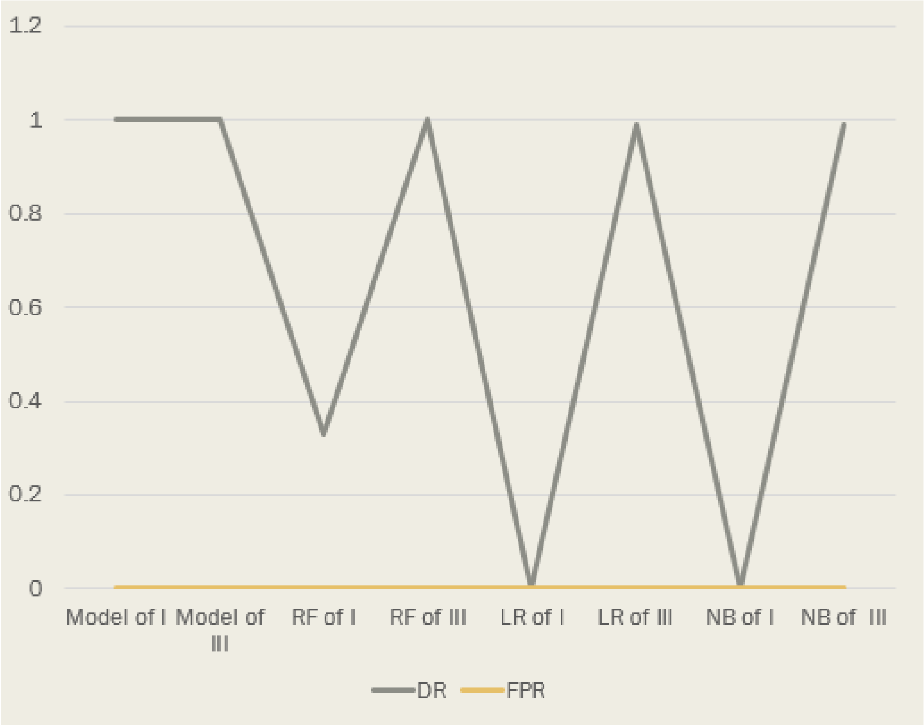


Fig. 6.12. Various Inter-Dataset Models Tested on Host N of Dataset II

The scatter plot matrix for this host is pretty similar to that of Host A of Dataset III discussed in below section.

Dataset III: Host A

Similar to Host N of Dataset II, there are few inter-dataset traditional models that didn't perform well for this host of Dataset III. These models are of Dataset I as shown in Figure 6.13. The scatter plot matrix of this host and Host N of Dataset II discussed in above section are pretty similar. They are shown in Figures 6.14 and 6.15.

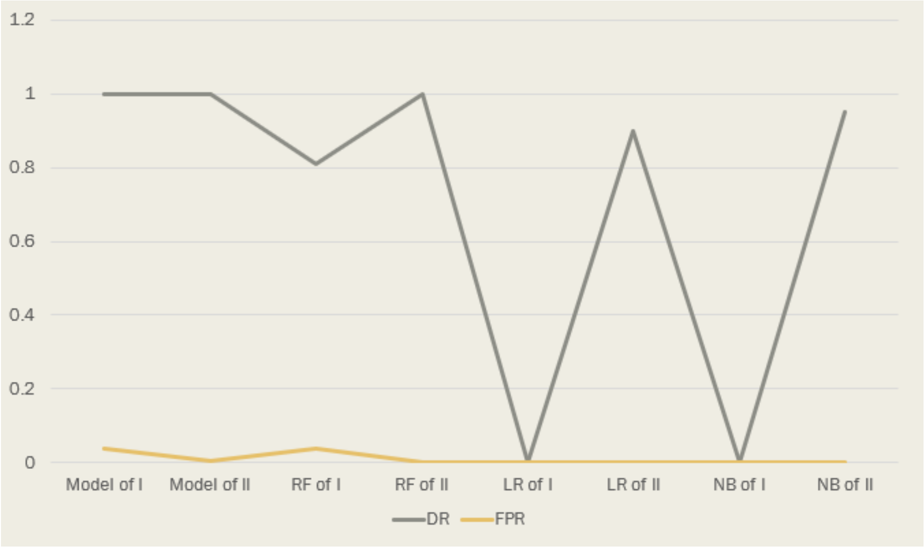


Fig. 6.13. Various Inter-Dataset Models Tested on Host A of Dataset III

Pattern of Dataset I Host B appears to be different than the others. So it is concluded that majority learning is from Botnet behavior of II III. It is possible that the undetected attack patterns are for DDOS or US bot category. The attack data detected for Dataset I appears to be of IRC bot category which is the bot category covered in Dataset I and II. Table 6.19 shows the botnets covered in our experiments. However, there are hosts like Host C and D of Dataset III where our approach outperformed traditional way of random forests as shown in Figures 6.16 (a) and (b).

