CHAPTER 1

INTRODUCTION

1.1 Overview

The traffic through the network is heterogeneous and consists of flows from multiple applications and utilities. These applications are unique and will have their own requirements with respect to network parameters (e.g.: delay, jitter, etc). The quality and usability of these applications will be severely affected if these requirements are not satisfied. While meeting these requirements in a Local Area Network (LAN) with its huge bandwidth might be easy, but when considering WANs it usually is a challenge because of its bandwidth constraints.

Thus, traffic management on the WANs must exist in order to properly prioritize different applications across the limited WAN bandwidth and ensure that these requirements are met. In order to implement appropriate security policies, the network managers must have a detailed knowledge about applications and protocols. The user application may allow large delays or jitter, but the users might be very sensitive to long wait times. Managing network traffic requires a judicious balance of all these priorities.

Classification of traffic helps identify different applications and protocols that exist in a network. Different methods such as monitoring, discovery, control, and optimization can be performed on the identified traffic in order to improve the network performance. Typically, once the packets are classified (identified) as belonging to a particular application or protocol, they are marked or flagged. These markings or flags help the router determine appropriate service policies to be applied for those flows. All generic classification techniques based on Destination IP address, Source IP address, or IP protocol, etc. are limited in their ability as the inspection is limited to the IP header only. Similarly, classifying based on Layer 4 ports only is also limited[1]. The problem with this approach is that not all current applications use standard ports. Some applications even obfuscate themselves by using well the defined ports of other applications. Hence the Layer 4 port mechanism of application identification is not always reliable.

1.2 Self Learning Networks

Self Learning Networks (SLN) architecture is a solution that combines powerful analytics, using a wide set of machine learning technologies hosted on edge devices, along with advanced networking, to allow the network to become intelligent, adaptive, proactive, and predictive. The Cisco SLN architecture relies on the use of distributed, lightweight, yet complex analytic engines, referred to as Distributed Learning Agents (DLAs), implemented across the network. The DLA reports findings to the SLN Orchestrator, which provides orchestration and interaction among the distributed DLAs, as well as supports additional features to be developed in the future.

1.3 Deep Packet Inspection

Deep packet inspection (DPI) also called complete packet inspection is a form of computer network packet filtering that examines the data part (also the header) of a packet it passes an inspection point, searching for protocol noncompliance, viruses, spam, intrusions, or defined criteria to decide whether the packet may pass or if it needs to be routed to a different destination, or, for the purpose of collecting statistical information that functions at the Application layer of the OSI (Open Systems Interconnection model). Deep Packet Inspection (and filtering) enables advanced network management, user service, and security functions as well as internet data mining, eavesdropping, and internet censorship[5]. Although DPI has been used for Internet management for many years, some advocates of net neutrality fear that the technique may be used anticompetitive or to reduce the openness of the Internet.

1.4 Artificial Neural Networks

Artificial neural networks (ANNs) or connectionist systems are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as

"cat" or "no cat" and using the analytic results to identify cats in other images. They have found most use in applications difficult to express in a traditional computer algorithm using rule-based programming.

An ANN is based on a collection of connected units called artificial neurons (analogous to biological neurons in an animal brain). Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. In common ANN implementation, the synapse signal is simply a real number, and the output of each neuron is calculated by a non-linear function of the sum of all its input. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream. Further, they may have a threshold such that only if the aggregate signal is below (or above) that level is the downstream signal sent, typically neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times.

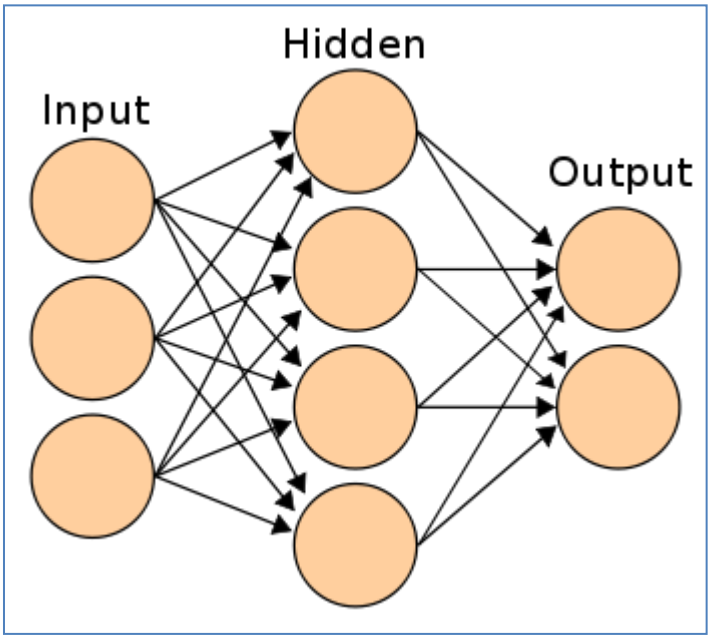

Figure 1.1 Artificial Neural Network

CHAPTER 2

LITERATURE SURVEY

2.1 Existing System

A number of researchers are looking particularly closely at the application of Machine Learning (ML) techniques (a subset of the wider Artificial Intelligence discipline) to IP traffic classification. The application of ML techniques involves a number of steps. First, features are defined by which future unknown IP traffic may be identified and differentiated. Features are attributes of flows calculated over multiple packets (such as maximum or minimum packet lengths in each direction, flow durations or inter-packet arrival times). Then ML classifier is trained to associate sets of features with known traffic classes (creating rules), and then the ML algorithm is applied to classify unknown traffic using previously learned rules. Every ML algorithm has a different approach to sorting and prioritizing sets of features, which leads to different dynamic behaviours during training and classification.

2.2 Port based IP traffic classification

TCP and UDP provide for the multiplexing of multiple flows between common IP endpoints through the use of port numbers. Historically many applications utilize a 'well known' port on their local host as a rendezvous point to which other hosts may initiate communication. A classifier in the middle of a network only have to look for TCP SYN packets (the first step in TCP's three-way handshake during session establishment) to know the server side of a new client-server TCP connection. The application is then inferred by looking up the TCP SYN packet's target port number in the Internet Assigned Numbers Authority (IANA)'s list of registered ports. UDP uses ports similarly (though without connection establishment nor the maintenance of connection state)[4].

However, this approach has limitations. An application may use ports other than its well known ports to avoid operating system access control restrictions (for example, non - privileged users on Unix-like systems may be forced to run HTTP servers on ports other than port 80.) Also, in some cases server ports are dynamically allocated as needed. For example,

the Real Video streamer allows the dynamic negotiation of the server port used for the data transfer.

2.3 Payload based IP traffic classification

To avoid total reliance on the semantics of port numbers, many current industry products utilise statefull reconstruction of session and application information from each packet's content. Payload based classification of P2P traffic (by examining the signatures of the traffic at the application level) could reduce false positives and false negatives to 5% of total bytes for most P2P protocols studied. The flows port number is examined when classification procedure starts. If no well-known port is used, the flow is passed through to the next stage. In the second stage, the first packet is examined to see whether it contains a known signature. If one is not found, then the packet is examined to see whether it contains a well-known protocol. If these tests fail, the protocol signatures in the first Kbytes of the flow are studied. Flows remaining unclassified after that stage will require inspection of the entire flow payload. Their results show that port information by itself is capable of correctly classifying 69% of the total bytes[4]. Including the information observed in the first Kbyte of each flow increases the accuracy to almost 79%. Higher accuracy (upto nearly 100%) can only be achieved by investigating the remaining unclassified flows entire payload .

2.4 Proposed System

The two main families of data mining are: supervised and unsupervised techniques. Supervised algorithms takes into account the availability of a training dataset in which each and every object is labelled, i.e., it is acts as a-priori associated to a particular class. Using this information suitable model is created describing groups of objects with the same label. Then, unlabelled objects can be classified, associated to a previously defined class, according to their features. For unsupervised algorithms, instead grouping is performed without any a-prior knowledge of labels. Groups of objects are clustered based on distance evaluated among samples, so that objects with similar features are part of the same cluster.

Supervised algorithms achieve high classification accuracy, provided that the training set is representative of the objects. However, labelled data may be difficult, or time-consuming to obtain. Semi-supervised classification addresses this issue by exploiting the information

available in unlabelled data to improve classifier performance. Many semi-supervised learning methods have been proposed unfortunately, no single method fits all problems. Both Labelled and unlabelled data are clustered by means of (variations of) known clustering algorithms (k-means and SOM). Then labelled data in each cluster is assigned labels to unlabelled data. Finally a new classifier is trained on the entire labelled dataset. While we exploit a different, iterative clustering approaches to group data. Due to its iterative refinement process, the approach adopted in "Self Learning Network Traffic Classification" is also particularly suited to model traffic flow changes, because it allows a seamless adaptation of the obtained traffic classes to traffic pattern evolution.

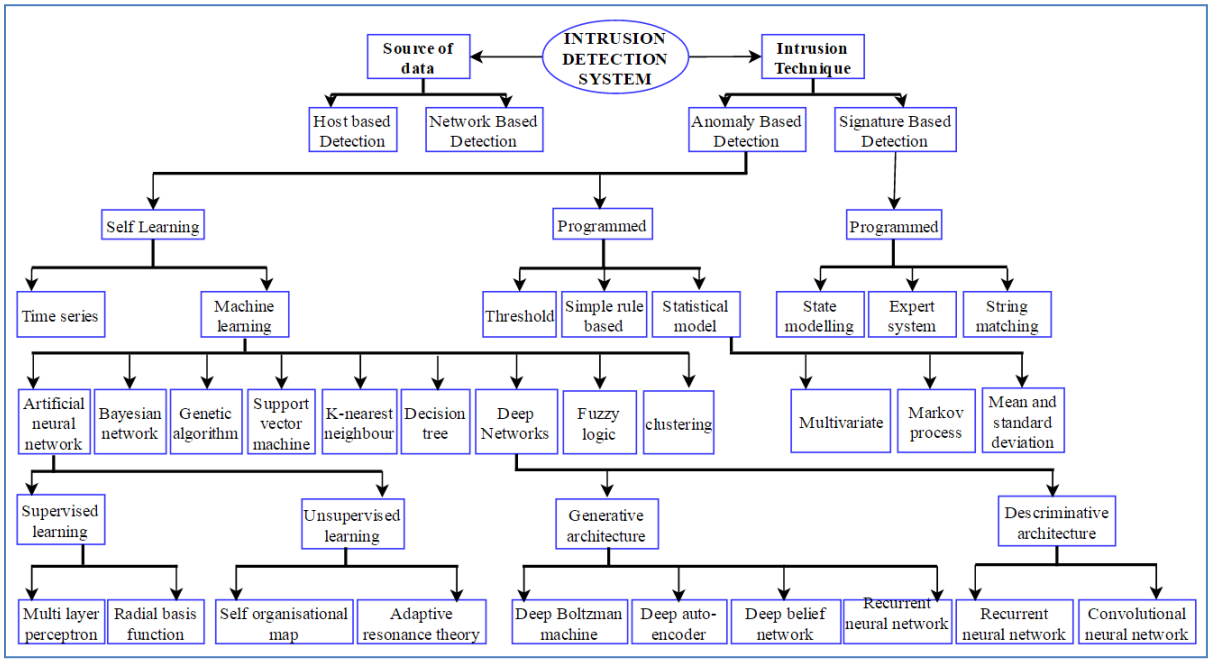

Figure 2.1 Classification of intrusion detection system

In the overall summary of literature survey is both payload based and port based IP traffic classification is not flexible with evolution in the fast growing networking technology and these methods significant drawback with false alarm rate in order to improve the performance of NIDS by using the advance technology such as Artificial Neural Networks and Deep Neural Networks in the networking domain.

CHAPTER 3

System Requirements and Specifications

3.1 Hardware Requirements

Hardware	Minimum Requirement
Processor	Pentium 4
RAM	2 GB
Hard-disk Space	5 GB
No of Cores	Dual core CPU

Table 3.1 Hardware Requirements

3.2 Software Requirements

Software	Minimum Requirement
Operating system	Windows 7 and above
MATLAB	Version above 2007
Weka	Version 3.8

Table 3.2 Software Requirements

CHAPTER 4

SYSTEM DESIGN

4.1 Design Architecture

System design is a process of problem solving and planning for a software solution. After the purpose and specification of software is determined, software developer will design or employ designers a plan for a solution. High level design gives an overview of the system flow. However, this describes a lot for the user to understand the logic. Here we see the basic knowledge about the system design and architecture.

