# Benchmark Datasets for Network Intrusion Detection: A Review

**Article** *in* International Journal of Network Security · January 2018

DOI: 10.6633/IJNS.2018xx.20(x).xx)

**4 authors:**

Yasir Hamid
Abu Dhabi Polytechnic
**27** PUBLICATIONS **145** CITATIONS
SEE PROFILE

Balasaraswathi Ranganathan
Pondicherry Engineering College
**8** PUBLICATIONS **120** CITATIONS
SEE PROFILE

Ludovic Journaux
University of Burgundy
**64** PUBLICATIONS **323** CITATIONS
SEE PROFILE

Sugumaran Muthukumarasamy
Pondicherry Engineering College
**34** PUBLICATIONS **167** CITATIONS
SEE PROFILE

# Benchmark Datasets for Network Intrusion Detection: A Review

Yasir Hamid[1], V. R. Balasaraswathi[1], Ludovic Journaux [2] and M. Sugumaran[1]

*(Corresponding author: Yasir Hamid)*

Department of Computer Science and Engineering, Pondicherry Engineering College[1]

East Coast Road, Pillaichavadi, Puducherry 605014, India

Lab. LE2I UMR, CNRS 6306 AgroSup, F-21000, Dijon, France[2]

(Email: bhatyasirhamid@email.address)

*(Received Feb. 27, 2017; revised and accepted June 25, 2017)*

## Abstract

Network Intrusion Detection is the process of monitoring the events occurring in a computer system or the network and analyzing them for the signs of possible intrusions. An intrusion is a potentially harmful activity of malicious user, aimed at compromising the confidentiality, availability and integrity of the system. Over the decades intrusion detection (ID) problem has been visited by the researchers in various available environments like finite state automata, rule based systems, Markov probabilistic approach, statically sought solutions and most popular of all data mining and machine learning techniques. The prerequisite for data mining is that data should be present and there should be some hidden patterns in the data which need to be unearthed. In this work, we intend to provide a thorough review of the benchmark datasets available for Network Intrusion Detection (NID) which researchers in the field can use to train and test their models. In addition, this work as the first of its kind implements $k$-NN a simple most instance based type of classifier over all the datasets that doesn't require a well planed and monolithic training phase, across different neighborhood sizes. Results show that off all the datasets $k$-NN performs better on NSL-KDD dataset due to the fact that NSL-KDD doesn't have any redundant network connections and connections being fairly distributed across all the classes.

*Keywords: Benchmark Data-sets; Classification; IDS; k-NN; Machine Learning; Network Security*

## 1 Introduction

Due to the popularity of Internet in various important institutions of life like health, business, education and military it has been under the continuous threat from the people with dangerous mindset [24]. Hence, the need of securing the networks from such people is more than ever before. Over the years a boundless of methods, tools and devices have surfaced to secure the networks and computer systems ranging from anti viruses, firewalls, vulnerability testing and elimination of the programming errors that provide an back door entry point to the system [3, 20, 21].

All these devices for network security can be categorized as reactive or proactive [14], where proactive methods are based on the assumption of that security can be guaranteed and hence it is absolutely possible. Since Internet is distributed in nature and doesn't have any central security component. With the landscape of potentially harmful tools growing with each day and the fact that hackers are getting more smarter than ever before, prevention is no more effective. No matter how more securely a network is laid down, hackers will eventually find a way to break into the system and hence threaten the confidentiality, availability and integrity of the system.

This leads us to the reactive system that make use of network and system logs to unearth the foot prints of possibly disastrous network connections, one such devices that has got popularity and attracted tremendous research is Intrusion Detection System (IDS. Operating on a series of steps an IDS starts with collection of data from the networks, followed by data manipulation and finally a test for abnormality, that classifies a network connection as normal or abnormal. In the earlier phases, the research focal point lied with rule-based expert methods and statistical procedures. However these rule based systems tend to be less effective on the larger datasets and mostly end up performing their worst. Along these lines, a considerable measure of machine learning (ML) techniques have been acquainted to tackle the problem of ID. Both supervised and unsupervised ML techniques have been equally popular in IDS. IDS comes handy not only in successful detection of intrusions but is also effective in monitoring attempts to break security, which is indispensable for timely countermeasures against the ongoing intrusion activity [25].

An IDS can be categorized as Host based Intrusion Detection Systems (HIDS) or Network based Intrusion Detection Systems (NIDS) determined by its position of placement over the network system [23]. HIDS uses the data stored on single host and compromises of an agent on host that is entrusted with identifying the intrusions by examining the system calls, application logs, file modifications *etc.*

The advantage of HIDS is its simple nature and ease of installation, but the problem is that they are not immune against distributed and coordinated attacks. Contrary to this, NIDS collects and analyses the traffic communicated over the network. So it can carry out intrusion detection with security measures reflecting entire network information, hence preventing the attacks from reaching the hosts [1]. In view of the recognition technique at the most fundamental level an IDS can be classified as misuse based or anomaly based. Misuse based detection is trained on the labeled set compsinsing of instances labeled as as normal or an attack. The algorithm can detect the known attacks with high confidence but fail to detect novel attacks or for that case the simple variations of the known attacks [9]. Contrary to this anomaly based approaches are trained on the unlabelled data and in the training phase they build up the model for the normal connections. Over the span of operation the system captures the data and compares it with the model for the normal data. If it differs by more than some threshold then it is classified as an attack otherwise a legitimate connection. The power of anomaly based system comes with the ability of the detecting novel attacks, but this comes with the higher false alarm rates.

For the last few decades researchers have approached the ID problem in various models and most popular of the all is the application of data mining methods [13]. Data mining explores and analyzes gigantic datasets to unearth the useful and understandable patterns from the data. A prerequisite for the data mining is the availability of lots of data, presence of patterns in the data and presence of no simple model to understand the data. As for network intrusion detection is considered many datasets have surfaced over the years, with varied applicability, attack ranges and various capturing methods. In this paper we attempt to document most of the datasets available for intrusion detection and in addition to that we try to classify all the datasets using $k$-NN. The objective of application of $k$-NN across datasets over different neighborhoods is just to get an insight how well the data is classified.