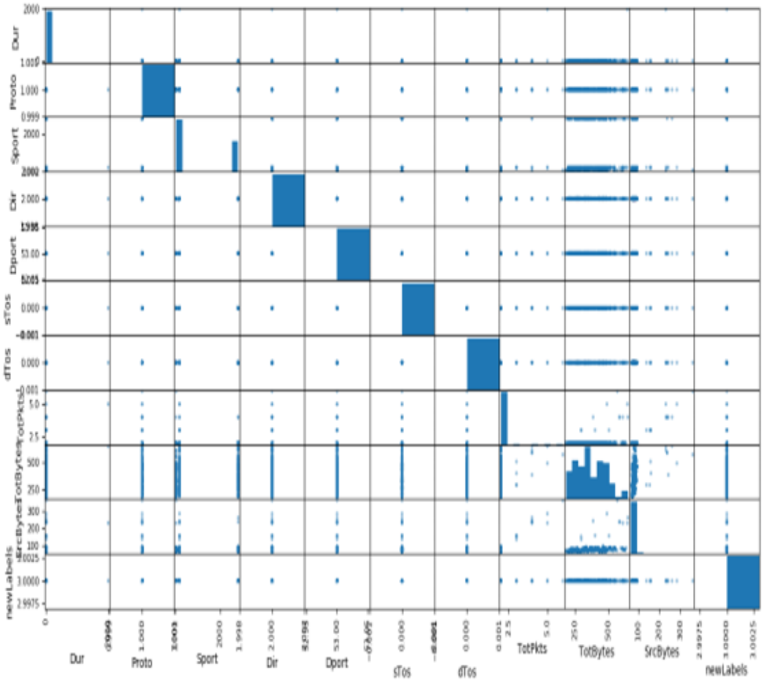


Fig. 6.14. Scatter Plot of Host N Dataset II

Table 6.19. Botnets Included in Datasets

Dataset I	Dataset II	Dataset III
IRC, DDOS, US	IRC, SPAM, CF	SPAM, PS, HTTP

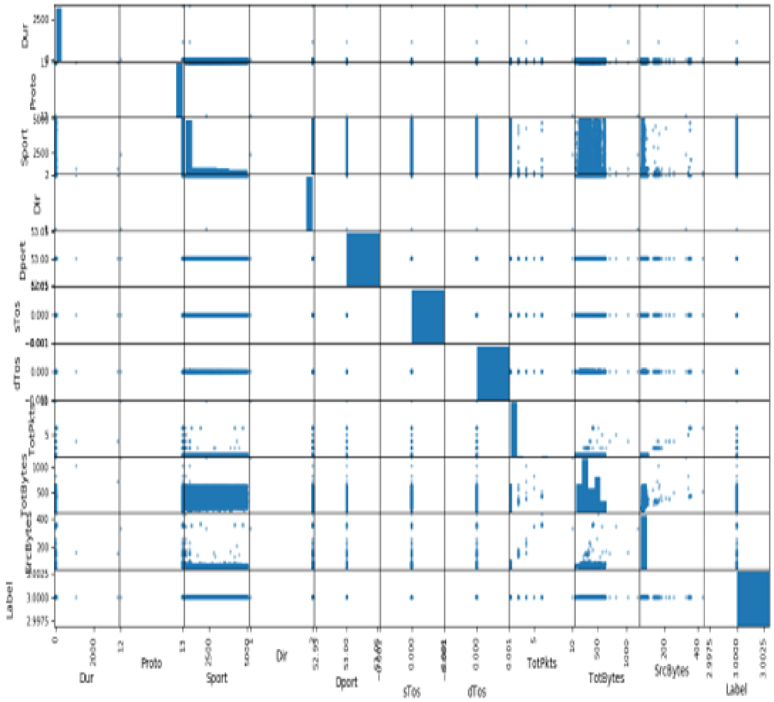
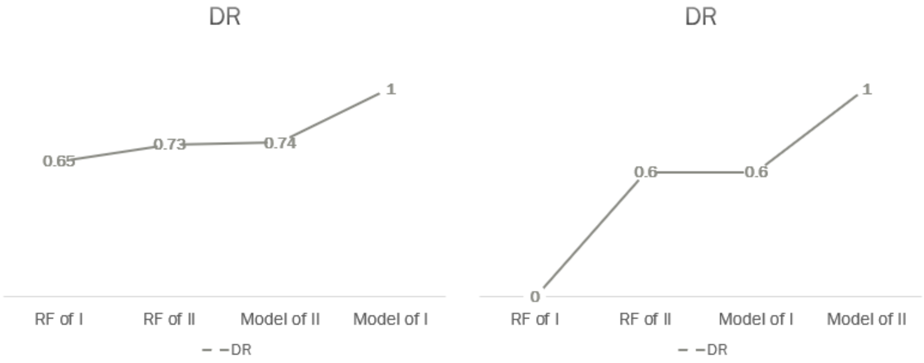


Fig. 6.15. Scatter Plot of Host A Dataset III



(a)

(b)

Fig. 6.16. Detection Rate for Host C and D of Dataset III

7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this research, we proposed and implemented an innovative way to cluster hosts utilizing machine learning models and data fusion algorithms with regards to intrusion detection. In a network with massive network flow data, clustering hosts that have similar behavior with regards to normal and attack behaviors will be very beneficial for detecting unusual or anomalous behavior. It can cluster a group of hosts with anomalous behavior and identify the spread of the attacking in fast manner. If the network flow behaviors are different among the host clusters in a network, the generated models of different host clusters can be integrated to become a more robust learning model that can detect different anomalous behaviors. The different generated models from different host clusters have been integrated to test whether the integrated model is robust compared to generating a new learning model based on the massive network flows over the whole network.