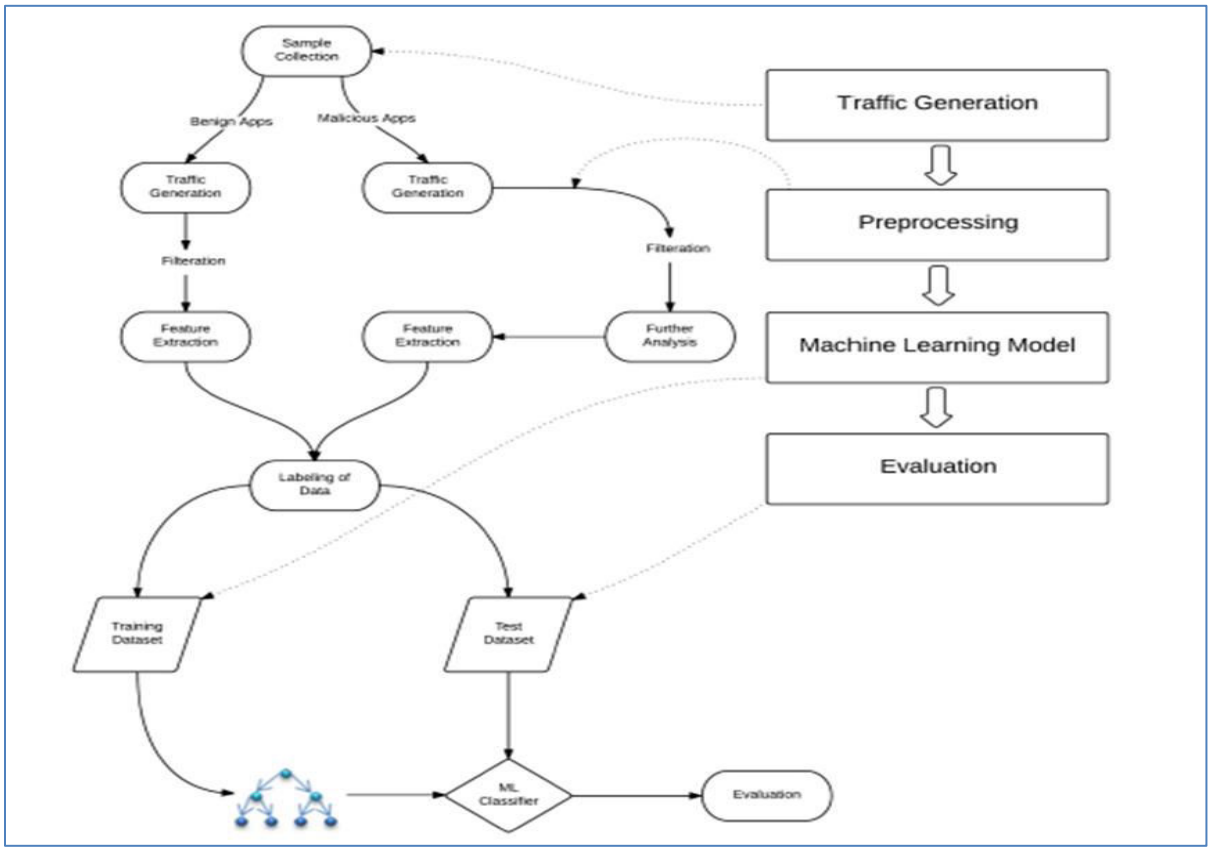

Figure 4.1 Model of Machine learning classification process

The problem of intrusion is gradually becoming nightmare for several organizations. To protect the valuable data of their clients, organizations implement security systems to detect and prevent security breaches. But since the intruders are using sophisticated techniques to

penetrate the systems, even the highly reputed secured systems have become vulnerable now[6].

4.2 Design Overview

To deal with the current scenario, advanced level of researches is required to be carried out to invent more sophisticated Intrusion Detection System (IDS). Among various methodologies, researchers consider Back Propagation Neural Network (BPNN) as a very effective and popular tool for developing IDS.

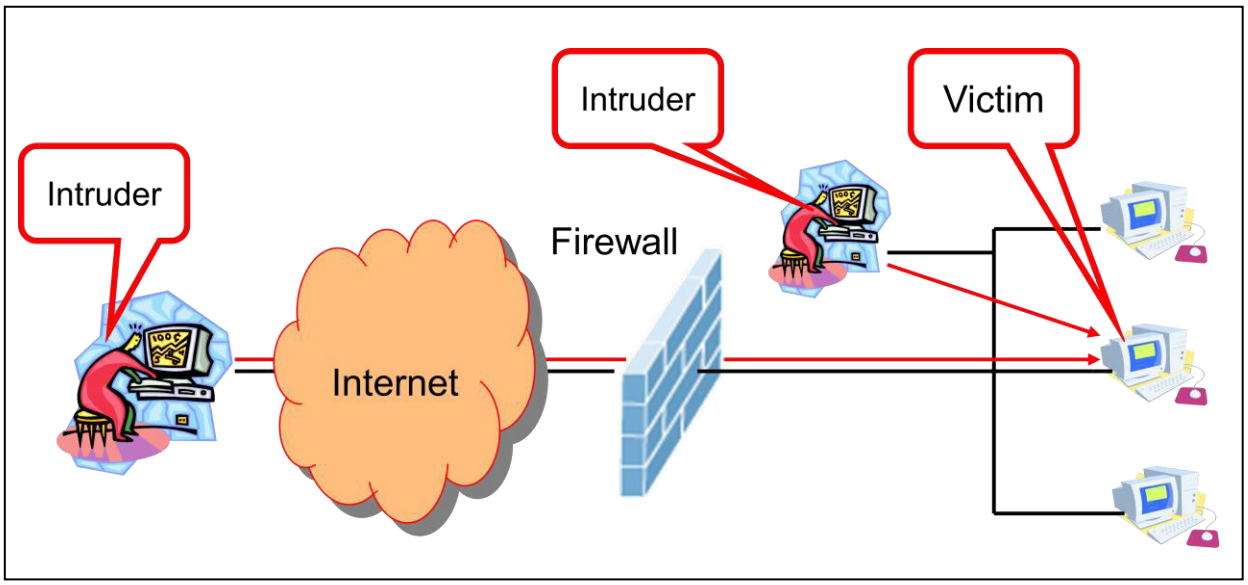

Figure 4.2 Real time intrusion detection system

The architecture addressed is a distributed IDS, where each node on the network will have an IDS agent running on it. The IDS agents on each node in the network work together via a cooperative intrusion detection algorithm to decide when and how the network is being attacked. Each Anomaly Detection Module is responsible for detecting a different type of anomaly. There can be from one to many Anomaly Detection Modules (ADM) on each mobile IDS agent, each working separately or cooperatively with other ADM. For example, one ADM might be looking for strange network traffic patterns, while another ADM might be watching user input speed.

CHAPTER 5

IMPLEMENTATION

5.1 Data Analysis

Data is collected from the connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

NSL-KDD dataset to detect attacks on four attack categories: Dos, Probe, R2L, and U2R.

1)**Dos[Denial of service] Attack**- In computing, a **denial-of-service attack** (**DoS attack**) is a cyber-attack where the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. Denial of service is typically accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled, the incoming traffic flooding the victim originates from many different sources. This effectively makes it impossible to stop the attack simply by blocking a single source.

2) **Probing Attack** –It is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls, it gains access to a computer and files using weak security point in the system mostly it will be internet security it simply keep sending some pings to the system as a normal packet but in background it captures all the data and gives it to the attacker.

3) **User to Root Attack** (U2R) – It is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.

4) **Remote to Local Attack** (R2L) – This attack occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that

machine exploits some vulnerability to gain local access as a user of that machine, the attacker will find vulnerable points in a computer or network's security software to access the machine or system. The main reasons for remote attacks are to view or steal data illegally, introduce viruses or other malicious software to another computer or network or system, and cause damage to the targeted computer or network[2].

The following are the advantages of the NSL-KDD over the original KDD data set:

1) It does not include redundant records in the training set, so the classifiers will not be biased towards more frequent records.

2) The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.

3) The numbers of records in the train and test sets is reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable[7].

feature name	***description***	***type***
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous

Table 5.1 Basic features of individual TCP connections

feature name	*description*	*Type*
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	Discrete
num_compromised	number of ``compromised'' conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	Discrete
su_attempted	1 if ``su root'' command attempted; 0 otherwise	Discrete
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	Discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	Discrete

Table 5.2 Content features within a connection suggested by domain knowledge

feature name	*description*	*Type*
count	number of connections to the same host as the current connection in the past two seconds	Continuous
	Note: The following features refer to these same-host connections.	
serror_rate	% of connections that have ``SYN'' errors	Continuous
rerror_rate	% of connections that have ``REJ'' errors	Continuous
same_srv_rate	% of connections to the same service	Continuous
diff_srv_rate	% of connections to different services	Continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	Continuous
	Note: The following features refer to these same-service connections.	
srv_serror_rate	% of connections that have ``SYN'' errors	Continuous
srv_rerror_rate	% of connections that have ``REJ'' errors	Continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Table 5.3 Traffic features computed using a two-second time window

5.1.1 Data Representation

After analyzing data found the following types of features in the NSL-KDD dataset.

Type	Features
Nominal	Protocol_type(2), Service(3), Flag(4)
Binary	Land(7), logged_in(12), root_shell(14), su_attempted(15), is_host_login(21),, is_guest_login(22)
Numeric	Duration(1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23) srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29) diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)

Table 5.4 Type of features

Following table used for the conversion of all nominal attributes to numeric attributes.

Type	Feature Name	Numeric Value
Attack or Normal	Normal	0
Attack or Normal	Attack	1
Protocol_type	TCP	2
	UDP	3
	ICMP	4
Flag	OTH	5
	REJ	6
	RSTO	7
	RSTOS0	8
	RSTR	9
	S0	10
	S1	11
	S2	12
	S3	13
	SF	14
	SH	15
Service	All services	16 to 81

Table 5.5 Nominal to numeric

Following shows the MATLAB code of converting feature name Normal to value 0 and feature name Attack to value 1.

```
for i=1:size(sdata,1)
   % sdata represents working dataset
   if((sdata{i,42}=='normal'))
      sdata.class(i) = '0'; %dataframe.variablename(row)=value%
   else
      sdata.class(i)= '1';
   end
end
```

5.1.2 Data Normalization

Data Normalization is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as feature scaling and is generally performed during the data preprocessing step. Since the range of values of raw data varies

widely, in some machine learning algorithms, objective functions will not work properly without normalization.

Minmax normalization is a normalization strategy which linearly transforms x to y.

Y = (X-min)/(max-min) where min and max are the minimum and maximum values in that particular vector, where X is the set of observed values in that vector.

It can be easily seen that when x=min, then y=0, and When x=max, then y=1. This means, the minimum value in X is mapped to 0 and the maximum value in X is mapped to 1. So, the entire ranges of values of X from min to max are mapped to the range 0 to 1.

```
s=size(sdata,1);  //To get the number of rows in a dataset to iterate
Max_range = 1;   //Maximum bound
Min_range = 0;   //Minimum bound
src_min=min(sdata.src_bytes);   //least value in the attribute src_bytes
src_max=max(sdata.src_bytes);  //highest value in the attribute src_bytes
for i=1:s  //To iterate
   p=sdata.src_bytes(i);  //first sample of src_bytes
   p=(src_max*((psrc_min)*(Max_rangeMin_range))/((src_max-src_min)))+Min_range;
end   //Finish
```

5.1.3 Feature Selection

Feature selection is the process of removing features from the original data set that are irrelevant with respect to the task that is to be performed. So not only the execution time of the classifier that processes the data reduces but also accuracy increases because irrelevant or redundant features can include noisy data affecting the classification accuracy negatively.