The remainder of the paper is organized as follows. Section 2 provides a brief review of the literature for network intrusion detection. A brief discussion of machine learning techniques is given in Section 3, Section 4 provides a detailed discussion of various datasets available, the results of $k$-NN over different neighborhoods across all the datasets is given in Section 5, finally the paper concludes in Section 6.

## 2 Literature Review

Machine Learning (ML) techniques have been well sought and highly popular for Network Intrusion Detection. A group of varied and widespread ML techniques spanning both supervised as well as unsupervised ML groups have been applied for network intrusion detection. As for supervised ML techniques are considered almost all the techniques have been equally popular, like Decision Trees [12] Artificial Neural Networks (ANN) [16], Support Vector Machines [10] and for unsupervised machine learning techniques $k$-NN [15], have been pretty popular. Moreover, there have a lot been of efforts in designing both hybrid [22] and ensemble [19] models for the network intrusion detection. Review of the literature enlightens about the fact that KDD99 has been well accepted and thoroughly used dataset among the community. Lately, Bio-inspired algorithms also have been applied in network intrusion detection mostly in pre-processing to select an optimal subset of features possibly non-redundant and highly correlated with the class and least correlation with other features Ant Colony Optimization [11], Cuttle Fish Algorithm [8], Particle Swarm Optimization [6]. Due to excessive computational requirements and inherent drawbacks of the dataset the full dataset has been used very seldom. So, mostly the researchers have relied on selecting a portion of the dataset and trained and tested the models on the extracted subset of the data.

## 3 A Review of Machine Learning Techniques

Machine Learning when considered in generic means to form meaningful predictive models from the tsunami of data. Its various prospects extend to anomaly detection, prevention of fraud, intrusion in networks, lifetime support such as DNA analysis, tumor detection, sentiment analysis in social networks, detection of objects in satellite imageries and making appropriate decisions in the automated vehicles. Machine learning actually boots in the training data, then learns from the samples and builds up a congruous model, next tests itself and then predicts the data or engenders decision from the live environment without being precisely programmed for each action [4].

Machine learning is segregated into three broad groups *i.e*, supervised learning, unsupervised learning and reinforcement learning.

### 3.1 Supervised Learning

Supervised learning confides over the training samples to build a prognosticative model. Supervised learning model employs over pair which encompasses the input object and desired output value [2]. There are many supervised learning techniques embracing Decision Tree learning, Support Vector Machine, K-nearest Neighbor, Naive Bayes and Random Forest. The decision tree learning

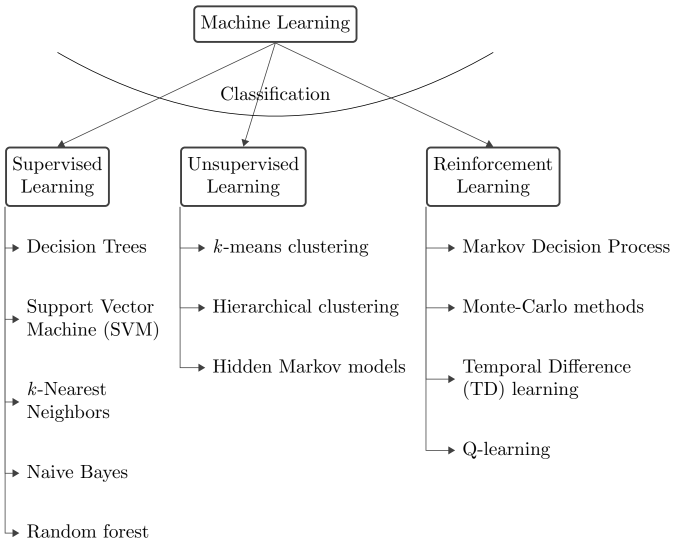

Figure 1: Machine learning classification

sets upon to create a prototype which will forecast the value of an objective variable by manipulating over a set of decision variables down the tree. Scrutinizing diverse decision variable along the path by traversing from the root node to leaf node which contains the value of the objective node. Support vector machine [5] endeavors to find out the perfect separating hyperplane that can perfectly classify the objects. Its ambition is to maximize the distance between the data points and minimize the distance between hyperplane and data points. The $k$-nearest neighbor is lazy non-parametric learning technique. The $k$-NN in contrast to SVM takes the entire training data for consideration where SVM disposes of the non-support vectors. Naive Bayes is a probabilistic classifier, where it weighs the probability of each event based on certain conditions associated with it. Random forest is aggregate of decision trees, with each tree of the forest promoting a vote over the decision made and finally aftermath will be label having a maximum number of votes.

**Algorithm 1** Supervised Learning Technique

```
1: procedure SUPERVISED LEARNING
2:     Fetch in training data set $X = \{\alpha^n, \beta^n\}$ $\alpha$ is data point; $\beta$ is labelling
3:     Build Predictive model based on the training set
4:     Fetch Objective variable $new\alpha$ which needs to be labelled based on $new\alpha = \begin{cases} \epsilon^1, & \text{Class A} \\ \epsilon^2, & \text{Class B} \\ \ldots. & \\ \epsilon^n & \text{Class N} \end{cases}$ Each $\epsilon$ is of different threshold values
5: end procedure
```

## 3.2 Unsupervised Learning

Unsupervised learning is enforced over data samples where there doesn't exist prior labeling of data samples. But Unsupervised Learning is still a conundrum, clear understanding will revolutionize this field. It produces a function to classify the data by hunting out the concealed structure within the data. It includes K-means clustering, Hierarchical Clustering and Hidden Markov model. K-means clustering works over the principle of feature congruence. Hierarchical clustering fashions up to form clusters building up either in top-down or bottom-up trend namely divisive and agglomerative. Hidden Markov model plays hardball over data with time series. Its principle is over probability based Bayesian network.