We have proposed a method to merge the decision regions of the similar hosts in a recursive way for the best features based on the information gain ranking. The intuition behind merging the decision regions of features selected after chi^2 feature selection is to avoid retraining. In addition to retraining, it proves to avoid overfitting when tested on new data.

The results obtained while testing the data with the integrated model are compared with the results obtained by traditional way of training a machine learning algorithm on network testing. It is observed that our proposed method gives much better detection rate and lower false positive rates when tested on unseen data. Our work has emphasized on the importance of finding similar hosts, so that they can be put into a cluster and the model of the individual hosts in a particular cluster can be integrated with hosts outside the cluster following an instance based learning. The integrated method considers both the merged and instance based, the one which gives the better detection rate and false positive rate for the host is considered.

7.2 Future Work

7.2.1 More Robust Method

In our research, we have tested the results with both the merged and instance based methods, the considered result is the one with optimum DR and FPR. By optimum we mean if the DR is good, FPR shouldn't be too high. DR with a lower value can be preferred but the FPR shouldn't be too high.

In order to make our proposed model more robust, a better approach could be if either the merged model or the instance based KNN detects an instance as attack, it will be considered attack. It would be better if instead of selecting the optimum DR and FPR for the hosts, each instance is tested with both the approaches and if either of them detects it as attack, it is considered attack.

7.2.2 Integrating the Models of Hosts in Cluster

A lot of more ways can be explored for merging the models of cluster hosts. We integrated the decision regions based on our observation that attack traffic possess lower Source Port numbers and shorter duration in our case. Probability measures may be deployed to integrate the decision regions. That means if decision region for a feature of model A shows attack traffic has value less than a value 'a' (let's suppose) and the decision region for the same feature for model B has attack traffic below a value 'b', the decision making node has the value either 'a' or 'b', the one having more probability. Different machine learning approaches other than decision trees may be explored for integrating

7.2.3 Dynamic Approach Towards Intrusion Detection

In our research, the proposed method is considered as misuse detection because the attacks are known. In future, if anomaly detection techniques are applied, new attack traffic can also be detected. K means clustering algorithm may be explored for clustering the normal traffic using some distance measure.

REFERENCES

REFERENCES

[1] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[2] C. H. Rowland, "Intrusion detection system," in *US Patent 6, 405, 318.* Google Patents, June 11, 2002.

[3] M. Sheikhan, Z. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size rnn based on feature grouping," *Neural Computing and Applications*, vol. 21, no. 6, pp. 1185–1190, 2012.

[4] H. Akramifard, L. M. Khanli, M. Balafar, and R. Davtalab, "Intrusion detection in the cloud environment using multi-level fuzzy neural networks," in *Proceedings of the International Conference on Security and Management (SAM).* The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015, p. 75.

[5] "https://securityintelligence.com/applying-machine-learning-to-improve-your-intrusion-detection-system/," Applying Machine Learning to Improve Your Intrusion Detection System, [Online]. Last Date Accessed: 12/1/2018.

[6] Y. Gong, S. Mabu, C. Chen, Y. Wang, and K. Hirasawa, "Intrusion detection system combining misuse detection and anomaly detection using genetic network programming," in *ICCAS-SICE, 2009.* IEEE, 2009, pp. 3463–3467.

[7] A. Tajbakhsh, M. Rahmati, and A. Mirzaei, "Intrusion detection using fuzzy association rules," *Applied Soft Computing*, vol. 9, no. 2, pp. 462–469, 2009.

[8] T. Mehmood and H. B. M. Rais, "Machine learning algorithms in context of intrusion detection," in *Computer and Information Sciences (ICCOINS), 2016 3rd International Conference.* IEEE, 2016, pp. 369–373.

[9] "http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html," KDD Cup 1999 Data, [Online]. Last Date Accessed: 12/1/2018.

[10] K. Park, Y. Song, and Y.-G. Cheong, "Classification of attack types for intrusion detection systems using a machine learning algorithm," in *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService).* IEEE, 2018, pp. 282–286.

[11] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation," in *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security.* ACM, 2011, pp. 29–36.

[12] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *Journal of network and computer applications*, vol. 36, no. 1, pp. 42–57, 2013.

[13] A. Patel, M. Taghavi, K. Bakhtiyari, and J. C. JúNior, "An intrusion detection and prevention system in cloud computing: A systematic review," *Journal of network and computer applications*, vol. 36, no. 1, pp. 25–41, 2013.

[14] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms," in *Computer and Communications (ICCC), 2016 2nd IEEE International Conference*. IEEE, 2016, pp. 2451–2455.

[15] S. Zanero and G. Serazzi, "Unsupervised learning algorithms for intrusion detection," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*. IEEE, 2008, pp. 1043–1048.