Due to the large amount of data processed, pattern recognition it's quite difficult whereas graphical representation it`s not possible, so PCA becomes a powerful tool for data analysis. Another advantage of PCA is that after finding the patterns, these are compressed, using dimensionality reduction, without any loss information[3].

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of distinct

principal components is equal to the smaller of the number of original variables or the number of observations minus one. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

Using weka tool found the following analysis after applying PCA.

=== Attribute Selection on all input data ===

Search Method:

Attribute ranking.

Attribute Evaluator (unsupervised):

Principal Components Attribute Transformer

Eigen value	**proportion**	**cumulative**	
7.37354	0.29494	0.29494	0.34129+0.32334-0.30238-0.30239-0.30125...
5.16652	0.20666	0.5016	0.40928+0.40927+0.40741+0.39840-0.22426...
2.06746	0.0827	0.5843	-0.5592-0.48236-0.35324-0.30237-0.25523...
1.87995	0.0752	0.6595	0.52824+0.38623-0.37337+0.35932-0.31431...
1.43559	0.05742	0.71692	0.43935-0.33537+0.3351+0.32930+0.3153...
1.1183	0.04473	0.76165	0.6171-0.50930+0.4685+0.1486-0.14135...
1.00046	0.04002	0.80167	-0.9556+0.2745+0.05931+0.0493+0.0461...

0.96493	0.0386	0.84027	0.8325-0.3261+0.31630+0.23 6-0.1236...
0.7468	0.02987	0.87014	-0.6783-0.54331+0.23 30-0.19632+0.19535...
0.66426	0.02657	0.89671	-0.60931+0.4363-0.3941-0.32230-0.27932...
0.55276	0.02211	0.91882	-0.50932+0.35 23+0.30833+0.30434+0.29724...
0.45662	0.01826	0.93709	0.64437-0.35236+0.31832-0.25831+0.2571...
0.44572	0.01783	0.95492	-0.63335+0.38730+0.33136-0.31937-0.20532...

Table 5.6 PCA Analysis

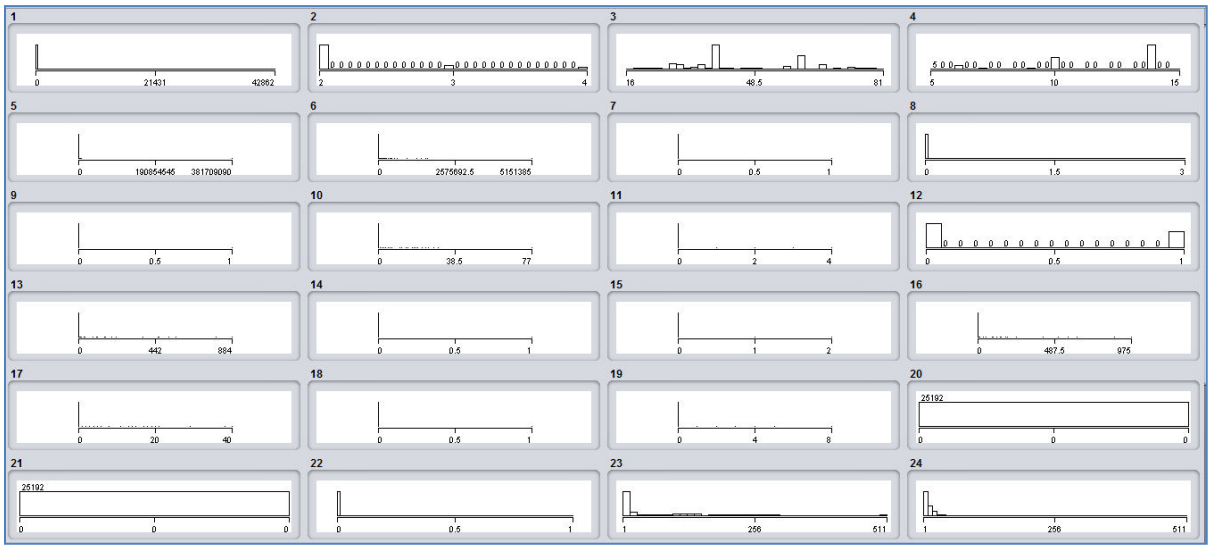

Figure 5.1 Snapshot of Variation in a attributes

After performing PCA to dataset choosing the attribute which is having high eigen value in the overall attribute vector and then one with high variance and uncorrelated feature are extracted from the original dataset following snapshot shows the selected features.

	A	B	C	D	E	F	G	H	I	J	K
1	0	2	35	14	491	0	2	2	150	25	0
2	0	3	56	14	146	0	13	1	255	1	0
3	0	2	61	10	0	0	123	6	255	26	1
4	0	2	38	14	232	8153	5	5	30	255	0
5	0	2	38	14	199	420	30	32	255	255	0
6	0	2	61	6	0	0	121	19	255	19	1
7	0	2	61	10	0	0	166	9	255	9	1
8	0	2	61	10	0	0	117	16	255	15	1
9	0	2	63	10	0	0	270	23	255	23	1
10	0	2.00	61	10	0	0	133	8	255	13	1
11	0	2	61	6	0	0	205	12	255	12	1
12	0	2	61	10	0	0	199	3	255	13	1
13	0	2	38	14	287	2251	3	7	8	219	0
14	0	2	35	14	334	0	2	2	2	20	1
15	0	2	48	10	0	0	233	1	255	1	1

Figure 5.2 Snapshot of selected features

5.2 Algorithm workflow

Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times.

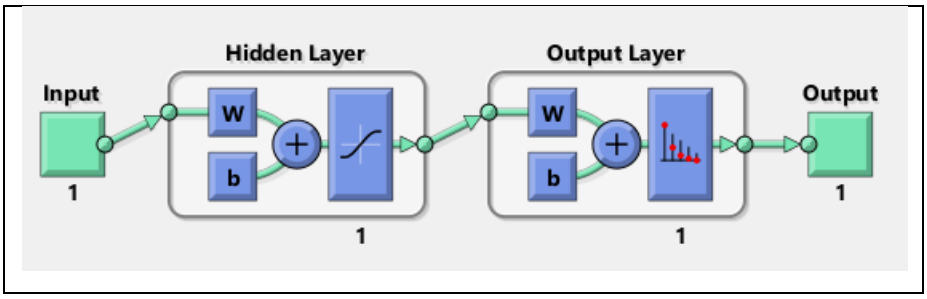

Figure 5.3 Snapshot of Neural Network

It is a two layer feed-forward network with sigmoid function at the hidden neuron and softmax function at the output neuron ,It can classify the vectors arbitrarily well given enough number of neurons at the hidden layer, The network will be trained with Scaled Conjugate Gradient Backpropogation Algorithm.

5.2.1 Neuron Model

An elementary neuron with *R* inputs is shown below. Each input is weighted with an appropriate *w*. The sum of the weighted inputs and the bias forms the input to the transfer function *f*. Neurons can use any differentiable transfer function *f* to generate their output.

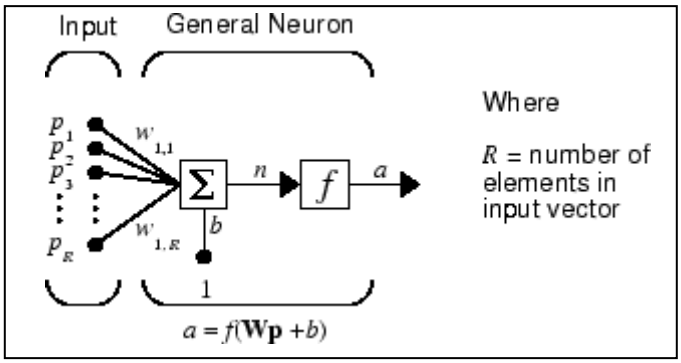

Figure 5.4 Basic Neuron Model

Neural network is an adjustable model of outputs as functions of inputs. It consists of several layers:

- Input layer, which consists of input data
- Hidden layer, which consists of processing nodes called neurons
- Output layer, which consists of one or several neurons, whose outputs are the network outputs.

All nodes of adjacent layers are interconnected. These connections are called synapses. Every synapse has an assigned scaling coefficient, by which the data propagated through the synapse is multiplied. These scaling coefficient are called weights (w[i][j][k]). In a Feed-Forward Neural Network (FFNN) the data is propagated from inputs to the outputs.

Feed-forward networks have the following characteristics:

1. Perceptrons are arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world, and hence are called hidden layers.
2. Each perceptron in one layer is connected to every perceptron on the next layer. Hence information is constantly "fed forward" from one layer to the next, and this explains why these networks are called feed-forward networks.
3. There is no connection among perceptron in the same layer.

5.2.2 Sigmoid Function

A sigmoid function is a bounded differentiable real function that is defined for all real input values and has a positive derivative at each point.

A sigmoid function is a mathematical function having an "S" shaped curve (sigmoid curve). Often, *sigmoid function* refers to the special case of the logistic function shown in the below figure and defined by the formula

$$sigm(x) = \frac{1}{1+e^{-x}}$$

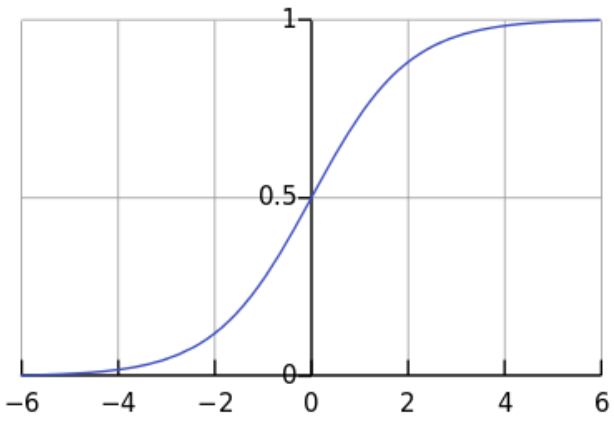

Figure 5.5 Sigmoid curve

The activation threshold of these functions is x=0. This threshold can be moved along the x axis thanks to an additional input of each neuron, called the bias input, which also has a weight assigned to it.

The number of inputs, outputs, hidden layers, neurons in these layers, and the values of the synapse weights completely describe a FFNN, i.e. the nonlinear model that it creates. In order to find weights the network must be trained. During a supervised training, several sets of past inputs and the corresponding expected outputs are fed to the network. The weights are optimized to achieve the smallest error between the network outputs and the expected outputs. The simplest method of weight optimization is the back-propagation of errors, which is a gradient descent method.

5.2.3 Softmax function

In mathematics,the softmax function or normalized exponential function is a generalization of the logistic function that "squashes" a *K*-dimensional vector Z of arbitrary real values to a *K*-dimensional vector of real values in the range [0, 1] that add up to 1. The function is given by

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

In probability theory, the output of the softmax function can be used to represent a categorical distribution – that is, a probability distribution over *K* different possible outcomes.

The softmax function is often used in the final layer of a neural network-based classifier. Such networks are commonly trained under a log loss (or cross-entropy) regime, giving a non-linear variant of multinomial logistic regression.

Since the function maps a vector and a specific index *i* to a real value, the derivative needs to take the index into account:

$$\frac{\partial}{\partial q_k}\sigma(\mathbf{q}, i) = \cdots = \sigma(\mathbf{q}, i)(\delta_{ik} - \sigma(\mathbf{q}, k))$$

5.2.4 Conjugate Gradient Algorithm

The basic backpropagation algorithm adjusts the weights in the steepest descent direction (negative of the gradient). This is the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. And in this a learning rate is used to determine the length of the weight update (step size). In most of the conjugate gradient algorithms, the step size is adjusted at each iteration. A search is made along the

conjugate gradient direction to determine the step size, which minimizes the performance function along that line.

It involves general purpose second order techniques that help minimize goal functions of several variables, Second order means that method uses second order derivatives of the goal function, while first-order techniques like standard backpropagation only use the first derivatives. A second order technique generally finds a better way to a (local) minimum than a first order technique, but at a higher computational cost.

Like standard backpropagation, CGMs iteratively try to get closer to the minimum. But while standard backpropagation always proceeds down the gradient of the error function, a conjugate gradient method will proceed in a direction which is conjugate to the directions of the previous steps. Thus the minimization performed in one step is not partially undone by the next, as it is the case with standard backpropagation and other gradient descent methods.