**Algorithm 2** Unsupervised Learning Technique

```
1: procedure UNSUPERVISED LEARNING
2:     Fetch data set $X = \{\alpha^1, \alpha^2, \alpha^3.....\alpha^n\}$ where $\alpha$ represents a data point
3:     Build Clustering model to form clusters
4: $\begin{cases} \text{if } \epsilon^1, & \text{then Group A} \\ \text{if } \epsilon^2, & \text{then Group B} \\ & \text{Using some distance measure} \\ \ldots. & \\ \text{if } \epsilon^n, & \text{then Group N} \end{cases}$
   form $\epsilon^n$ threshold values, thus forming N Clusters
5:     Fetch Objective variable $new\alpha$ which needs to be classified $new\alpha = \begin{cases} \epsilon^1, & \text{Group A} \\ \epsilon^2, & \text{Group B} \\ \ldots. & \\ \epsilon^n & \text{Group N} \end{cases}$
6: end procedure
```

## 3.3 Reinforcement Learning

Reinforcement Learning is primarily used by software agents which need to make a decision in an environment. This learning doesn't bother about immediate rewards, rather its ultimate motive is a win over the environment with high reward tardily [17]. Markov decision process, Monte Carlo methods, Temporal Difference learning and Q-learning are some of the reinforcement learning. The Markov decision process is used in footing where the fallouts are only partially in control. Monte Carlo methods equate over averaging the returns of the sample. Temporal Difference Learning is a perfect blending of Monte Carlo and Dynamic programming. Q-Learning works by recommending best selection policy for each action by analyzing the play by play over the past period of time.

## 3.4 *k*-NN

$k$-NN is an intelligible method which performs preposterously well. $k$-NN is dexterous in nature and used across various domains. It is actually a lazy learning algorithm.

**Algorithm 3** Reinforcement Learning Technique

```
1: procedure Reinforcement Learning
2:     Fetch states $\gamma$ in the Environment $\xi$
3:     for all $\gamma_i$ in $\gamma$ do:
4:         Find Best $\delta_i$ from $\delta$ assign reward value $\omega_i$ to it
5:     end for
6: where $\delta$ is the action and $\omega$ is the reward
7:     Given $new\gamma$ find action $\delta$
8:     if $\delta_i$ have greater $\omega$ then:
9:         $new\gamma$ is assigned $\delta_i$
10:    end if
11:    Ultimate goal to Win the $\xi$ with best $\omega$
12: end procedure
```

The $k$-factor gives the value of how many closest data points to be considered to conclude the classification of the observatory data point. In the point of fact, the $k$-factor is the deciding parameter of how well the algorithm can perform efficiently. Once the $k$-factor's value is decided, the voting among the $k$ closest points are taken into consideration and label which has the maximum votes wins the classification. If the $k$-factor is determined is one, initially, it will have zero error in classification but the error curve will drop down to zero and then maneuver an upward movement indicating the increase in error rate, when $k$-factor is taken large, then happens under-fitting dispute. So it is indispensable to decide the impeccable value of the $k$-factor.

**Algorithm 4** $k$-NN

```
1: procedure k-NN Algorithm
2:     Fetch in training data set $X = \{\alpha^n, \beta^n\}$ $\alpha$ is data point; $\beta$ is labelling
3:     Determine the Value of K-factor
4:     Fetch Objective variable $new\alpha$ which needs to be labelled
5:     Based on K-Factor $\delta^n$ fetch the nearby data-points of the $new\alpha$ based on Distance measure $\omega$
6:     for all $i$ in $Countof\delta$ do:
7:         if $\omega$ have lesser $\xi$ then:
8:             $\delta[] = \alpha_i$
9:         end if
10:    end for
11: Each $\xi$ is of threshold distance measure
12:    Find the votes $\zeta_i$ $\delta_i = \begin{cases} \epsilon^1, & \text{Class A} \\ \epsilon^2, & \text{Class B} \\ .... & \\ \epsilon^n & \text{Class N} \end{cases}$
13:    Find $\beta$ label which has maximum no of votes in $\zeta_i$ and this is the Final labelling to the $new\alpha$
14: end procedure
```

## 4 Data Sets

In the next few subsections, we provide an overview of various datasets used for network intrusions over the years. Although this list is by no ways a complete documentations of all the datasets, as users over the course of time have used randomly drawn subsets of the full dataset. We provide a review of the stable and very well known datasets, that are pretty popular in the ML fraternity.

### 4.1 DARPA98

DARPA98 was the first standard corpora for the assessment of network intrusion detection, collected and distributed by MIT Lincoln under the joint sponsorship of DARPA and ARFL. Over the decades the dataset has been popularly used for training and testing models of IDS. These assessments contributed fundamentally to the intrusion detection by giving guidance for research endeavors and a target adjustment of the specialized cutting edge. The dataset was formed by recording whole network traffic including the payload in Tcpdump format for each connection. Both real and simulated machines were used for the data collection and in general, the data collected was in the form of sniffed network traffic, Solaris BSM audit data, Windows NT audit data (in the case of DARPA 1999), file system snapshots aimed at identifying the intrusions that were being carried out against a test network while the data was being collected. The dataset is comprised of 7 weeks data for the training and 2 weeks for the testing purposes. More than three hundred instances of 38 attacks broadly categorized in 4 different groups like DOS, U2R, R2L and Probe are present in the dataset.