[16] "Model-based machine learning christopher m. bishop microsoft research," Cambridge CB3 0FB, UK, [Online]. Last Date Accessed: 12/1/2018.

[17] C. Gao, H. Sun, T. Wang, M. Tang, N. I. Bohnen, M. L. Müller, T. Herman, N. Giladi, A. Kalinin, C. Spino *et al.*, "Model-based and model-free machine learning techniques for diagnostic prediction and classification of clinical outcomes in parkinsons disease," *Scientific reports*, vol. 8, no. 1, p. 7129, 2018.

[18] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[19] "https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f," One Hot Encode, [Online]. Last Date Accessed: 12/1/2018.

[20] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 420–424.

[21] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Computing and Applications*, vol. 28, no. 1, pp. 1051–1058, 2017.

[22] S. Drazin and M. Montag, "Decision tree analysis using weka," *Machine Learning-Project II, University of Miami*, pp. 1–3, 2012.

[23] A. I. Saleh, F. M. Talaat, and L. M. Labib, "A hybrid intrusion detection system (hids) based on prioritized k-nearest neighbors and optimized svm classifiers," *Artificial Intelligence Review*, pp. 1–41, 2017.

[24] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "Cann: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-based systems*, vol. 78, pp. 13–21, 2015.

[25] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference*, vol. 2. IEEE, 2002, pp. 1702–1707.

[26] L. Khan, M. Awad, and B. Thuraisingham, “A new intrusion detection system using support vector machines and hierarchical clustering,” *The VLDB journal*, vol. 16, no. 4, pp. 507–521, 2007.

[27] L. Wu, S. Li, X. Gan *et al.*, “Network anomaly intrusion detection cvm model based on pls feature extraction,” *Control and Decision. China*, vol. 32, pp. 755–758, 2017.

[28] C.-C. Chang and C.-J. Lin, “Libsvm: a library for support vector machines,” *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[29] S. Garcia, “http://mcfp.weebly.com/the-ctu-13-dataseta-labeled-dataset-with-botnet-normal-and-background-traffic.html.” The CTU-13 dataset. a labeled dataset with botnet, normal and background traffic, [Online]. Last Date Accessed: 12/1/2018.

[30] “https://en.wikipedia.org/wiki/irc_bot,” IRC Bot, [Online]. Last Date Accessed: 12/1/2018.

[31] S. Wei, J. Mirkovic, and E. Kissel, “Profiling and clustering internet hosts.” *DMIN*, vol. 6, pp. 269–75, 2006.

[32] K. Xu, F. Wang, and L. Gu, “Network-aware behavior clustering of internet end hosts,” in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 2078–2086.

[33] B. Li, M. H. Gunes, G. Bebis, and J. Springer, “A supervised machine learning approach to classify host roles on line using sflow,” in *Proceedings of the first edition workshop on High performance and programmable networking*. ACM, 2013, pp. 53–60.

[34] N. C. S. Iyengar, A. Banerjee, and G. Ganapathy, “A fuzzy logic based defense mechanism against distributed denial of services attack in cloud environment,” *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 6, no. 3, 2014.

[35] M. Sheikhan and Z. Jadidi, “Flow-based anomaly detection in high-speed links using modified gsa-optimized neural network,” *Neural Computing and Applications*, vol. 24, no. 3-4, pp. 599–611, 2014.

[36] S. Mukkamala, A. Sung, and B. Ribeiro, “Model selection for kernel based intrusion detection systems,” in *Adaptive and Natural Computing Algorithms*. Springer, 2005, pp. 458–461.

[37] P. Casas, J. Mazel, and P. Owezarski, “Unsupervised network intrusion detection systems: Detecting the unknown without knowledge,” *Computer Communications*, vol. 35, no. 7, pp. 772–783, 2012.

[38] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, “On the capability of an som based intrusion detection system,” in *Neural Networks, 2003. Proceedings of the International Joint Conference*, vol. 3. IEEE, 2003, pp. 1808–1813.

[39] Y. Guan, A. A. Ghorbani, and N. Belacel, “Y-means: A clustering method for intrusion detection,” in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference*, vol. 2. IEEE, 2003, pp. 1083–1086.

[40] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[41] J. Yang, Z. Ye, L. Yan, W. Gu, and R. Wang, “Modified naive bayes algorithm for network intrusion detection based on artificial bee colony algorithm,” in *2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*. IEEE, 2018, pp. 35–40.

[42] M. Panda and M. R. Patra, “Network intrusion detection using naive bayes,” *International journal of computer science and network security*, vol. 7, no. 12, pp. 258–263, 2007.

[43] S. Mukherjee and N. Sharma, “Intrusion detection using naive bayes classifier with feature reduction,” *Procedia Technology*, vol. 4, pp. 119–128, 2012.

[44] “https://en.wikipedia.org/wiki/bayes_theorem,” Bayes Theorem, [Online]. Last Date Accessed: 12/1/2018.

[45] G. D’Agostini, “A multidimensional unfolding method based on bayes’ theorem,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 362, no. 2-3, pp. 487–498, 1995.

