INTRUSION DETECTION SYSTEM FOR CLOUD ENVIRONMENTS BASED SUPERVISED MACHINE LEARNING ALGORITHMS

A PROJECT REPORT

Submitted by

JEBINESH E H [Register No:211417104094]

HRITISH.S [Register No:211417104086]

GOKULAKRISHNAN. P [Register No:211417104070]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

MARCH 2021

BONAFIDE CERTIFICATE

Certified that this project report **“INTRUSION DETECTION SYSTEM FOR CLOUD ENVIRONMENTS BASED SUPERVISED MACHINE LEARNING ALGORITHMS”** is the bonafide work of **“JEBINESH. E.H. (211417104094), HRITISH.S (211417104086), GOKULAKRISHNAN. P (211417104070)”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.S.MURUGAVALLI,M.E.,Ph.D.,	**Mr. M.KRISHNAMOORTHY,BE.,ME.,MBA.,(Ph.D),**
HEAD OF THE DEPARTMENT	**ASSISTANT PROFESSOR**
DEPARTMENT OF CSE,	DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,	PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,	NAZARATHPETTAI,
POONAMALLEE,	POONAMALLEE,
CHENNAI-600 123.	CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**
Dr.C.SAKTHIKUMAR,M.E.,Ph.D., and
Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A., for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal,
Dr.K.MANI, M.E.,Ph.D., for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department,
Dr. S.MURUGAVALLI , M.E.,Ph.D., for the support extended throughout the project.

We would like to thank my Project Guide
Mr. M.KRISHNAMOORTHY, BE.,ME.,MBA.,(Ph.D), and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

JEBINESH E H

HRITISH.S

GOKULAKRISHNAN. P

ABSTRACT

There are many risks of network attacks under the Internet environment. Internet Security is a vital issue and therefore, the intrusion detection is one major research problem for business and personal networks to resist external attacks. A Network Intrusion Detection System (NIDS) is a software application that monitors the network or system activities for malicious activities and unauthorized access to devices. The goal of designing a NIDS is to protect data's confidentiality and integrity. Our project focuses on these issues with the help of Machine Learning. This project includes the implementation of different machine learning algorithms including Linear regression, K-Means Clustering and Artificial Neural Networks to automatically generate the rules for classify network activities. A comparative analysis of these techniques to detect intrusions has also been made. To learn the patterns of the attacks, NSL-KDD dataset has been used.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

NIDS	A Network Intrusion Detection System
IDS	Intrusion Detection System
HIDS	Host Based Detection System
ML	Machine Learning
SVM	Support Vector Machine
DF	Data Frame
NY	Numpy
RF	Random Forest

CHAPTER 1

INTRODUCTION

The main aim of this project is intruder detection system using KDD Dataset based on the Machine Learning Algorithms. An Intrusion Detection System (IDS) is a software application that monitors network or system activities for malicious activities and unauthorized access to devices.IDS come in a variety of “flavors” and approach the goal of detecting suspicious traffic in different ways. There are network based (NIDS) and host based (HIDS) intrusion detection systems. There are IDS that detect based on comparing traffic patterns against a baseline and looking for anomalies. There are IDS that simply monitor and alert and there are IDS that perform an action or actions in response to a detected threat. We’ll cover each of these briefly.

Network Intrusion Detection Systems are placed at a strategic point or points within the network to monitor traffic to and from all devices on the network. Ideally you would scan all inbound and outbound traffic, however doing so might create a bottleneck that would impair the overall speed of the network.

When designing a IDS, the mission is to protect the data’s

- Confidentiality – read
- Integrity – read/write

Two major levels/types

- Misuse-based IDS
- Anomaly-based IDS

Eavesdropping

In general, the majority of network communications occur in an unsecured or "cleartext" format, which allows an attacker who has gained access to data paths in your network to "listen in" or interpret (read) the traffic. When an attacker is eavesdropping on your communications, it is referred to as sniffing or snooping. The ability of an eavesdropper to monitor the network is generally the biggest security problem that administrators face in an enterprise. Without strong encryption services that are based on cryptography, your data can be read by others as it traverses the network.

Data Modification

After an attacker has read your data, the next logical step is to alter it. An attacker can modify the data in the packet without the knowledge of the sender or receiver. Even if you do not require confidentiality for all communications, you do not want any of your messages to be modified in transit. For example, if you are exchanging purchase requisitions, you do not want the items, amounts, or billing information to be modified.

Identity Spoofing (IP Address Spoofing)

Most networks and operating systems use the IP address of a computer to identify a valid entity. In certain cases, it is possible for an IP address to be falsely assumed— identity spoofing. An attacker might also use special programs to construct IP packets that appear to originate from valid addresses inside the corporate intranet.

Compromised-Key Attack

A key is a secret code or number necessary to interpret secured information. Although obtaining a key is a difficult and resource-intensive process for an attacker, it is possible. After an attacker obtains a key, that key is referred to as a compromised key.