Scaled Conjugate Gradient Backpropogation Algorithm

SCG is a supervised learning algorithm for feedforward neural networks, and is a member of the class of conjugate gradient methods.

Line Search

In optimization, the line search strategy is one of the two basic iterative approaches to find a local minimum X*, of an objective function f: $R^n \rightarrow R$.other approach is trust region.The line search approach first finds a descent direction along which the objective function f will be reduced and then computes a step size that determines how far X* should move along that direction.

Line search algorithm in Conjugate Gradient methods:-

1. Set iteration counter $k = 0$, and make an initial guess $\mathbf{x}_0$ for the minimum
2. Repeat:
3. Compute a descent direction $\mathbf{p}_k$
4. Choose α_k to 'loosely' minimize $h(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{p}_k)$ over $\alpha \in \mathbb{R}_+$
5. Update $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$, and $k = k + 1$
6. Until $\|\nabla f(\mathbf{x}_k)\| <$ tolerance

Trust Region

Trust region is a term used in mathematical optimization to denote the subset of the region of the objective function that is approximated using a model function (often a quadratic). If an adequate model of the objective function is found within the trust region then the region is expanded; conversely, if the approximation is poor then the region is contracted. Trust region methods are also known as restricted step methods.

In conjugate gradient algorithms requires a line search at each iteration. This line search is computationally expensive, since it requires that the network response to all training inputs be computed several times for each search. The scaled conjugate gradient algorithm (SCG), developed by Moller was designed to avoid the time-consuming line search.

Scaled Conjugate Gradient Back propagation Learning Algorithm

step 1: Normalize the inputs and outputs with respect to their maximum values. it is proved that the neural networks work better if input and outputs lie between 0-1 for each training pair is in a normalized form.

step 2: Assume the number of neurons in the hidden layer to lie between $1<m<2l$.

step 3: [V] Represents the weights of synapses connecting input neurons to hidden neurons and [w] represents weights of synapses connecting hidden neurons to output neurons. Initialize the weights to small random values usually from -1 to 1. for general problems, λ can be assumed as 1 and the threshold values can be taken as zero.

$[V]^o$ ={random weights}

$[W]^o$ = {random weights}

$[\Delta V]^o = [\Delta W]^o = [o]$

step 4: for the training data, represent one set of inputs and outputs, Present the pattern to the input layer {I}, as inputs to the input layer may be evaluated as $\{O\}_r = \{I\}_I$ for (l*1).

step 5: Compute the inputs to the hidden layer by multiplying corresponding weights of synapses as $\{I\}_H = \{V\}^T\{O\}_I$ where I and V sets are m*1 and and O is l*1.

step 6 : Let the hidden layer units evaluate the output using the sigmoidal function as

$\{O\}H = \{- - -1/(1+e^{(-I_{Hi})})- - -\}$

step 7 : Compute the inputs to the output layer by multiplying corresponding weights of synapses as $\{I\}o = \{w\}^T\{O\}_H$ where I is set of n*1,w is set of n*m matrix and O is set of m*1 matrix

step 8 : Let the output layer units evaluate the output using softmax function as $\{O\}_O = \{- - - ((e^{(i)}/ \sum_{k=1}^{k} e^i_k ---\}$this is the network output.

step 9 : Calculate the error and the difference between the network output and the desired output as for the ith training set as $E^P = \sqrt{\Sigma((Tj - Ooj)2)}/n$

step 10 : Find {d} as $\{d\} = \{(T_K\text{-}O_{ok})O_{ok}(1\text{-}O_{ok})\}$ of matrix n*1

step 11 : Find {Y} matrix as $[Y] = \{O\}_H$ <d> where Y is matrix of m*n, o is matrix of m*1, d is matrix of 1*n.

step 12 : Find $[\Delta W]_{t+1} = \alpha[\Delta w]_t + \eta[Y]$ where each matrix is of m*n.

step 13 : Find {e} = [w] {d} where e is matrix of m*1, w is matrix of m*n, and d is matrix of n*1 i.e $\{d^*\} = \{ei(O_{Hi})(1\text{-}O_{Hi})\}$ where d is matrix of m*1

step 14: for finding X matrix as $[x] = \{O\}_r\langle d^*\rangle = \{I\}_I\langle d^*\rangle$ where x = matrix of 1*m, O,I = single value, d =matrix of 1*m

step 15: Find $[\Delta V]^{t+1} = \alpha[\Delta V]^t + \eta[X]$ all matrix of 1*m

step 16: To Find the updated weights and threshold[Θ]

$[V]^{t+1} = [V]^t + [\Delta V]^{t+1}$

$[W]^{t+1} = [W]^t + [\Delta W]^{t+1}$

$[\Theta]_o t+1 = [\Theta]_o t + [\Delta\Theta]_o t+1$

$[\Theta]_H t+1 = [\Theta]_H t + [\Delta\Theta]_H t+1$

step 17: Find error rate as Error rate = ((ΣEP)/nset)

step 18 : Repeat steps 4-18 untill the convergences in the error rate is less than the tolerance value[2].

Complexity of SCG

The number of epochs is not relevant when comparing SCG to other algorithms like standard backpropagation, Indeed one iteration in SCG needs the computation of two gradients, and one call to the error function, while one iteration in standard backpropagation needs the computation of one gradient and one call to the error function. Moller defines a *complexity*

unit (cu) to be equivalent to the complexity of one forward passing of all patterns in the training set. Then computing the error costs 1 cu while computing the gradient can be estimated to cost 3 cu. According to Møller's metric, one iteration of SCG is as complex as around 10^{-16} iterations of standard backpropagation[2].

Following MATLAB code shows the implementation of above explained algorithm.

```
function [Y,Xf,Af] = NeuralNetworkFunction(X,~,~)

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = 0;
x1_step1.gain = 2;
x1_step1.ymin = -1;

% Layer 1
b1 = [-13.324760466907521916;-
10.900504512945476066;7.7777516391799013107;4.6666666664439002687;1.5555555556
795430672;1.5555555556026188224;4.6666666539768257849;7.7777984863246585334;-
10.898343380655127532;15.165303780350770424];
IW1_1 = [14.675239533092478084;13.988384375943413218;-14.000026138597876368;-
14.000000000222767582;-
13.999999999875473833;13.999999999952766672;14.000000012689840291;13.9999792914
53120922;-13.990545508233759975;12.834696219649229576];

% Layer 2
b2 = -0.12315652314979817772;
LW2_1 = [1.9579826377084112998 1.6603031876655682808 -4.0253905328637697991 -
2.7792172905021814699 -1.2636267177841775666 2.2495251525014055005
3.0539400757714054713 1.8665795616648681587 -1.9427155124482371473
0.52085338496705602118];

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
End

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
```

```
    Q = size(X{1},1); % samples/series
Else
    Q = 0;
End

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = logsig_apply(repmat(b2,1,Q) + LW2_1*a1);

    % Output 1
    Y{1,ts} = a2;
    Y{1,ts} = Y{1,ts}';
End

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
End
End

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
End

% Sigmoid Positive Transfer Function
```

```
function a = logsig_apply(n,~)
a = 1 ./ (1 + exp(-n));
End

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
End
```

Parameters used for trainscg is epochs, show, goal, time, min_grad, max_fail, sigma, lambda. The parameter sigma determines the change in the weight for the second derivative approximation. The parameter lambda regulates the indefiniteness of the Hessian.

During training divide the input dataset for three major steps involves in the machine learning process:-

Training set - to fit the parameters [i.e., weights]

Validation set - to tune the parameters [i.e., architecture]

Test set -to assess the performance [i.e., generalization and predictive power]

Following MATLAB Code shows the above explanation

```
Net=fitnet(Number of hidden neurons,Dataset); //To create neural network

Net.divideParam.trainRatio=0.7;        //Percentage of dataset used in the training of whole dataset

Net.devideParam.valRatio=0.15;    // Percentage of dataset used in the validation of whole dataset

Net.divideParam.testRatio=0.15;   // Percentage of dataset used in the testing of whole dataset

[net,performance]=train(net,Input dataset,Targer dataset);   //Learning with trained dataset
```

CHAPTER 6

TESTING

Testing systems that don't always return the same answers require new approaches. This is especially true when testing systems whose responses adapt to what they have learned from previous training. Software testing, in theory, is a fairly straightforward activity. For every input, there should be a defined and known output. We enter values, make selections, or navigate an application and compare the actual result with the expected one.

But there is a type of software where having a defined output is no longer the case. Actually, two types. One is machine learning systems; the second is predictive analytics. Most machine learning systems are based on neural networks. A neural network is a set of layered algorithms whose variables can be adjusted via a learning process. The learning process involves using known data inputs to create outputs that are then compared with known results. When the algorithms reflect the known results with the desired degree of accuracy, the algebraic coefficients are frozen and production code is generated.

Selected features are applied to three different machine learning methods to measure the accuracy of classification as follows:-

6.1 Tree Based Classifier

Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making).

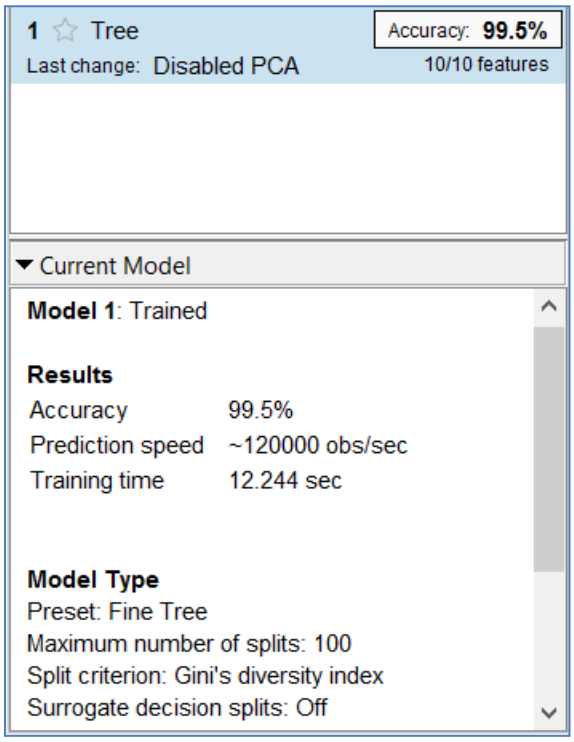

Figure 6.1 Snapshot of Tree classifier results

Above figure shows the 99.5% accuracy of classification after applying Decision tree based classifier .

Limitation of Tree classifier:-

- Trees do not tend to be as accurate as other approaches.
- Trees can be very non-robust. A small change in the training data can result in a big change in the tree, and thus a big change in final predictions.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristics such as the greedy algorithm where locally-optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally-optimal decision tree. To reduce the greedy effect of local-optimality some methods such as the dual information distance (DID) tree were proposed.

- Decision-tree learners can create over-complex trees that do not generalize well from the training data. (This is known as overfitting) Mechanisms such as pruning are necessary to

avoid this problem (with the exception of some algorithms such as the Conditional Inference approach, that does not require pruning).