[46] Y. Wang, “A multinomial logistic regression modeling approach for anomaly intrusion detection,” *Computers & Security*, vol. 24, no. 8, pp. 662–674, 2005.

[47] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, “Intrusion detection by machine learning: A review,” *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[48] “https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html,” Logistic Regression, [Online]. Last Date Accessed: 12/1/2018.

[49] J. Friedman, T. Hastie, R. Tibshirani *et al.*, “Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors),” *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[50] N. Farnaaz and M. Jabbar, “Random forest modeling for network intrusion detection system,” *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[51] Y. Chang, W. Li, and Z. Yang, “Network intrusion detection based on random forest and support vector machine,” in *Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), 2017 IEEE International Conference*, vol. 1. IEEE, 2017, pp. 635–638.

[52] A. Andrzejak, F. Langner, and S. Zabala, “Interpretable models from distributed data via merging of decision trees,” in *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium*. IEEE, 2013, pp. 1–9.

[53] P. Strecht, “A survey of merging decision trees data mining approaches,” in *Proc. 10th Doctoral Symposium in Informatics Engineering*, 2015, pp. 36–47.

[54] J. Brownlee, “Machine learning mastery with python,” *Machine Learning Mastery Pty Ltd*, pp. 100–120, 2016.

[55] “https://nlp.stanford.edu/ir-book/html/htmledition/feature-selectionchi2-feature-selection-1.html,” Chi Square Feature Selection, [Online]. Last Date Accessed: 12/1/2018.

[56] “https://chrisalbon.com/machine_learning/feature_selection/chi-squared-for-feature-selection,” Chi Square Fearure Selection, [Online]. Last Date Accessed: 12/1/2018.

[57] G. H. John and P. Langley, “Estimating continuous distributions in bayesian classifiers,” in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.

APPENDIX

A. APPENDIX

Table A.1. Instances Count for Dataset I

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Total
Normal	9045	471	43	42	44	55	33	26	30	48	43	31	35	44	46	41	**10077**
Botnet	153	9848	1	1	1	1	1	4	3	2	6	1	3	2	4	2	**10033**
Total	9198	10319	44	43	45	56	34	30	33	50	49	32	38	46	50	43	**20110**

Table A.2. Instances Count for Dataset II

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	Total
Normal	1	10	1	3	243	1	15	12	1	23	1	5	295	5893	9	3	1	8	3	6	8	**6300**
Botnet	5	2	5	3	2	30	2	8	1	1	15	1	1	785	3	1	1	8	4	5	3	**886**
Total	6	12	6	6	245	31	17	20	2	24	16	6	296	6678	12	4	2	16	7	11	11	**7186**

Table A.3. Instances Count for Dataset III

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	Total
Normal	20376	18	18	3	1	2	5	30	39	21	16	50	19	26	14	52	58	6	**20991**
Botnet	7518	35	23	10	71	5	1	3	1	1	1	1	2	1	2	1	1	1	**7678**
Total	27894	53	41	13	72	7	6	33	40	22	17	51	21	27	16	53	59	7	**28669**

Table A.4. Dataset II Cross-testing F1-Measure of Decision Tree

Hosts	B	D	E	G	H	N	O	R	S	T	U
B	1	1	0.01	1	0.37	0.17	1	0.42	1	0.4	1
D	1	1	0.01	1	0.37	0.17	1	0.42	1	0.4	1
E	1	1	1	1	1	1	1	1	1	1	1
G	0.4	0	1	0.42	0.28	0.45	0.4	0.2	0	0.25	0.34
H	1	1	1	1	1	1	1	1	1	1	1
N	1	1	1	1	1	1	1	1	1	1	1
O	1	1	1	1	1	1	1	1	1	1	1
R	1	1	1	1	1	1	1	1	1	1	1
S	1	1	0.01	1	0.37	0.17	1	0.73	1	0.4	1
T	0.25	1	1	0.2	1	0.45	0.25	0.42	1	1	0.34
U	1	1	0.01	1	0.37	0.17	1	0.73	1	0.4	1