An attacker uses the compromised key to gain access to a secured communication without the sender or receiver being aware of the attack. With the compromised key, the attacker can decrypt or modify data, and try to use the compromised key to compute additional keys, which might allow the attacker access to other secured communications.

Sniffer Attack

A sniffer is an application or device that can read, monitor, and capture network data exchanges and read network packets. If the packets are not encrypted, a sniffer provides a full view of the data inside the packet. Even encapsulated (tunneled) packets can be broken open and read unless they are encrypted and the attacker does not have access to the key.

Using a sniffer, an attacker can do any of the following:

- Analyze your network and gain information to eventually cause your network to crash or to become corrupted.
- Read your communications.

Application-Layer Attack

An application-layer attack targets application servers by deliberately causing a fault in a server's operating system or applications. This results in the attacker gaining the ability to bypass normal access controls. The attacker takes advantage of this situation, gaining control of your application, system, or network, and can do any of the following:

- Read, add, delete, or modify your data or operating system.
- Introduce a virus program that uses your computers and software applications to copy viruses throughout your network.
- Introduce a sniffer program to analyze your network and gain information that can eventually be used to crash or to corrupt your systems and network.
- Abnormally terminate your data applications or operating systems.
- Disable other security controls to enable future attacks.

Password-Based Attacks

A common denominator of most operating system and network security plans is password-based access control. This means your access rights to a computer and network resources are determined by who you are, that is, your user name and your password.

Older applications do not always protect identity information as it is passed through the network for validation. This might allow an eavesdropper to gain access to the network by posing as a valid user.

When an attacker finds a valid user account, the attacker has the same rights as the real user. Therefore, if the user has administrator-level rights, the attacker also can create accounts for subsequent access at a later time.

After gaining access to your network with a valid account, an attacker can do any of the following:

- Obtain lists of valid user and computer names and network information.
- Modify server and network configurations, including access controls and routing tables.
- Modify, reroute, or delete your data.

Man-in-the-Middle Attack

As the name indicates, a man-in-the-middle attack occurs when someone between you and the person with whom you are communicating is actively monitoring, capturing, and controlling your communication transparently. For example, the attacker can re-route a data exchange. When computers are communicating at low levels of the network layer, the computers might not be able to determine with whom they are exchanging data.

Man-in-the-middle attacks are like someone assuming your identity in order to read your message. The person on the other end might believe it is you because the attacker might be actively replying as you to keep the exchange going and gain more information. This attack is capable of the same damage as an application-layer attack, described later in this section.

TYPES OF CURRENT IDS

- Misuse Detection
- Record the specific patterns of intrusions.
- Monitor current event sequences and pattern matching
- Report the matched events as intrusions

The IDS analyzes the information it gathers and compares it to large databases of attack signatures. Essentially, the IDS look for a specific attack that has already been documented.

Like a virus detection system, misuse detection software is only as good as the database of attack signatures that it uses to compare packets against.
Examples: Expert Systems, Signature Analysis, State Transition Analysis

- Anomaly Detection
- Establish the normal behavior profiles.
- Compare current activities with normal profiles.
- Report significant deviations as intrusions.

The system administrator defines the baseline, or normal, state of the networks traffic load, breakdown, protocol, and typical packet size.

The anomaly detector monitors network segments to compare their state to the normal baseline and look for anomalies.

Examples: Statistical Analysis, Expert System

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

A false positive is a situation where something abnormal (as defined by the IDS) happens, but it is not an intrusion. User will quit monitoring IDS because of noise. No Confidentiality. Not an real time network data implemented.

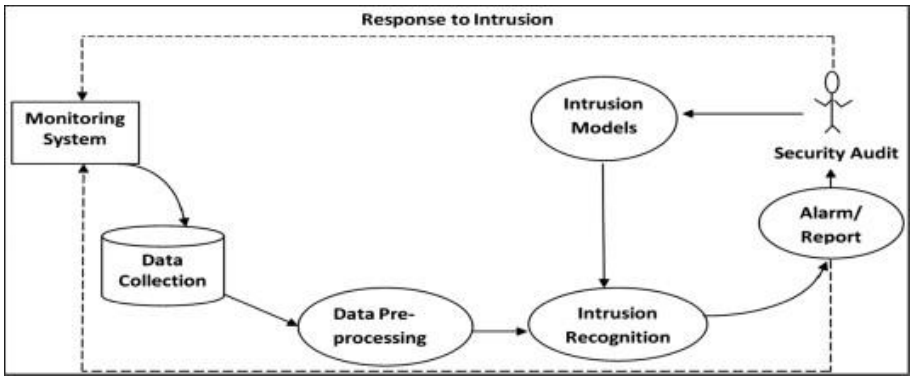
No Data Confidentiality. No Data integrity. Less prediction accuracy. Not a real time analysis.

Existing system not used for an long dataset. The prediction of identity malicious activity is not much accurate.