### 4.2 KDD99

The Third International Knowledge Discovery and Data Mining Tools Competition was held in conjunction with KDD99, The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks and "good" normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment and the dataset that was used was processed DARPA98 data and became popular as a KDD99 dataset. The KDD99 dataset is the most popular dataset among machine learning practitioners for network intrusion detection. Each connection record consists of 41 data fields and there is 42nd attribute designating the class to which the connection belongs. In total there are 23 attack groups in the data and there is also a class for normal connections. The 23 attack groups are broadly categorized into four attack families i.e, Dos, Probe, R2L and U2R. Each connection record is mixed in nature, with some attributes

Table 1: KDD attributes and their types

| BASIC | | | CONTENT | | | TRAFFIC | | | HOST | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **SNO** | **FEATURE** | **TYPE** | **SNO** | **FEATURE** | **TYPE** | **SNO** | **FEATURE** | **TYPE** | **SNO** | **FEATURE** | **TYPE** |
| 1 | duration | C | 10 | hot | C | 23 | count | C | 32 | dst_host_count | C |
| 2 | protocol_type | N | 11 | num_failed_logins | C | 24 | serror_rate | C | 33 | dst_host_srv_count | C |
| 3 | service | N | 12 | logged_in | N | 25 | rerror_rate | C | 34 | dst_host_same_srv_rate | C |
| 4 | src_bytes | C | 13 | num_compromised | C | 26 | same_srv_rate | C | 35 | dst_host_diff_srv_rate | C |
| 5 | dst_bytes | C | 14 | root_shell | N | 27 | diff_srv_rate | C | 36 | dst_host_same_src_port_rate | C |
| 6 | flag | N | 15 | su_attempted | N | 28 | srv_count | C | 37 | dst_host_srv_diff_host_rate | C |
| 7 | land | N | 16 | num_root | C | 29 | srv_serror_rate | C | 38 | dst_host_serror_rate | C |
| 8 | wrong_fragment | C | 17 | num_file_creations | C | 30 | srv_rerror_rate | C | 39 | dst_host_srv_serror_rate | C |
| 9 | urgent | C | 18 | num_shells | C | 31 | srv_diff_host_rate | C | 40 | dst_host_rerror_rate | C |
| | | | 19 | num_access_files | C | | | | 41 | dst_host_srv_rerror_rate | C |
| | | | 20 | num_outbound_cmds | C | | | | 42 | class | |
| | | | 21 | is_hot_login | N | | | | | | |
| | | | 22 | is_guest_login | N | | | | | | |

being nominal and others being numeric. The distribution of connections across different groups is uneven and there are more no of connections pertaining to two dominating attack groups i.e, Probe and R2L and also the normal connections. There are only a few instances of the R2L and U2R attacks. A large number of variations of KDD99 datasets some being the corrections of the base dataset while others being the carefully drawn subsets have surfaced up over the years. These 41 attributes are categorized into four broad groups as

- Basic features: attributes representing individual connections.
- Content features: The features within a connection as suggested by domain knowledge.
- Traffic features: features comprised of 2 second time window.
- Host features: attributes for assessing attacks lasting for more than 2 seconds.

Some of the attributes are nominal and most are continuous. To provide a better picture of the dataset, Table 1 records and groups the attributes into different categories, moreover for each attribute, we mention its nature, whether it is nominal or numeric.

In the following few subsections, we will provide a brief discussion of variations of KDD99 datasets.

#### 4.2.1 Full KDD

Full KDD in total consists of 48, 98, 430 instances, the file is of 750mb. All the connections of the dataset fall under four groups. The frequency of the attacks is not normal, where there are few dominating classes like Neptune and smurf with 10, 72, 017 and 28, 07, 886 instances respectively. But there are also less frequent attack groups like Spy, Perl, PHP. This uneven distribution of connections across different attack groups hampers the detection rate of any classification system and results in a biased classifier.

#### 4.2.2 Corrected KDD

The problem with the full KDD99 dataset is the massive repetitions of few instances. This hampers the effective classification as the results of most of the classifier are biased towards the dominant classes. Just to provide the researchers a potentially fair dataset to train and test model a variation of the full dataset called corrected KDD99 dataset was formed. A total of 3, 11, 029 network connections are present in corrected KDD dataset. Care was taken to maintain the representation of all the attacks and at the same time reduces the redundancy and repetitions.

#### 4.2.3 10%KDD

Due to the enormous size of KDD dataset, it is practically not possible to use it fully for the training set and hence usually researchers select a subset of the dataset from the full set to train and test their models. One such carefully drawn a subset of the total subset has been pretty popular in the machine learning fraternity and is known as 10%KDD. In total, this fraction of the dataset has 4, 89, 843 instances. This compact size of 10%KDD makes it suitable for training and testing of IDS models. In total 4, 89, 843 connections are present in the dataset.

#### 4.2.4 NSL KDD

NSL-KDD data set is a refined version of its predecessor KDD99 dataset formulated with the aim of minimizing the bias of the classifier. The advantage of NSL KDD dataset over full KDD are unbiased classification and reliable results as there are no redundant records in the dataset. Better detection rate offered by eliminating the duplicate results. A sort of balance and representation across the groups of attacks by selecting the number of instances from each group inversely proportional to original KDD dataset.

Table 2 given below provides a thorough analysis of all the variations of the KDD dataset present in literature. For each dataset, we present the number of instances of different attack classes. Of all the variations of the dataset corrected KDD99 data has highest a number of attacks, There are in total 38 attacks in the corrected KDD99 dataset, 23 of them are same as that of the other variations with 15 new attacks being incorporated.

Table 2: Analysis of variations of dataset

| Attack | Total Instances | | | | | | Attack Group |
|---|---|---|---|---|---|---|---|
| | Full KDD | Corrected | 10%KDD | NSLKDD | NEW_ATTACKS | #Count | |
| back | 2203 | 1098 | 2203 | 956 | apache2 | 794 | |
| land | 21 | 9 | 21 | 18 | processtable | 759 | |
| neptune | 1072017 | 58001 | 107201 | 41214 | | | DOS |
| pod | 264 | 87 | 264 | 201 | | | |
| smurf | 2807886 | 164901 | 280790 | 2646 | | | |
| teardrop | 979 | 12 | 979 | 892 | | | |
| satan | 15892 | 1633 | 1589 | 3633 | saint | 736 | |
| ipsweep | 12481 | 306 | 1247 | 3599 | snmpguess | 2406 | PROBE |
| nmap | 2316 | 84 | 231 | 1493 | mscan | 1053 | |
| portsweep | 10413 | 354 | 1040 | 2931 | | | |
| normal | 972781 | 60593 | 97277 | 67343 | | | NORMAL |
| guess_passwd | 53 | 4367 | 53 | 53 | worm | 2 | |
| ftp_write | 8 | 3 | 8 | 8 | xlock | 9 | |
| imap | 12 | 1 | 12 | 11 | xsnoop | 4 | |
| phf | 4 | 2 | 4 | 4 | xterm | 13 | R2L |
| multihop | 7 | 18 | 7 | 7 | httptunnel | 158 | |
| warezmaster | 20 | 1602 | 20 | 20 | named | 17 | |
| warezclient | 1020 | 0 | 1020 | 1020 | sendmail | 17 | |
| spy | 2 | 0 | 2 | 2 | snmpgetattack | 7741 | |
| buffer_overflow | 30 | 30 | 30 | 30 | ps | 16 | |
| loadmodule | 9 | 9 | 9 | 9 | sqlattack | 2 | U2R |
| perl | 3 | 3 | 3 | 3 | | | |
| rootkit | 10 | 10 | 10 | 2931 | | | |