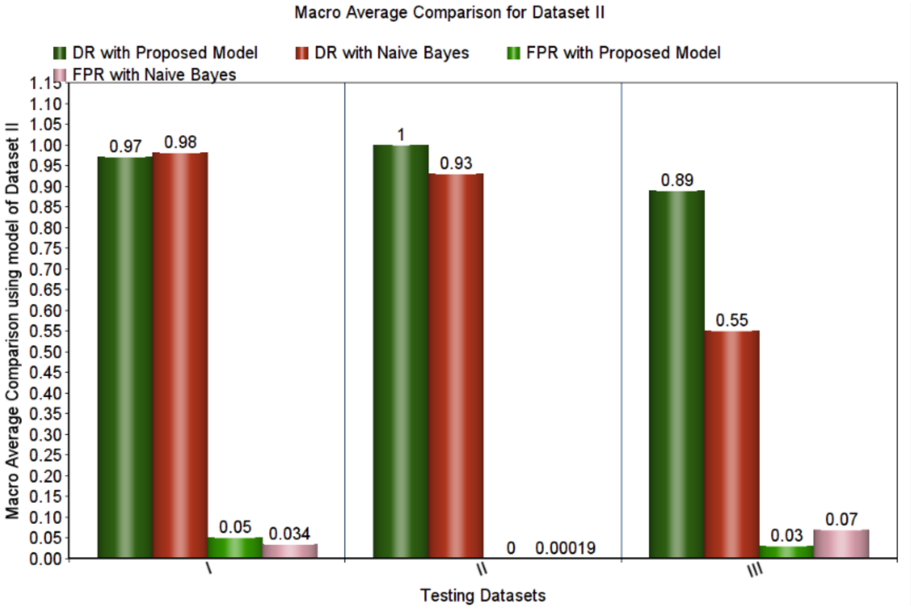


Fig. A.1. Macro Average Comparison for Dataset II with NB

Table A.5. Dataset II Cross-testing F1-Measure of KNN

Hosts	B	D	E	G	H	N	O	R	S	T	U
B	1	1	1	1	1	1	1	1	1	1	1
D	1	1	1	1	1	1	1	1	1	1	1
E	1	1	1	1	1	1	1	1	1	1	1
G	1	1	1	1	1	1	1	1	1	1	1
H	1	1	1	1	1	1	1	1	1	1	1
N	1	1	1	1	1	1	1	1	1	1	1
O	1	1	1	1	1	1	1	1	1	1	1
R	1	1	1	1	1	1	1	1	1	1	1
S	1	1	1	1	1	1	1	1	1	1	1
T	1	1	1	1	1	1	1	1	1	1	1
U	1	1	1	1	1	1	1	1	1	1	1

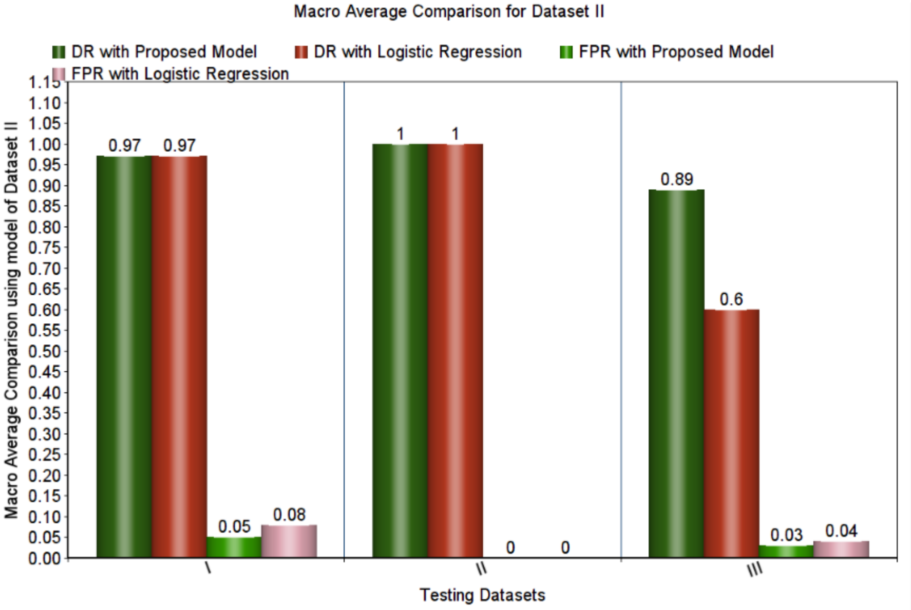


Fig. A.2. Macro Average Comparison for Dataset II with LR

Table A.6. Dataset II Cross-testing F1-Measure of SVM

Hosts	B	D	E	G	H	N	O	R	S	T	U
B	1	1	1	1	1	0.54	1	1	1	1	1
D	1	1	1	1	1	1	1	1	1	1	1
E	1	1	1	1	0.8	1	1	1	1	0.67	1
G	1	0	1	1	0.8	0.6	1	1	0	0.67	1
H	0.25	1	1	0.5	1	1	0.25	1	1	1	1
N	1	1	1	1	1	1	1	1	1	1	1
O	1	1	1	1	1	1	1	1	1	1	1
R	0.67	1	1	0.73	1	1	0.67	1	1	1	1
S	1	1	1	1	1	1	1	1	1	1	1
T	0.25	1	1	0.2	1	1	0.25	1	1	1	0.34
U	1	1	1	1	1	1	1	1	1	1	1