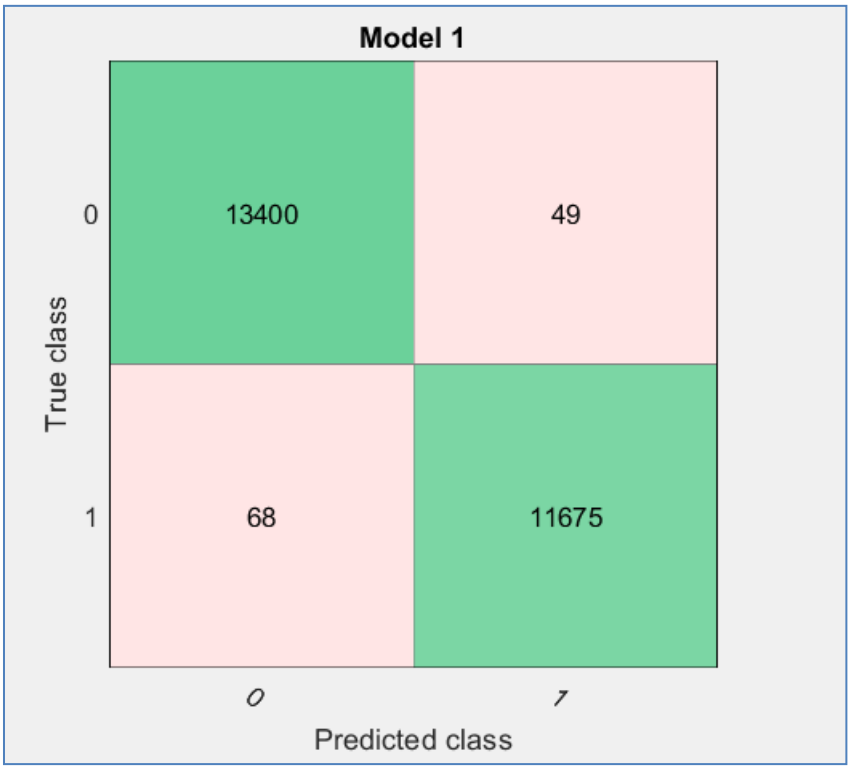

Figure 6.2 Snapshot of Tree classifier confusion matrix

Above figure 6.2 shows the confusion matrix of tree classifier represents the number of samples misclassified verses number of samples correctly classified.in this overall 25192 samples 49 normal class misclassified as attack class and 68 attack class misclassified as normal class.

In statistics, a receiver operating characteristic curve, i.e. ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate(FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning. The false-positive rate is also known as the fall-outor probability of false alarm and can be calculated as (1 − specificity). The ROC curve is thus the sensitivity as a function of fall-out. In general, if the probability distributions for both detection and false alarm are known, the ROC curve can be generated by plotting the cumulative distribution function (area under the probability

distribution from to the discrimination threshold) of the detection probability in the y-axis versus the cumulative distribution function of the false-alarm probability on the x-axis.

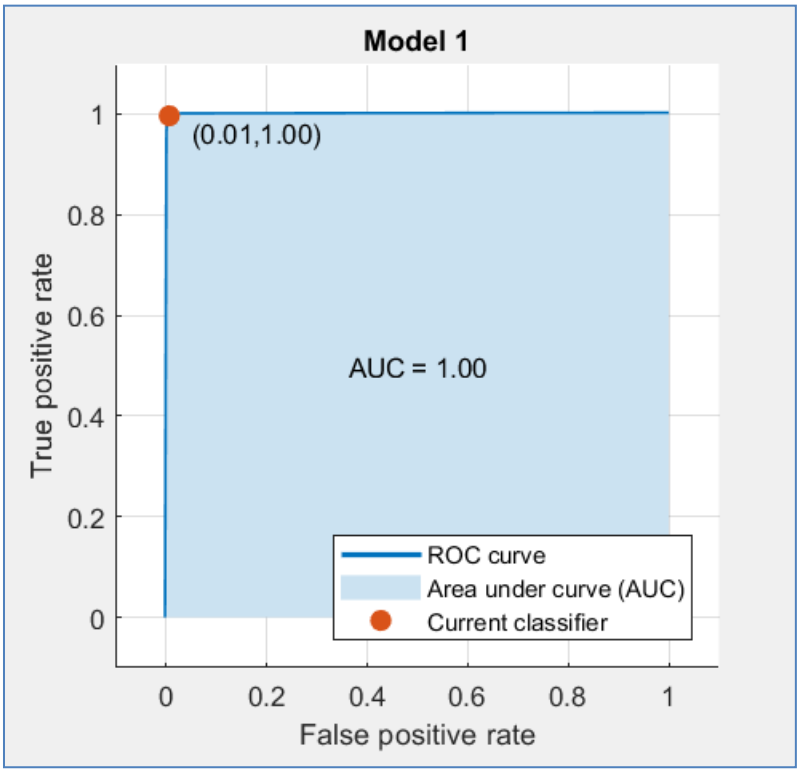

Figure 6.3 Snapshot of Tree classifier ROC curve

In the above figure shows the ROC curve of tree based classifier area under the curve is going close to maximum amount but ROC curve gives the error rate of 0.01% as false alarm rate.

6.2 Logistic Regression

In statistics, logistic regression, or logit regression, or logit model is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

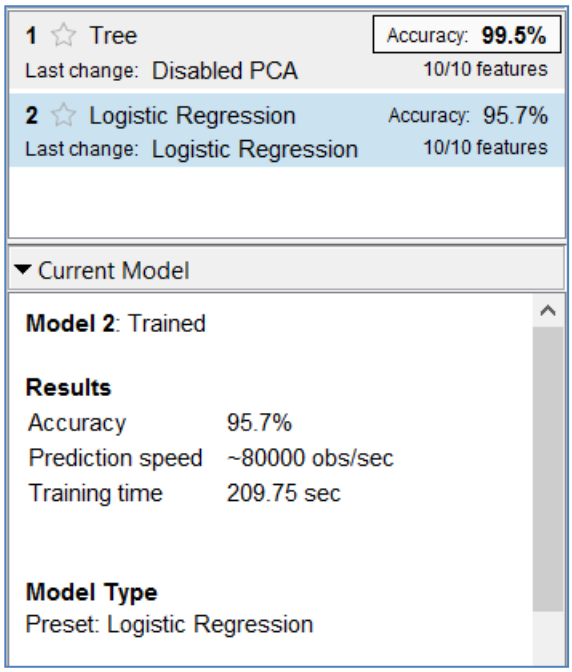

Figure 6.4 Snapshot of Logistic Regression classifier performance

Above figure shows the 95.7% accuracy of classification after applying Logistic Regression based classifier. compare to tree its giving less accuracy because of number of dependent variable is very less in the chosen dataset.

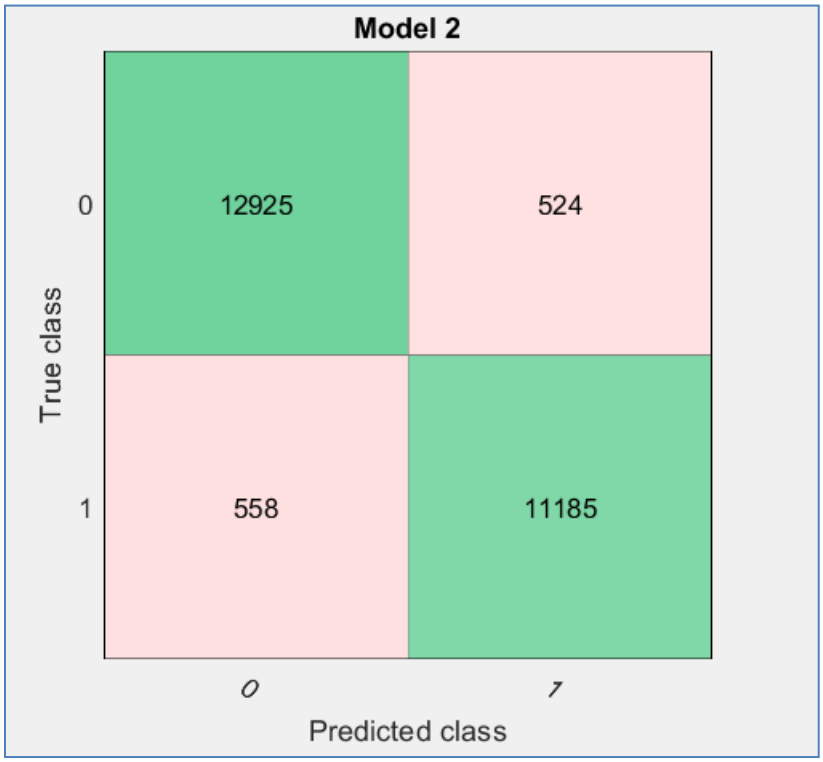

Figure 6.5 Snapshot of Logistic Regression classifier confusion matrix

Above figure 6.5 shows the confusion matrix of logistic regression classifier represents the number of samples misclassified verses number of samples correctly classified. in this overall 25192 samples 524 normal class misclassified as attack class and 558 attack class misclassified as normal class.

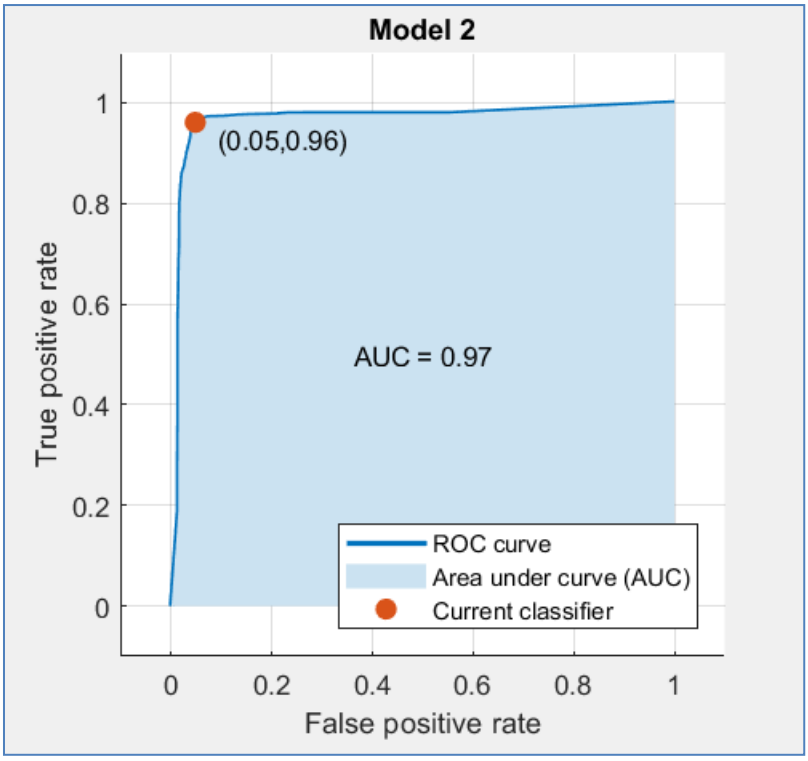

Figure 6.6 Snapshot of Logistic Regression classifier ROC curve

In the above figure shows the ROC curve of logistic regression based classifier area under the curve is 0.97 amount but ROC curve gives the error rate of 4% as false alarm rate.

6.3 Support Vector Machine

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting)[8]. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that

is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

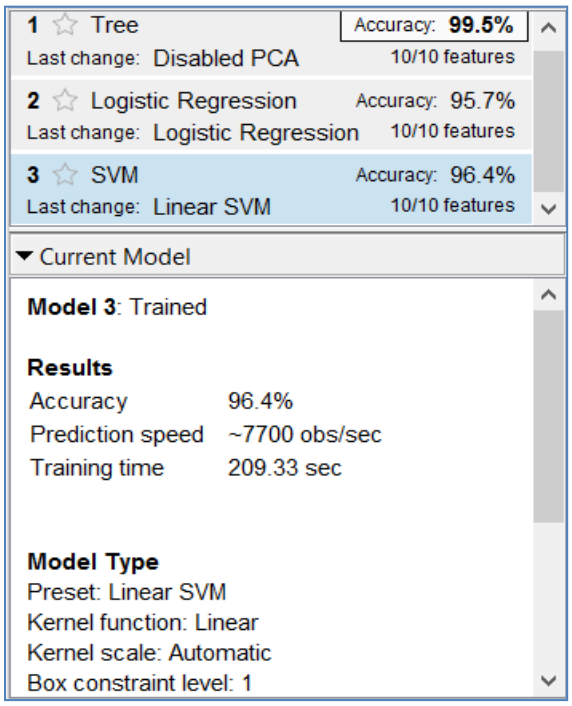

Figure 6.7 Snapshot of SVM classifier performance

Above figure shows the 96.4% accuracy of classification after applying SVM based classifier. compare to logistic regression it is better SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

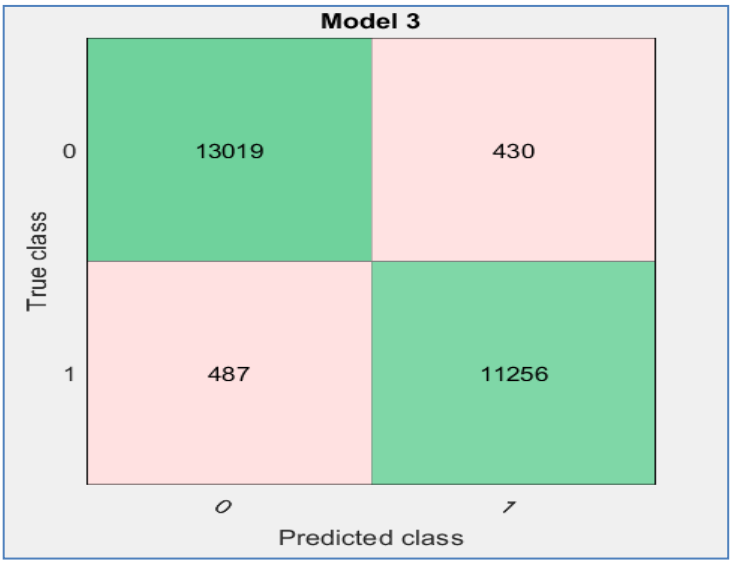

Figure 6.8 Snapshot of SVM classifier confusion matrix

Above figure 6.7 shows the confusion matrix of SVM classifier represents the number of samples misclassified verses number of samples correctly classified. in this overall 25192 samples 430 normal class misclassified as attack class and 487 attack class misclassified as normal class.