### 4.3 Caida DDoS Dataset

CAIDA aims at collecting and sharing of the data for research analysis such as security-related purpose, internet traffic analysis, performance, routing *etc.* It collects data at topologically and geographically different locations and makes the data accessible to the research community. Many categories of datasets are available for various scientific analysis. Internet traces datasets from 2008 to 2016 are available. A most important threat in internet service is Distributed Denial of Service (DDoS) attack [3]. Denial of service attacks try to compromise the availability of the system by keeping it too busy to respond to the service requests of the legitimate users. This type of attack attempts to block access to the targeted server by consuming computing resources on the server and by consuming all of the bandwidth of the network connecting the server to the Internet. This dataset contains approximately one hour of anonymized traffic traces from a DDoS attack on August 4, 2007 (20:50:08 UTC to 21:56:16 UTC). The one-hour trace is split up in 5-minute pcap files. The total size of the dataset is 5.3 GB (compressed; 21 GB uncompressed). Only attack traffic to the victim and responses to the attack from the victim are included in the traces. Non-attack traffic has as much as possible been removed. Traces in this dataset are anonymized using CryptoPAn prefix-preserving anonymization using a single key. The payload has been removed from all packets.These traces can be read with any software that reads the pcap (tcpdump) format, including the CoralReef Software Suite, tcpdump, Wireshark and many others.

### 4.4 ADFA Linux

Creech and Lu proposed an ADFA Linux (ADFA-LD) cyber security benchmark dataset for evaluation of IDS in the year of 2013 [7]. Ubuntu Linux version 11.04 was used as host Operating System for generating ADFA-LD. It has a higher similarity between normal and attack dataset and contains updated and modern attacks. ADFA-LD is being used by soft computing, cyber security, data mining, machine learning research communities to evaluate the performance of IDS. This dataset provides a contemporary Linux and Windows dataset for evaluation of for host-based intrusion detection system (HIDS).

### 4.5 UNM Dataset

UNM benchmark dataset was proposed in the year of 2004. System calls data was captured to form this dataset. The dataset contains a very rich set of intrusions such as buffer overflows, symbolic link attacks and trojan programs. Since its scope was very much limited, it cannot replace KDD dataset.

### 4.6 UNSW-NB15 Dataset

UNSW-NB 15 data set was created in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) by the IXIA Perfect Storm tool. This dataset contains a hybrid of normal activities and attack behaviors. Tcpdump tool was used to capture 100 GB of the raw traffic. Twelve algorithms and tools such as Argus, Bro-IDS were used to generate UNSW-NB15. It contains 49 features including a class label [18]. The features, their type and a detailed description is given in Table 3.

A total of 49 attributes determining the features of connections are present for each data instance. The attributes are mixed in nature with some being nominal, some being numeric and some taking on time-stamp values as given in Table 4.

The attributes of the dataset are categorized into 6 broad groups, the details of which are given in Table 5.

This dataset contains a total of 25,40,044 labeled instances, each being labeled either normal or attack. The distribution of connections across the two groups is presented in Table 6.

In total there are nine types of attacks in the dataset in addition to one group representing the normal data. The attacks are categorized as Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The details of attacks, subcategory of attacks and the protocols they use are given in Table 7.

## 5 A Comparison of Data Sets

$k$-NN classifier with various neighborhood sizes ranging from five to ten was used as a classification model across all the datasets. From the variations of KDD we selected 10 % KDD, Corrected and NSL KDD only. For other datasets since they are having some nominal features like payload, an arbitrary coding was done to convert the nominal values to numeric values. Other datasets having separate files for different attacks and transactions, we merged them into single file to represent each different dataset. Euclidean distance was used to measure the similarity between the instances so as to select the neighbors of a data point. The reasons for using this distance measure is its simplicity and sound mathematical background. As for performance metrics are considered this