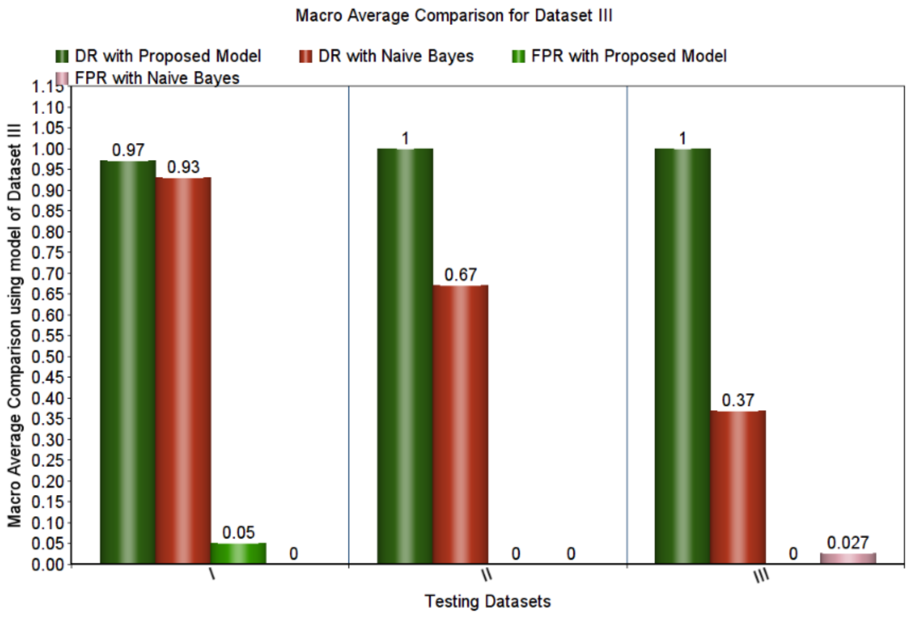


Fig. A.3. Macro Average Comparison for Dataset III with NB

Table A.7. Dataset III Cross-testing F1-Measure of Decision Trees

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
A	1	0.5	0.5	0.3	0.2	0.5	0.4	0.8	1	1	0.4	1	0.8	1	0.4	1	1	1
B	0.9	1	1	1	1	0.7	0.7	0.8	0.8	0.7	0.7	0.7	0.8	0.8	0.8	0.6	0.6	1
C	0.9	1	1	1	1	0.7	1	1	1	1	1	1	1	1	1	1	1	1
D	0.4	0.2	0.2	1	0.4	0.2	0.4	0.2	0.4	0.3	0.4	0.4	0.4	0.3	0.4	0.4	0.4	0.3
E	0.2	0.8	0.7	0.7	1	0.4	0.7	0.7	0.2	0.4	0.5	0.3	0.6	0.4	0.8	0.1	0	0.1
F	0.2	0.4	0.3	0.4	0.5	1	0.1	0	0	0	0	0	0	0	0.1	0	0	0.1
G	0.9	0.9	0.9	1	0.6	0.7	1	1	1	1	1	1	1	1	1	1	1	1
H	0.9	0.9	0.8	0.6	0.5	0.7	1	1	1	1	0.4	1	1	1	0.8	1	1	1
I	0.9	0.9	0.8	0.6	0.5	0.7	1	1	1	1	0.4	1	1	1	0.8	1	1	1
J	0.2	0.2	0.3	0.1	0	0.4	0.4	0.7	0.5	1	0.4	1	0.4	0.6	0.4	0.1	0	0.4
K	0.2	0.8	0.7	0	0.3	0.6	0.4	0.4	0.2	0.5	1	0.3	0.4	0.4	0.4	0	0	1
L	0.2	0.2	0.3	0.1	0	0.4	0.4	0.7	0.5	0.8	0.4	1	0.4	0.4	0.4	0.1	0	0.4
M	0.9	0.9	0.9	1	0.6	0.7	1	1	1	1	1	1	1	1	1	1	1	1
N	0.2	0.2	0.3	0.1	0	0.4	0.4	0.7	0.5	1	0.4	1	0.4	1	0.4	0.1	0	0.4
O	0.9	1	1	1	1	0.7	1	1	1	1	1	1	1	1	1	1	1	1
P	0.9	0.9	0.8	0.8	0.6	0.7	1	1	1	1	0.4	1	1	1	1	1	1	1
Q	0.9	0.9	0.8	0.6	0.5	0.7	1	1	1	1	0.4	1	1	1	0.8	1	1	1
R	0.2	0.7	0.7	0	0.2	0.4	0.3	0.5	0.1	0.3	0.3	0.1	0.5	0.3	0.2	0	0	1