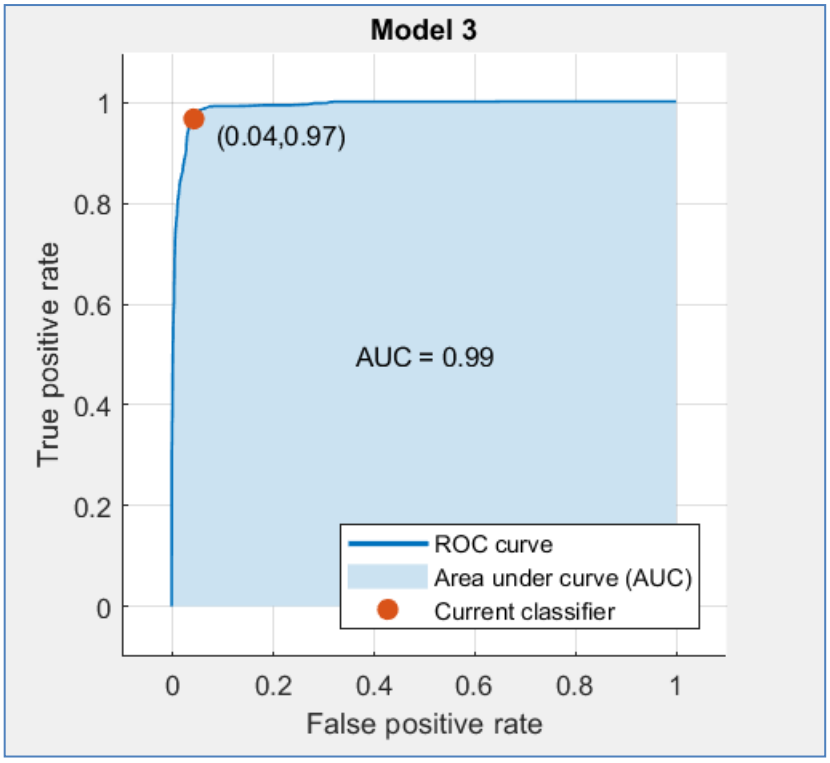

Figure 6.9 Snapshot of SVM classifier ROC curve

In the above figure shows the ROC curve of SVM based classifier area under the curve is 0.99 amount but ROC curve gives the error rate of 3% as false alarm rate.

CHAPTER 7

RESULTS AND SCREEN SHOTS

1) If needed, open the Neural Network Start GUI with this command:

nnstart

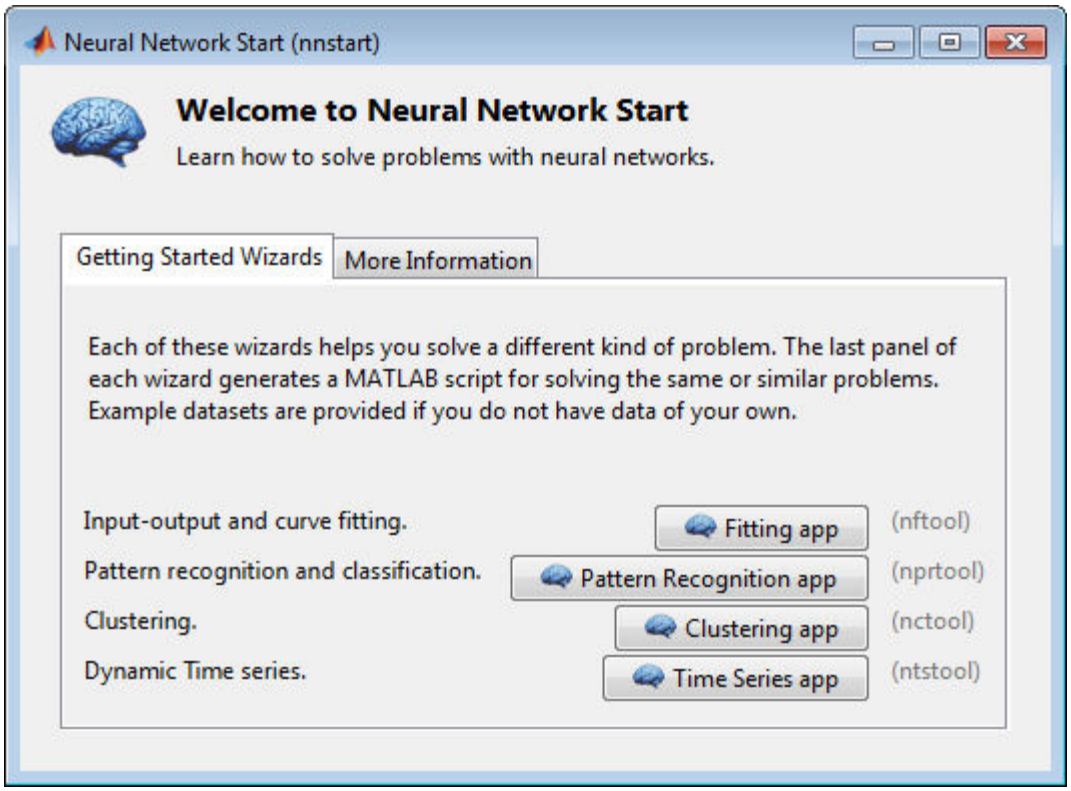

Figure 7.1 Snapshot of nnstart window in MATLAB

Curve fitting is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points, possibly subject to constraints.

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

2) Click Pattern Recognition app to open the Neural Network Pattern Recognition app. (You can also use the command nprtool.)

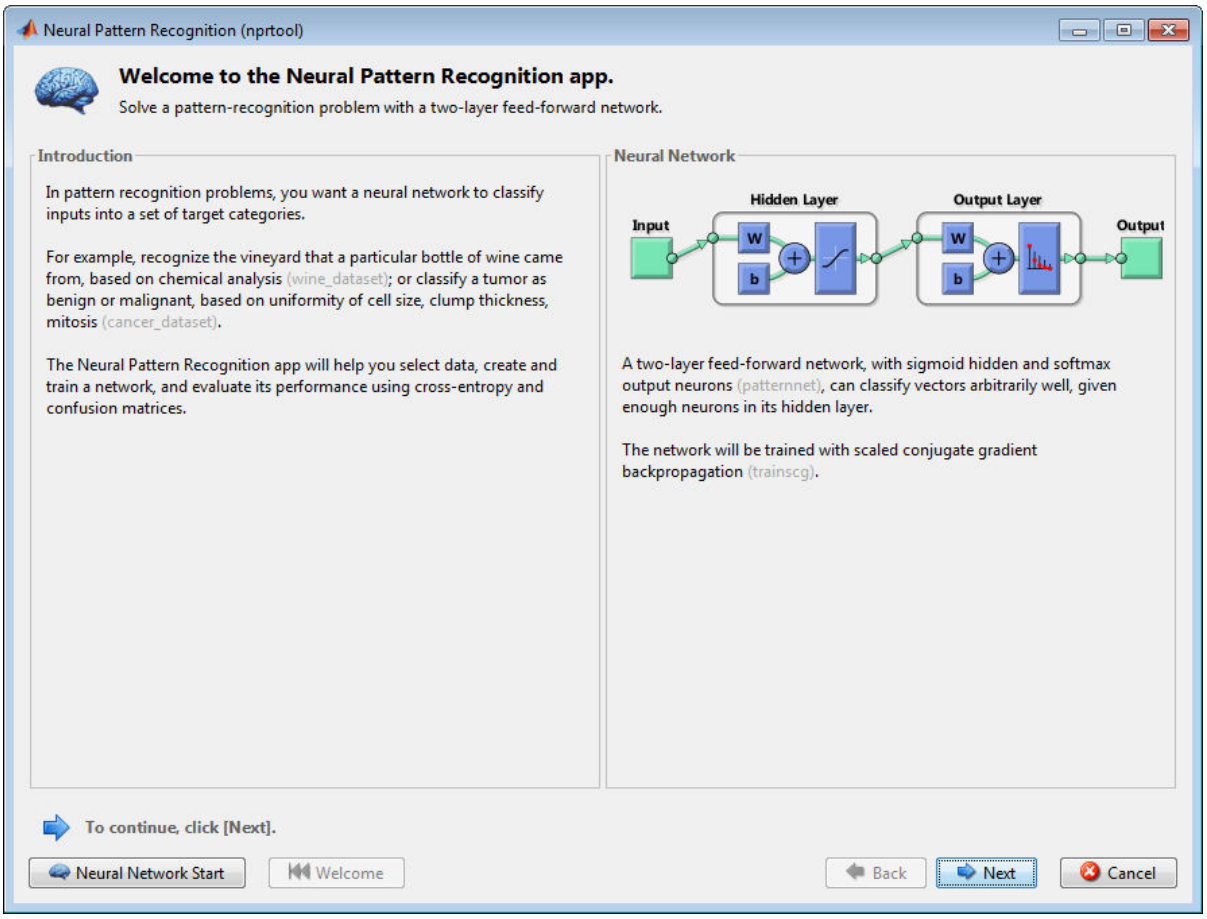

Figure 7.2 Snapshot of nprtool window in MATLAB

An important application of neural networks is pattern recognition. Pattern recognition can be implemented by using a feed-forward neural network that has been trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern.

3) Click Next to proceed. The Select Data window opens.