Table 3: Features of UNSW-NB15 dataset

| SNo. | Name | Type | Description |
|---|---|---|---|
| 36 | is_sm_ips_ports | Binary | "If source (1) and destination (3)IP addresses equal and port numbers (2)(4) equal thenthis variable takes value 1 else 0" |
| 39 | is_ftp_login | | If the ftp session is accessed by user and password then 1 else 0. |
| 49 | Label | | 0 for normal and 1 for attack records |
| 7 | dur | Float | Record total duration |
| 15 | Sload | | Source bits per second |
| 16 | Dload | | Destination bits per second |
| 27 | Sjit | | Source jitter (mSec) |
| 28 | Djit | | Destination jitter (mSec) |
| 31 | Sintpkt | | Source interpacket arrival time (mSec) |
| 32 | Dintpkt | | Destination interpacket arrival time (mSec) |
| 33 | tcprtt | | "TCP connection setup round-trip time, the sum of synack and ackdat." |
| 34 | synack | | "TCP connection setup time the time between the SYN and the SYN_ACK packets." |
| 35 | ackdat | | "TCP connection setup time the time between the SYN_ACK and the ACK packets." |
| 2 | sport | Integer | Source port number |
| 4 | dsport | | Destination port number |
| 8 | sbytes | | Source to destination transaction bytes |
| 9 | dbytes | | Destination to source transaction bytes |
| 10 | sttl | | Source to destination time to live value |
| 11 | dttl | | Destination to source time to live value |
| 12 | sloss | | Source packets retransmitted or dropped |
| 13 | dloss | | Destination packets retransmitted or dropped |
| 17 | Spkts | | Source to destination packet count |
| 18 | Dpkts | | Destination to source packet count |
| 19 | swin | | Source TCP window advertisement value |
| 20 | dwin | | Destination TCP window advertisement value |
| 21 | stcpb | | Source TCP base sequence number |
| 22 | dtcpb | | Destination TCP base sequence number |
| 23 | smeansz | | Mean of the ?ow packet size transmitted by the src |
| 24 | dmeansz | | Mean of the ?ow packet size transmitted by the dst |
| 25 | trans_depth | | Represents the pipelined depth into the connection of http request/response transaction |
| 26 | res_bdy_len | | Actual uncompressed content size of the data transferred from the servers http service. |
| 37 | ct_state_ttl | | No. for each state (6) according to specific range of values for source/destination time to live (10) (11). |
| 38 | ct_flw_http_mthd | | No. of flows that has methods such as Get and Post in http service. |
| 40 | ct_ftp_cmd | | No of flows that has a command in ftp session. |
| 41 | ct_srv_src | | No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26). |
| 42 | ct_srv_dst | | No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26). |
| 43 | ct_dst_ltm | | No. of connections of the same destination address (3) in 100 connections according to the last time (26). |
| 44 | ct_src_ ltm | | No. of connections of the same source address (1) in 100 connections according to the last time (26). |
| 45 | ct_src_dport_ltm | | No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26). |
| 46 | ct_dst_sport_ltm | | No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26). |
| 47 | ct_dst_src_ltm | | No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26). |
| 1 | srcip | Nominal | Source IP address |
| 3 | dstip | | Destination IP address |
| 5 | proto | | Transaction protocol |
| 6 | state | | Indicates to the state and its dependent protocol |
| 14 | service | | "http ftp smtp ssh dns ftp-data |
| 48 | attack_cat | | "The name of each attack category. In this data set nine categories *eg* Fuzzers Analysis Backdoors DOS exploits Generic Reconnaissance Shellcode and Worms" |
| 29 | Stime | Timestamp | record start time |
| 30 | Ltime | | record last time |

work takes into consideration various measures like Accuracy, Precision and Recall for $k$-NN for all the data sets.

Table 4: Features type of UNSW-NB15 dataset

| SNo. | Feature Type | Count |
|---|---|---|
| 1 | Nominal | 6 |
| 2 | Integer | 28 |
| 3 | Binary | 3 |
| 4 | Float | 10 |
| 5 | Timestamp | 2 |

Table 5: UNSW-NB15 dataset feature categorization

| SNo. | Name of the category | Description |
|---|---|---|
| 1 | Flow features | It contains the identifier attributes between hosts such as client-to-serve or server to-client |
| 2 | Basic features | It includes the attributes that characterize the connections of protocols. |
| 3 | Content features | It contains the attributes of TCP/IP and also contain some attributes of http services |
| 4 | Time features | It includes the attributes of time such as round trip time of TCP protocol start/end packet time arrival time between packets *etc.* |
| 5 | Additional generated features<br>General purpose features(from number 36 - 40) | Own purpose features which to care for the protocols service. |
|  | Connection features (from number 41- 47) | Built based on the chronological order of the last time feature |
| 6 | Labelled Features | It represents the label of the record. |

Table 6: Details of events in UNSW-NB15 dataset

| Name | Count |
|---|---|
| Total Number of events | 2540044 |
| Normal | 2218761 |
| Attacks | 321283 |

Table 7: Categorizations of attacks in UNSW-NB15 dataset

| Attack Category | Attack Subcategory | Number of Events |
|---|---|---|
| Fuzzers | FTP,HTTP,RIP,SMB,Syslog,PPTP,FTP,DCERPC, OSPF,TFTP,DCERPC,OSPF,BGP | 24246 |
| Reconnaissance | Telnet, SNMP, SunRPC Portmapper (TCP) UDP Service , SunRPC Portmapper (TCP) UDP Service, SunRPC Portmapper (TCP) TCP Service, SunRPC Portmapper (UDP) UDP Service, NetBIOS, DNS, HTTP, SunRPC Portmapper (UDP), ICMP, SCTP, MSSQL, SMTP,NETBIOS,DNS | 13987 |
| Shellcode | FreeBSD, HP-UX, NetBSD, AIX, SCO Unix, Linux, Decoders, IRIX, OpenBSD, Mac OS X, BSD, Windows, BSDi, Multiple OS, Solaris | 1511 |
| Analysis | HTML,Portscanner,Spam | 2677 |
| Backdoors | - | 2329 |
| DoS | Ethernet, Microsoft Office, VNC, IRC, RDP, TCP, VNC, FTP, LDAP, Oracle, TCP, TFTP, DCERPC, XINETD, IRC, SNMP, ISAKMP, NTP, Telnet, CUPS, Hypervisor, ICMP, SunRPC, IMAP, Asterisk, Browser | 16353 |
| Exploits | Evasions, SCCP, SSL, VNC, Backup Appliance, Browser, Clientside Microsoft Office, Interbase, Miscellaneous Batch, SOCKS, TCP, Apache,IMAP, Microsoft IIS, SOCKS, Clientside, Clientside Microsoft Paint, IDS, SSH, ICMP, IDS, DCERPC, FTP, RADIUS, SSL, WINS, Clientside, Clientside Microsoft, POP3, Unix r Service, Cisco IOS, Clientside Microsoft Media Player, Dameware,LPD,MSSQL ,Office Document, RTSP,SCADA,VNC, Webserver,All,LDAP,NNTP,IGMP, Oracle,RDesktop,Telnet,Apache,PHP,SMB,SunRPC,Web Application,DNS,Evasions, RADIUS,BrowserFTP,PPTP,SCCP,SIP,TFTP | 44525 |
| Generic | All,SIP,HTTP,SMTP,IXIA,TFTP,SuperFlow,HTTP,TFTP | 215481 |
| Worms | - | 174 |

Table 8 given below presents the results of $k$-NN on all datasets across varying size of neighborhoods. For each run of $k$-NN, we check the Accuracy, Precision and Recall on all the datasets. As can be seen from the table $k$-NN has better detection rate and for NSL-KDD data followed by Caida and UNSW-NB respectively in terms of accuracy, precision and recall. The reason for better results on NSL-KDD as compared to other is that this dataset doesn't contain any redundant connections and more over the data is distributed evenly across different classes of attacks and also for normal connections.