Table A.8. Dataset III Cross-testing F1-Measure of KNN

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
A	1	0.6	0.5	0.7	0.3	0.5	0.4	1	1	1	0.4	1	0.8	1	0.4	1	1	1
B	0.9	1	0.9	1	0.4	0.7	0.7	0.9	0.8	0.7	1	0.8	0.7	0.8	0.8	0.6	0.6	1
C	0.9	0.8	1	0.3	0.4	0.7	1	1	1	1	1	1	1	1	1	1	1	1
D	0.9	0.6	0.6	1	0.5	0.6	1	0.3	0.6	0.5	0.4	0.5	0.7	0.4	0.7	1	0.8	0.7
E	0.9	0.8	0.9	1	1	0.7	1	0.3	0.5	0.4	0.6	0.5	0.6	0.3	0.7	0.6	0.6	0.6
F	0.9	0.8	1	0.3	0.3	1	1	0.9	0.5	1	1	1	0.8	0.5	1	1	0.5	1
G	0.9	0.8	0.9	0.3	1	0.7	1	1	1	1	1	1	1	1	1	1	1	1
H	0.9	0.8	0.8	0.3	0.6	0.7	1	1	1	1	0.4	1	1	1	0.8	1	1	1
I	0.9	0.8	0.8	0.1	0.5	0.7	1	1	1	1	0.4	1	1	1	0.8	1	1	1
J	0.9	0.8	0.8	0.1	0.2	0.7	1	0.9	1	1	0.4	1	1	1	0.8	1	1	1
K	0.9	0.8	1	0.3	0.3	0.7	1	1	1	1	1	1	1	1	1	1	1	1
L	0.9	0.8	0.8	0.1	0.4	0.7	1	1	1	1	0.4	1	1	1	0.8	1	1	1
M	0.9	0.8	0.9	0.1	0.4	0.7	1	1	1	1	1	1	1	1	1	1	1	1
N	0.9	0.7	0.8	0.1	0.4	0.7	1	1	1	1	0.4	1	1	1	0.8	1	1	1
O	0.9	0.8	1	0.3	1	0.7	1	1	1	1	1	1	1	1	1	1	1	1
P	0.9	0.8	0.8	0.3	0.6	0.7	1	1	1	1	0.4	1	1	1	1	1	1	1
Q	0.9	0.8	0.8	0.3	0.5	0.7	1	1	1	1	0.4	1	1	1	0.8	1	1	1
R	0.9	0.8	1	0.3	0.5	0.7	1	1	1	1	1	1	1	1	1	1	1	1

Table A.9. Dataset III Cross-testing F1-Measure of SVM

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
A	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
B	0.2	1	0.3	1	1	1	0.1	0	0	0	0	0	0	0	0.1	0	0	0.1
C	0.2	0.4	1	1	1	1	0.1	0	0	0	0	0	0	0	0.1	0	0	0.1
D	0.2	0.4	0.3	1	1	1	0.1	0	0	0	0	0	0	0	0.1	0	0	0.1
E	0.2	0.4	0.3	1	1	1	0.1	0	0	0	0	0	0	0	0.1	0	0	0.1
F	0.2	0.4	0.3	1	1	1	0.1	0	0	0	0	0	0	0	0.1	0	0	0.1
G	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
H	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
I	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
J	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
K	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
L	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
M	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
N	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
O	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
P	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
Q	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1
R	1	0.2	0.3	0.1	0	0.2	1	1	1	1	1	1	1	1	1	1	1	1

Table A.10. Random Forest Model of Dataset II Tested on Itself

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	Avg
DR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Table A.11. Random Forest Model of Dataset II Tested on Dataset I

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Avg
DR	1	0.69	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0.98**
FPR	0	0.9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.05**

Table A.12. Random Forest Model of Dataset II Tested on Dataset III

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	Avg
DR	1	0.91	0.73	0.6	0.85	1	1	1	1	1	0	1	1	1	0.5	1	1	1	**0.81**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Table A.13. Random Forest Model of Dataset III Tested on Itself

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	Avg
DR	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	**1**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Table A.14. Random Forest Model of Dataset III Tested on Dataset I

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Avg
DR	1	0.78	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0.98**
FPR	0	0.9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.05**

Table A.15. Random Forest Model of Dataset III Tested on Dataset II

Hosts	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	Avg
DR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**1**
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

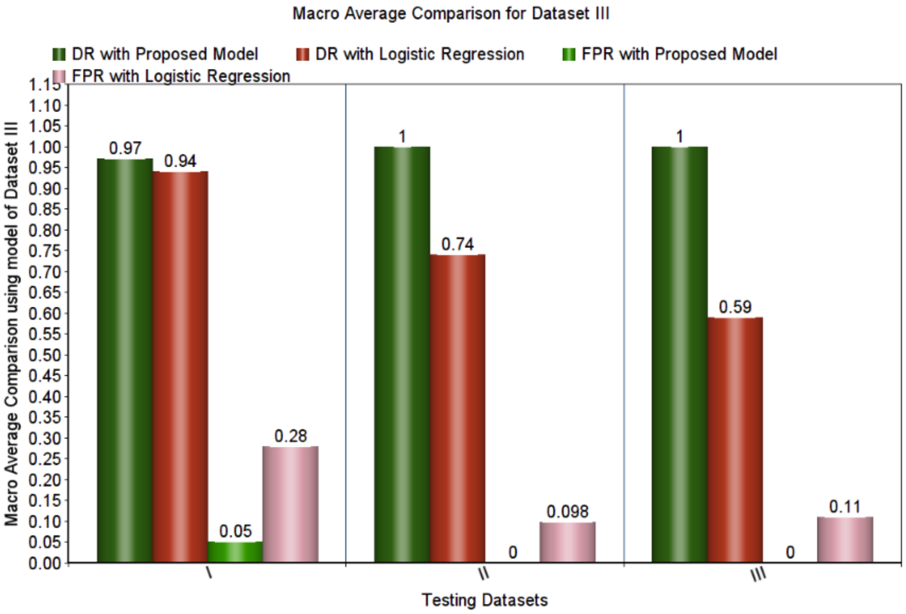


Fig. A.4. Macro Average Comparison for Dataset III with LR