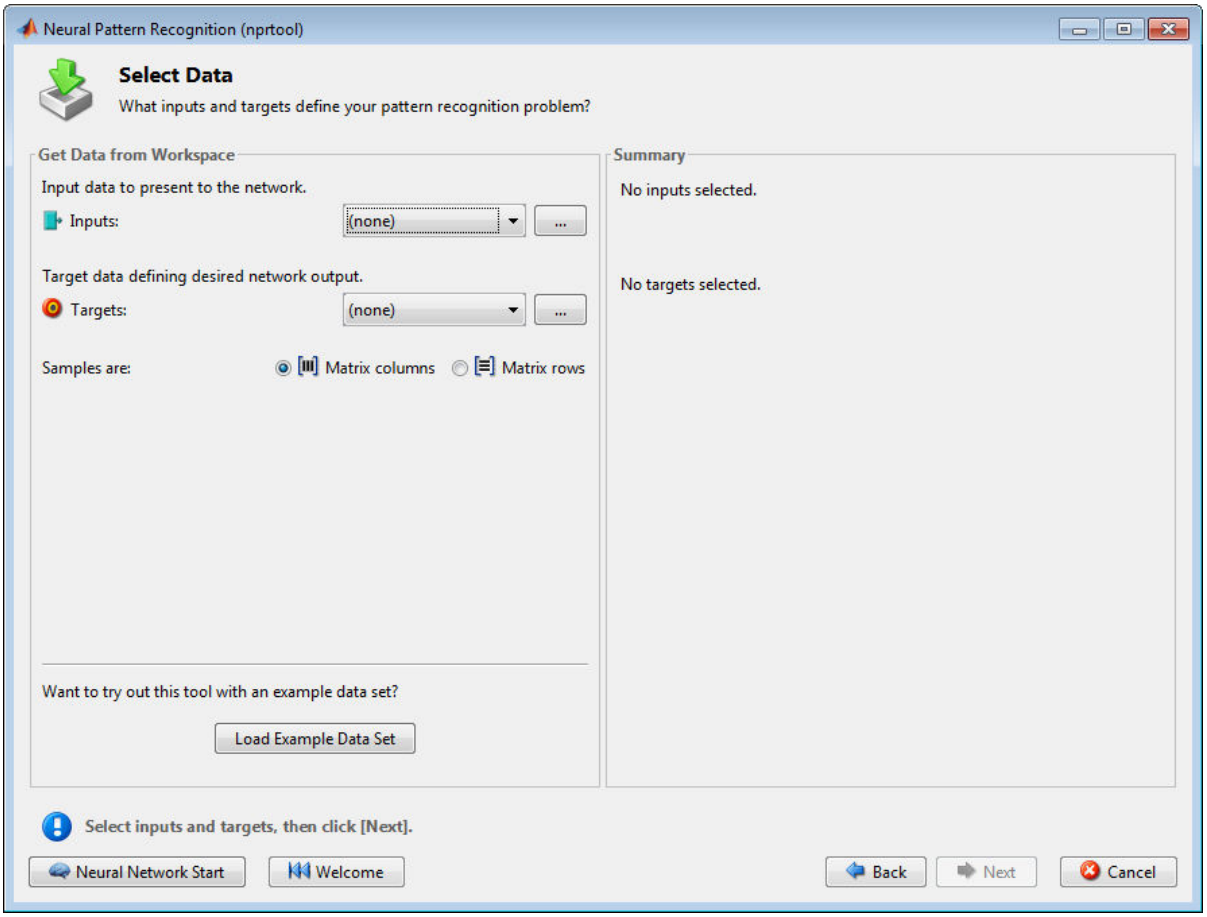

Figure 7.3 Snapshot of matlab window to load the dataset

Above snapshot shows the training data to be load on to the Neural network MATLAB platform and also load the target dataset so that after training during prediction it checks its learning accuracy based on the target value.

In this MATLAB we can have option to make our sample as rows in the dataset and columns in the dataset and in our NSL-KDD dataset considered one complete set of rows as one sample.

4) Click Next to continue to the Validation and Test Data window.

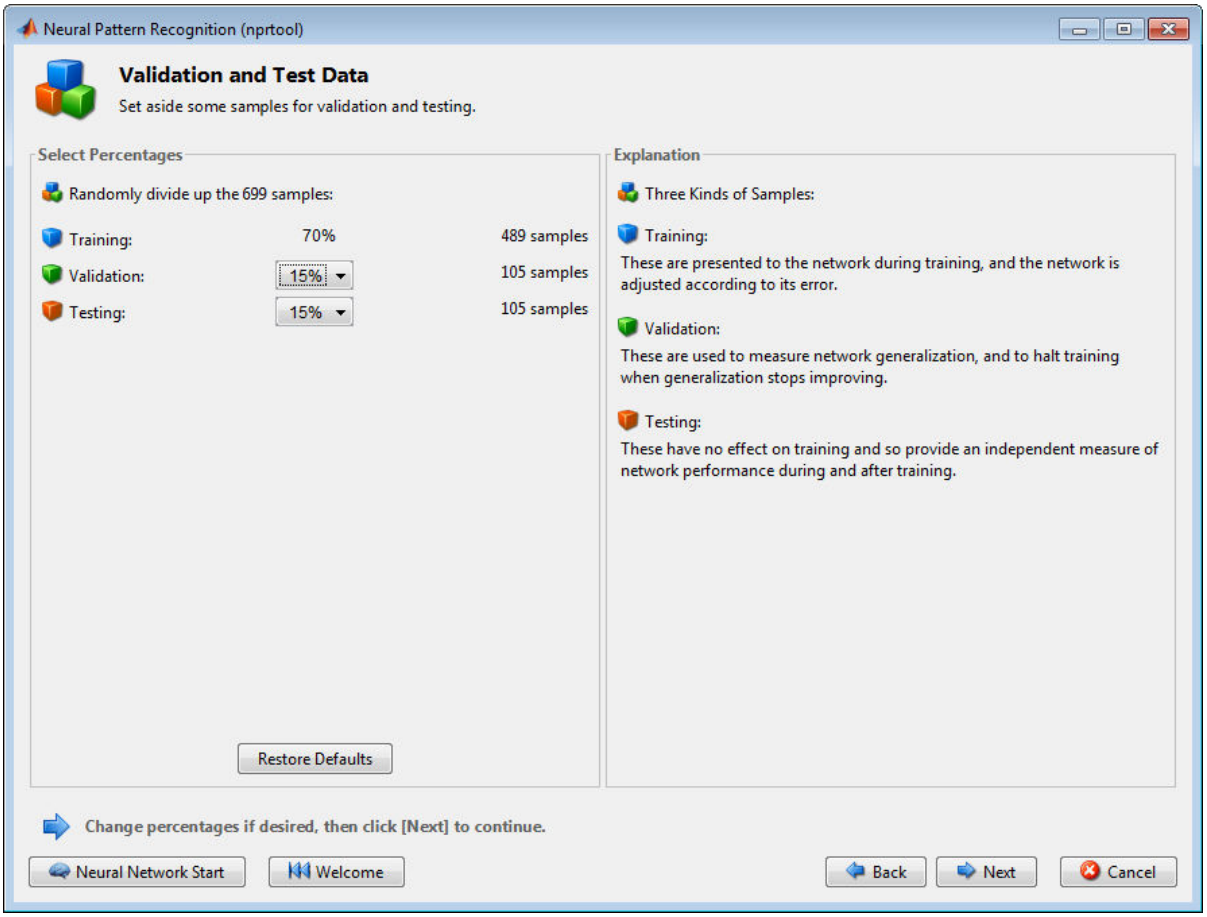

Figure 7.4 Snapshot of matlab window to divide the dataset

Validation and test data sets are each set to 15% of the original data. With these settings, the input vectors and target vectors will be randomly divided into three sets as follows:

- 70% are used for training.
- 15% are used to validate that the network is generalizing and to stop training before overfitting.
- The last 15% are used as a completely independent test of network generalization.

5) Click Next.

The standard network that is used for pattern recognition is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer, and a softmax transfer function in the output layer. The default number of hidden neurons is set to 10. You might want to come back and increase this number if the network does not perform as well as you expect. The number of output neurons is set to 2, which is equal to the number of elements in the target vector (the number of categories).

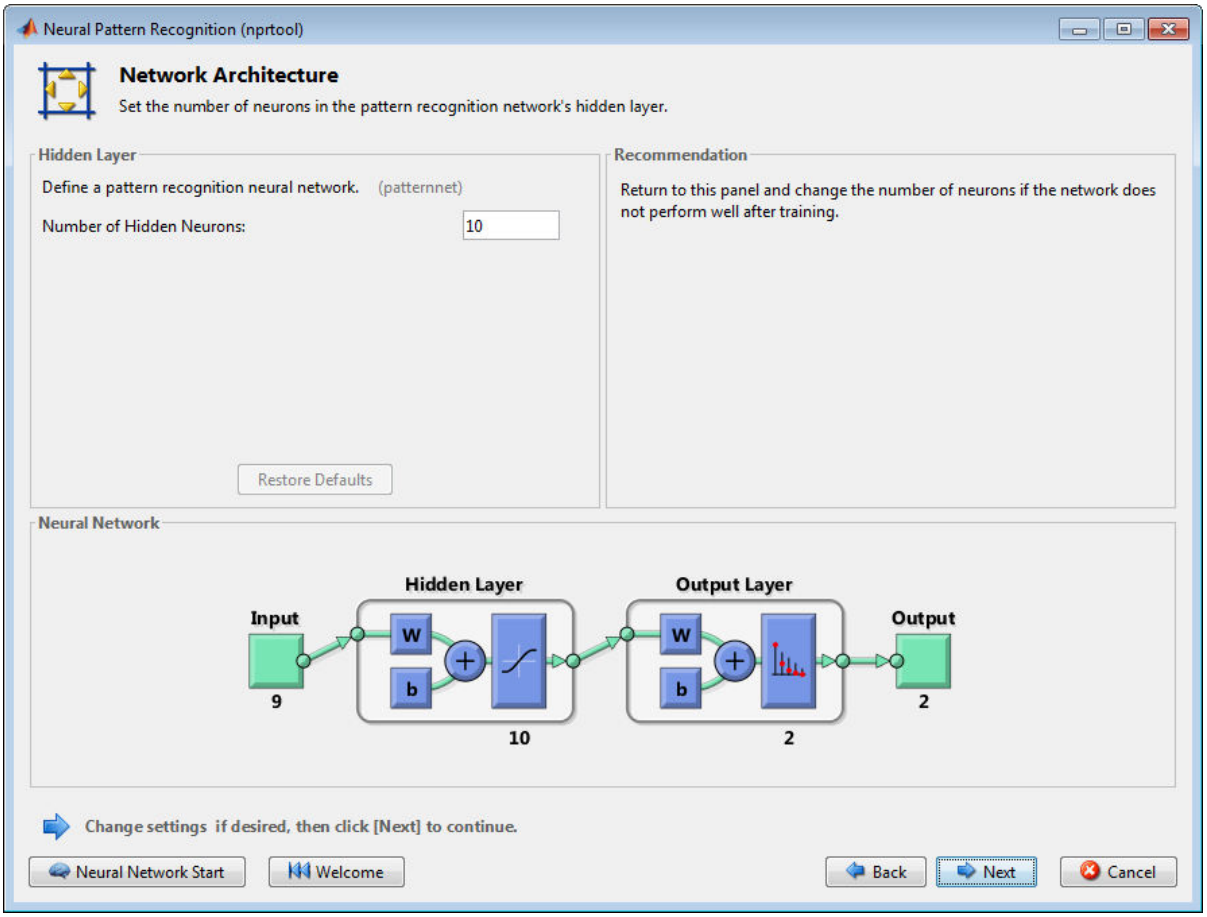

Figure 7.5 Snapshot of matlab window to specify the no.of neurons

6) Click Next.

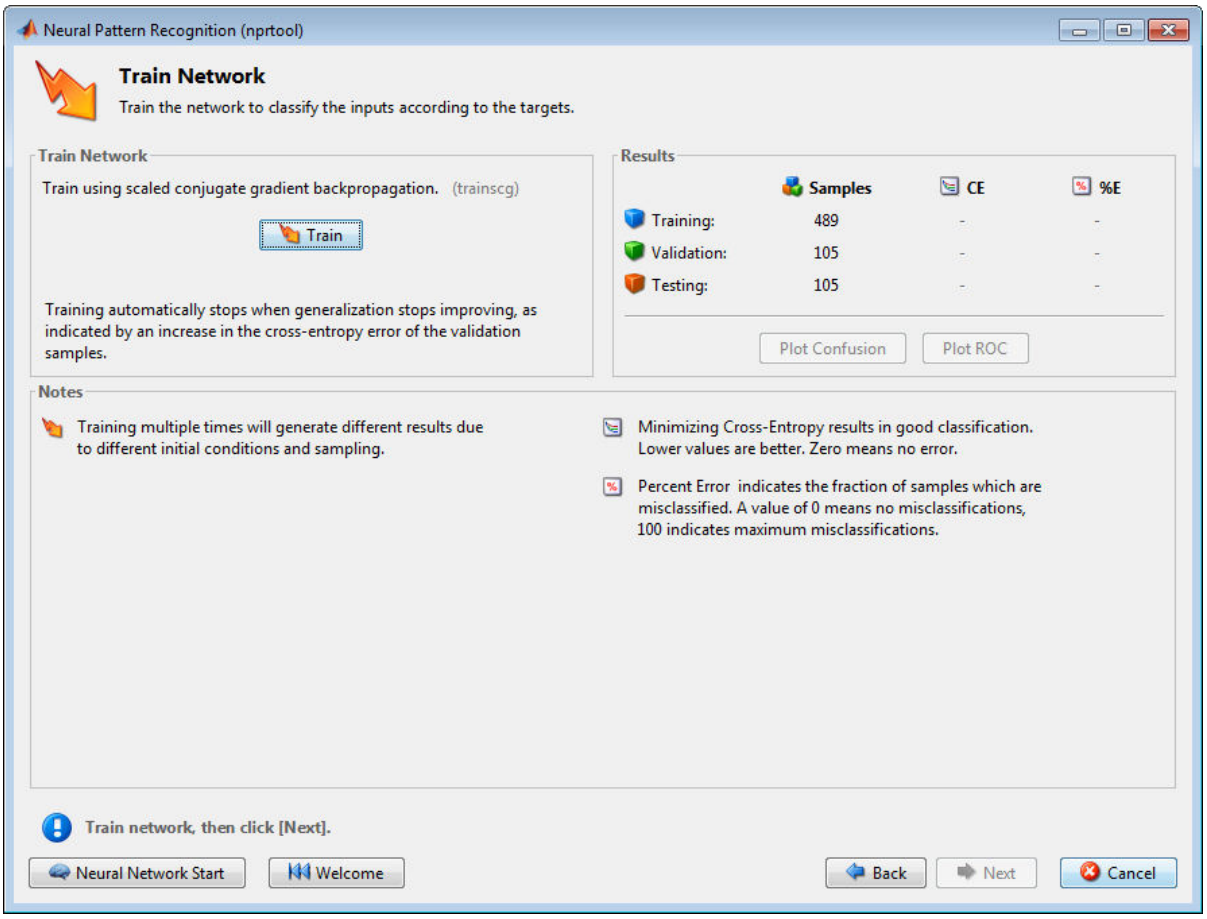

Figure 7.6 Snapshot of matlab window to train the network

The training of patterns and the subsequent response of the network can be done in which the network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units.