In Figure 2 given below a plot of accuracy of $k$-NN on all the datasets over five different neighborhood sizes is presented. The accuracy is calculated as $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$. As can be seen from the figure on the average $k$-NN has better accuracy on NSL-KDD dataset.

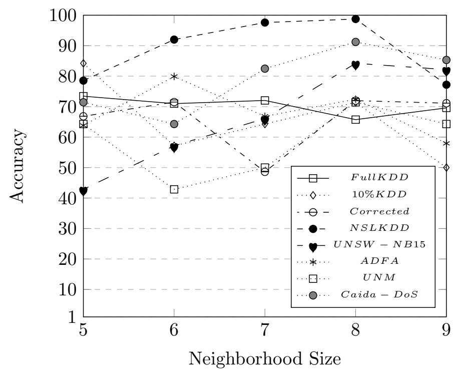

Figure 2: Accuracy

In Figure 2 given below a plot of precision of $k$-NN on all the datasets over five different neighborhood sizes is presented. The accuracy is calculated as $Precision = \frac{TP}{TP+FP}$. As can be seen from the figure on the average $k$-NN has better precision on NSL-KDD dataset.

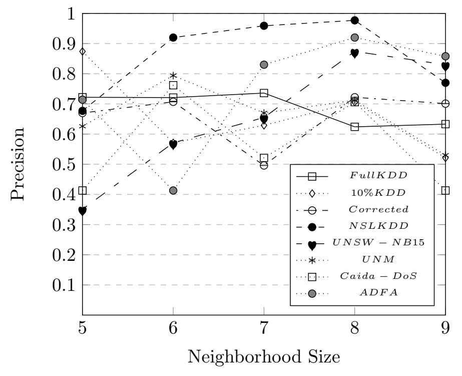

Figure 3: Precision

In Figure 2 given below a plot of Recall of $k$-NN on all the datasets over five different neighborhood sizes is presented. The accuracy is calculated as $Recall = \frac{TP}{TP+TN}$. As can be seen from the figure like accuracy and precision $k$-NN has better recall as well on NSL-KDD dataset.

Table 8: $k$-NN results on datasets

| Dataset | Neighborhood | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 5 | | | 6 | | | 7 | | | 8 | | | 9 | | |
| | Accuracy | Precision | Recall | Accuracy | Precision | Recall | Accuracy | Precision | Recall | Accuracy | Precision | Recall | Accuracy | Precision | Recall |
| Full KDD99 | 73.426 | 0.722 | 0.734 | 70.979 | 0.721 | 0.710 | 72.028 | 0.736 | 0.720 | 65.734 | 0.624 | 0.657 | 69.5804 | 0.633 | 0.696 |
| Corrected KDD | 66.822 | 0.670 | 0.668 | 71.4953 | 0.707 | 0.715 | 48.598 | 0.496 | 0.486 | 71.962 | 0.722 | 0.720 | 71.028 | 0.701 | 0.710 |
| NSLKDD | 78.537 | 0.677 | 0.785 | 92.000 | 0.920 | 0.920 | 97.592 | 0.959 | 0.976 | 98.750 | 0.977 | 0.988 | 77.193 | 0.770 | 0.772 |
| 10% KDD | 84.210 | 0.874 | 0.842 | 57.142 | 0.571 | 0.571 | 64.285 | 0.629 | 0.643 | 71.428 | 0.706 | 0.714 | 50.000 | 0.521 | 0.500 |
| UNSW | 42.857 | 0.351 | 0.429 | 57.142 | 0.571 | 0.571 | 66.083 | 0.655 | 0.661 | 84.210 | 0.874 | 0.842 | 82.4561 | 0.830 | 0.825 |
| Caida | 64.285 | 0.413 | 0.643 | 42.857 | 0.762 | 0.351 | 50 | 0.521 | 0.500 | 71.428 | 0.706 | 0.714 | 64.285 | 0.413 | 0.643 |
| ADFA Windows | 71.428 | 0.714 | 0.714 | 64.285 | 0.413 | 0.643 | 82.456 | 0.830 | 0.825 | 91.228 | 0.920 | 0.912 | 85.308 | 0.858 | 0.853 |
| UNM Dataset | 63.827 | 0.626 | 0.638 | 79.906 | 0.794 | 0.799 | 66.822 | 0.670 | 0.668 | 72.429 | 0.712 | 0.724 | 57.943 | 0.530 | 0.579 |

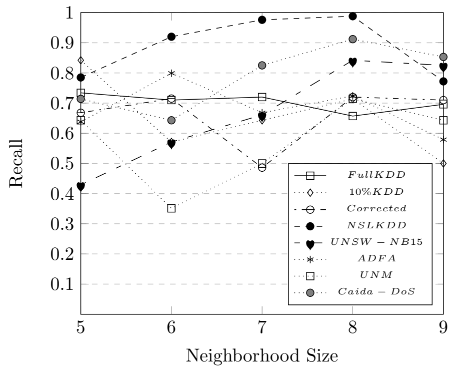

Figure 4: Recall

## 6 Conclusion

In this work, a thorough review of various benchmark datasets for network intrusion detection is given. The main objective of this work is to provide researchers an idea about what all the datasets are available for the network intrusion detection and what each dataset is comprised of of in terms of features, attacks *etc.* For each dataset, we provided a detailed discussion of its instances, attributes, classes and also the nature of attributes. In addition to that just to get a feel of classification we implemented $k$-NN classifier employing Euclidean distances over different neighborhoods across all the datasets. The results showed that k-NN performs better and has better Accuracy, Precision and Recall on NSL-KDD, because of the even distribution of instances across various classes and minimal redundancy among the records.