In this training automatically stops when generalization stops improving as indicated by an increase in the cross entropy error of the validation samples.

7) Click Train**.**

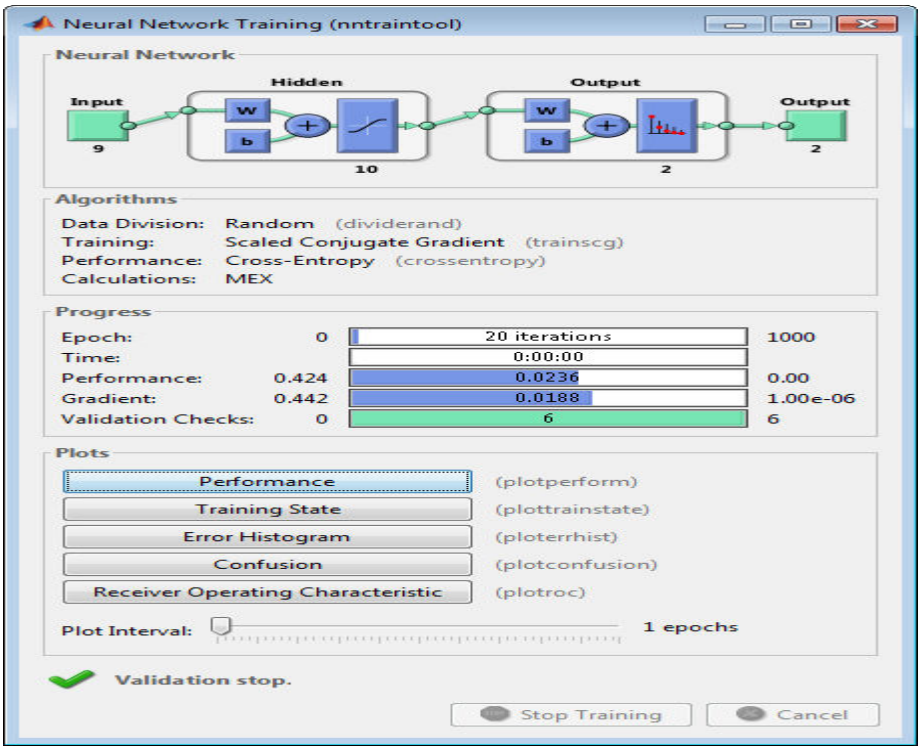

Figure 7.7 Snapshot of matlab window for progressing the model

An epoch is one complete presentation of the *data set to be learned* to a learning machine. Learning machines like feed-forward neural nets that use iterative algorithms often need many epochs during their learning phase.

Stochastic gradient descent (often shortened to SGD), also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization and iterative method for minimizing an objective function that is written as a sum of differentiable functions. In other words, SGD tries to find minima or maxima by iteration.

8) Under the Plots pane, click Confusion in the Neural Network Pattern Recognition App.

The next figure shows the confusion matrices for training, testing, and validation, and the three kinds of data combined. The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies.

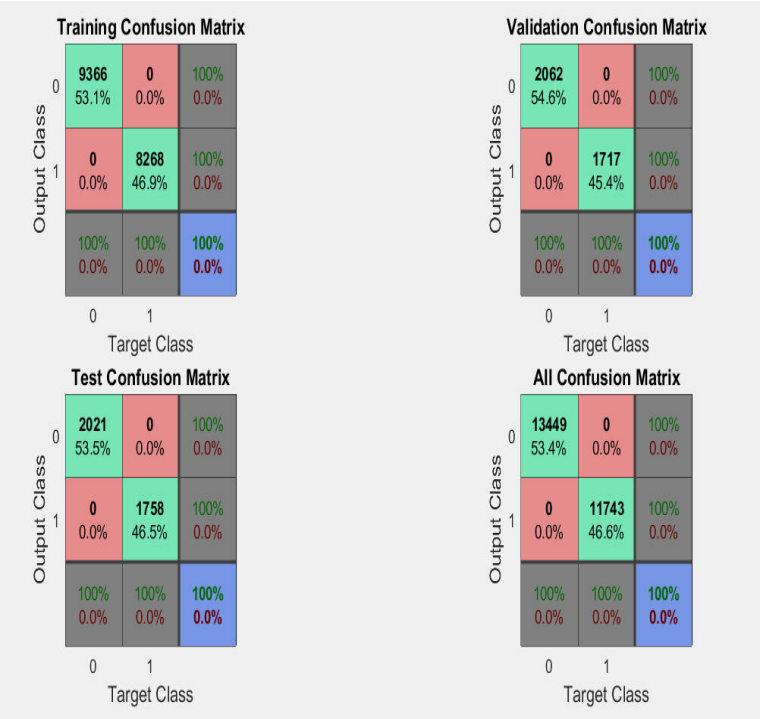

Figure 7.8 Snapshot of matlab window of confusion matrix

In the overall complete dataset of 25192 instances using ANN its correctly classifies normal of class 0 with 13449 samples and attack of class 1 with 11743 samples overall it gives accuracy of classification as 100% with no false alarm rate.

9) Plot the Receiver Operating Characteristic (ROC) curve. Under the Plots pane, click Receiver Operating Characteristic in the Neural Network Pattern Recognition App.

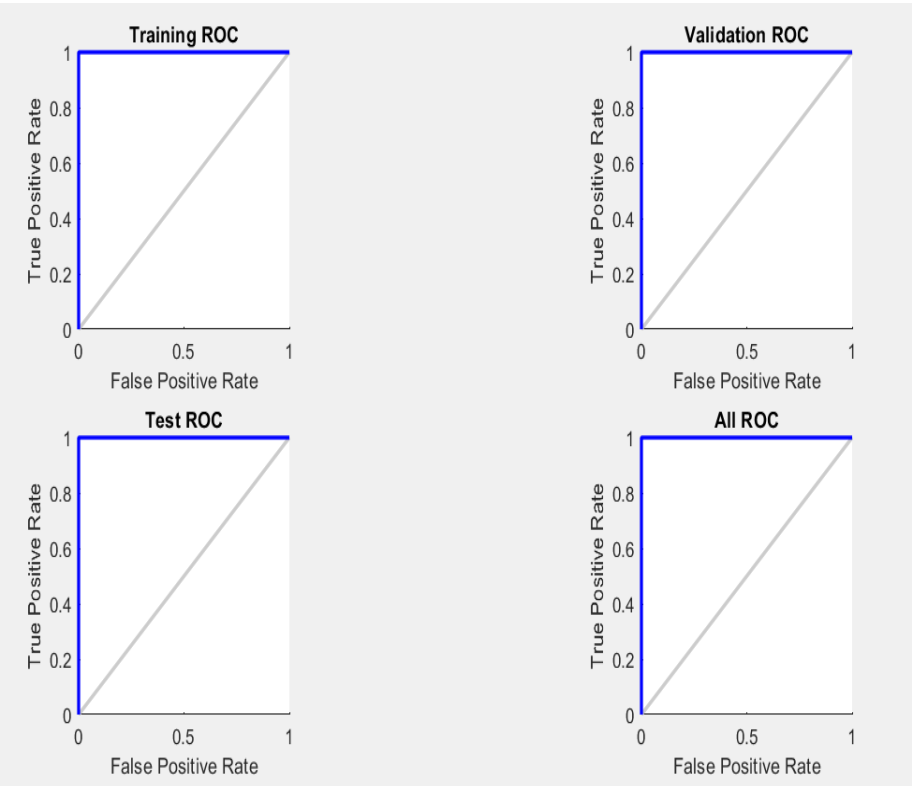

Figure 7.9 Snapshot of matlab window of ROC curve

In statistics, a receiver operating characteristic curve, i.e. ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning. The false-positive rate is also known as the fall-outer probability of false alarm and can be calculated as. The ROC curve is thus the sensitivity as a function of fall-out.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

The computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture.

The classification results were slightly better in the three layer network. From practical point of view. The experimental result is suggested that there is more to do in the IDS based on ANN. The cause of an intrusion detection system is to invention a potential intruder as possible as. An approach for a neural network based intrusion detection system motivated to classify the normal and attack patterns and the pattern of the attack. “Self Learning Network Traffic Classification” simplifies the manual bootstrapping of network traffic labels, once provided to the system, it leads to excellent performance.

6.2 FUTURE ENCHANCEMENT

In the future focusing on Deep Neural Networks and enhancing the problem statement much more broader so that problem can be solved effectively in Deep Neural Networks once in the traffic it identified that traffic leads to attack then classifying that traffic leads to what type of attack and how we can overcome with automatic network troubleshooting .

In future, planning to implement a real-time NIDS for real networks using deep learning technique. Additionally,making feature learning on raw network traffic headers instead of derived features using raw headers can be another high impact research in this area.

REFERENCES

1) Aggarwal, Preeti, and Sudhir Kumar Sharma. "Analysis of KDD dataset attributes-class wise for intrusion detection." Procedia Computer Science 57 (2015): 842-851.

2) Prof.Dighe Mohit S., Kharde Gayatri B., Mahadik Vrushali G., Gade Archana L., Bondre Namrata R , ”Using Artificial Neural Network Classification and Invention of Intrusion in Network Intrusion Detection System” International Journal of Innovative Research in Computer and Communication Engineering Vol. 3, Issue 2, February 2015.

3) Palechor, Fabio Mendoza, et al. "Feature selection, learning metrics and dimension reduction in training and classification processes in intrusion detection systems." Journal of Theoretical and Applied Information Technology 82.2 (2015): 291.

4) Vandana, M., and Sruthy Manmadhan. "Self learning network traffic classification." Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on. IEEE, 2015.

5) Bakhshi, Taimur, and Bogdan Ghita. "On Internet Traffic Classification: A Two-Phased Machine Learning Approach." Journal of Computer Networks and Communications 2016 (2016).

6) Kumar, Sanjay, Ari Viinikainen, and Timo Hamalainen. "Machine learning classification model for Network based Intrusion Detection System." Internet Technology and Secured Transactions (ICITST), 2016 11th International Conference for. IEEE, 2016.

7) Zhang, Zhuo, et al. "Proword: an unsupervised approach to protocol feature word extraction." INFOCOM, 2014 Proceeding IEEE, 2014.

8) Jaiswal, Rupesh Chandrakant, and Shashikant D. Lokhande. "Machine learning based internet traffic recognition with statistical approach." India Conference (INDICON), 2013 Annual IEEE, 2013.