## References

[1] R. A. R. Ashfaq, X. Z. Wang, J. Z. Huang, H. Abbas and Y. L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484–497, 2017.

[2] M. C. Belavagi and B. Muniyal, "Performance evaluation of supervised machine learning algorithms for intrusion detection," *Procedia Computer Science*, vol. 89, pp. 117–123, 2016.

[3] M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita, "An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection," *Pattern Recognition Letters*, vol. 51, pp. 1–7, 2015.

[4] K. J. Chabathula, C. D. Jaidhar and M. A. A. Kumara, "Comparative study of principal component analysis based intrusion detection approach using machine learning algorithms," in *3rd International Conference on Signal Processing, Communication and Networking (ICSCN'15)*, pp. 1–6, Mar. 2015.

[5] N. Chand, P. Mishra, C. R. Krishna, E. S. Pilli and M. C. Govil, "A comparative analysis of svm and its stacking with other classification algorithm for intrusion detection," in *2016 International Conference on Advances in Computing, Communication, Automation (ICACCA'16)*, pp. 1–6, Apr. 2016.

[6] M. H. Chen, P. C. Chang and J. L. Wu, "A population-based incremental learning approach with artificial immune system for network intrusion detection," *Engineering Applications of Artificial Intelligence*, vol. 51, pp. 171–181, 2016.

[7] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," in *IEEE Wireless Communications and Networking Conference (WCNC'13)*, pp. 4487–4492, 2013.

[8] A. S. Eesa, Z. Orman and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Systems with Applications*, vol. 42, no. 5, pp. 2670–2679, 2015.

[9] K. Gai, M. Qiu, L. Tao and Y. Zhu, "Intrusion detection techniques for mobile cloud computing in heterogeneous 5G," *Security and Communication Networks*, vol. 9, no. 16, pp. 3049–3058, 2016.

[10] Y. Hamid, M. Sugumaran and L. Journaux, "A fusion of feature extraction and feature selection technique for network intrusion detection," *International Journal of Security and Its Applications*, vol. 10, no. 8, pp. 151–158, 2016.

[11] S. Janakiraman and V. Vasudevan, "ACO based distributed intrusion detection system," *International Journal of Digital Content Technology and its Applications*, vol. 3, no. 1, pp. 66–72, 2009.

[12] G. Kim, S. Lee and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1690–1700, 2014.

[13] L. Koc, T. A. Mazzuchi and S. Sarkani, "A network intrusion detection system based on a hidden naive bayes multiclass classifier," *Expert Systems with Applications*, vol. 39, no. 18, pp. 13492–13500, 2012.

[14] H. J. Liao, C. H. R. Lin, Y. C. Lin and K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[15] 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.

[16] G. Liu, Z. Yi and S. Yang, "A hierarchical intrusion detection model based on the PCA neural networks," *Neurocomputing*, vol. 70, no. 7, pp. 1561–1568, 2007.

[17] K. Malialis and D. Kudenko, "Distributed response to network intrusions using multiagent reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 270–284, 2015.

[18] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, pp. 18-31, 2016.

[19] S. Mukkamala, A. H. Sung and A. Abraham, "Intrusion detection using an ensemble of intelligent paradigms," *Journal of network and computer applications*, vol. 28, no. 2, pp. 167–182, 2005.

[20] A. A. Orunsolu, A. S. Sodiya, A. T. Akinwale, B. I. Olajuwon, M. A. Alaran, O. O. Bamgboye, and O. A. Afolabi, "An empirical evaluation of security tips in phishing prevention: A case study of nigerian banks," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 25-39, 2017.

[21] A. A. Orunsolu, A. S. Sodiya, A. T. Akinwale, B. I. Olajuwon, "An anti-phishing kit scheme for secure web transactions," *International Journal of Electronics and Information Engineering*, vol. 6, no. 2, pp. 72–86, 2017.

[22] S. Peddabachigari, A. Abraham, C. Grosan and J. Thomas, "Modeling intrusion detection system using hybrid intelligent systems," *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 114–132, 2007.

[23] R. R. Reddy, "Network intrusion anomaly detection using radial basis function networks," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 3, pp. 1011–1014, 2017.

[24] S. Shamshirband, A. Patel, N. B. Anuar, M. L. M. Kiah and A. Abraham, "Cooperative game theoretic approach using fuzzy q-learning for detecting and preventing intrusions in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 228–241, 2014.

[25] A. Tayal, N. Mishra and S. Sharma, "Active monitoring & postmortem forensic analysis of network threats: A survey," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 49–59, 2017.

## Biography

**Yasir Hamid** received his Master's degree in Computer Applications from University of Kashmir in the year 2014. He is currently as a Ph.D Scholar in Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry. His areas of interests are Machine Learning, Network Security, Non Linear Dimension Reduction and data visualization.

**V. R. Balasaraswathi** recieved her M.E Degree in Computer Science from Anna University Chennai, currently she is working towards Ph.D degree at Dept. Computer Science and Engineering, Pondicherry Engineering College.

**Ludovic Journaux** received his PhD in image processing and computer sciences from the University of Burgundy (France) in 2006. He is currently working as associate professor at Agrosup Dijon and is a member of LE2I laboratory (UMR 6306) : Laboratory of Electronics, Computer Sciences and Images. His research interests include image processing, data mining, statistical analysis, artificial intelligence and classification.

**M. Sugumaran** received his M.Sc degree in mathematics from University of Madras in 1986, M.Tech degree in computer science and data processing from Indian Institute of Technology, Kharagpur, India in 1991 and obtained his Ph.D from Anna University, Chennai in 2008. He is currently working as Professor and Head of Computer Science and Engineering at Pondicherry Engineering College, India. His areas of interests are theoretical computer science, analysis of algorithms, parallel and distributed computing and spatial-temporal data.