2.2 PROPOSED SYSTEM

This research focuses on solving the issues in intrusion detection communities that can help the administrator to make pre-processing, classification, labeling of data and to mitigate the outcome of Distributed Denial of Service Attacks. Since, the network administrator feels difficult to pre-process the data. Due to the overwhelming growth of attacks which makes the task hard, attacks can be identified only after it happens. To overcome this situation, frequent updating of profiles is needed. Reduced workload of administrator increases the detection of attacks. Data mining includes many different algorithms to accomplish the desired tasks. All of these algorithms aims to fit a model to the prescribed data and even analyzes the data and simulate a model which is closest to the data being analyzed.

Some of the open issues have been taken to detect attacks over the network. To achieve this, the framework of the proposed methods has been discussed below. The entire framework of the proposed methodology in Intrusion Detection System is described in Fig.



Advantages

- When designing a IDS, the mission is to protect the data's

  Confidentiality – read

  Integrity – read/write

- Real time data analytics.

- High accuracy.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 SOFTWARES USED

3.1.1 Anaconda

Anaconda is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. Anaconda Distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS.

Python is a high-level programming language devised by Guido van Rossum & first released in 1991. It's the most popular coding language used by software developers to build, control, manage and for testing. It is also an interpreter which executes Python programs. The python interpreter is called python.exe on Windows.

Python Packages

Packages or additional libraries help in scientific computing and computational modelling. In Python, the packages are not the part of the Python standard library. Few major packages are – numpy (Numeric Python): matrices and linear algebra scipy (Scientific Python): many numerical routines matplotlib: (Plotting Library) creating plots of data sympy (Symbolic Python): symbolic computation pytest (Python Testing): a code testing framework.

Together with a list of Python packages, tools like editors, Python distributions include the Python interpreter. Anaconda is one of several Python distributions. Anaconda is a new distribution of the Python and R data science package. It was formerly known as Continuum Analytics. Anaconda has more than 100 new packages.

This work environment, Anaconda is used for scientific computing, data science, statistical analysis, and machine learning. The latest version of

Anaconda 5.0.1 is released in October 2017.The released version 5.0.1 addresses some minor bugs and adds useful features, such as updated R language support. All of these features weren’t available in the original 5.0.0 release.

This package manager is also an environment manager, a Python distribution, and a collection of open source packages and contains more than 1000 R and Python Data Science Packages.

3.1.2 Ipython Notebooks

IPython is a command shell for interactive computing in multiple programming languages, originally developed for the Python programming language, that offers introspection, rich media, shell syntax, tab completion, and history. IPython provides the following features:

Interactive shells (terminal and Qt-based).

- A browser-based notebook interface with support for code, text, mathematical expressions, inline plots and other media.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into one's own projects.
- Tools for parallel computing.

IPython is based on an architecture that provides parallel and distributed computing. IPython enables parallel applications to be developed, executed, debugged and monitored interactively. Hence, the I (Interactive) in IPython.[3]This architecture abstracts out parallelism, which enables IPython to support many different styles of parallelism[4]including:

With the release of IPython 4.0, the parallel computing capabilities have been made optional and released under the ipyparallel python package.IPython frequently draw from SciPy stack[5] libraries like NumPy and SciPy, often installed alongside from one of many Scientific Python distributions. IPython provide integration some library of the SciPy stack like matplotlib, like inline graph when in used with the Jupyter notebook. Python libraries can implement IPython specific hooks to customize object Rich object display. SymPy for example implement rendering of Mathematical Expression as rendered LaTeX when used within IPython context.

Other features:

IPython also allows non-blocking interaction with Tkinter, PyGTK, PyQt/PySide and wxPython (the standard Python shell only allows interaction with Tkinter). IPython can interactively manage parallel computing clusters using asynchronous status call-backs and/or MPI. IPython can also be used as a system shell replacement. Its default behaviour is largely similar to Unix shells, but it allows customization and the flexibility of executing code in a live Python environment. Using IPython as a shell replacement is less common and it is now recommended to use Xonsh which provide most of the IPython feature with better shell integrations.

3.1.3 Jupyter Notebook

It is a web based interface that allows for rapid prototyping and sharing of data-related projects. It works with many kernels (this is the name given to the code env that it can run), including but not limited to Python and R (even though its more famous and suited for Python).Previously it used to be called IPython Notebook but has been renamed and moved to the Jupyter project. Jupyter is an open source project aiming at creating a better work experience for (data) scientists.

3.2 INTRODUCTION

AI (artificial intelligence) is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning (the acquisition of information and rules for using the information), reasoning (using the rules to reach approximate or definite conclusions) and self-correction. Particular applications of AI include expert systems, speech recognition and machine vision.AI was coined by John McCarthy, an American computer scientist, in 1956 at The Dartmouth Conference where the discipline was born. Today, it is an umbrella term that encompasses everything from robotic process automation to actual robotics. It has gained prominence recently due, in part, to big data, or the increase in speed, size and variety of data businesses are now collecting. AI can perform tasks such as identifying patterns in the data more efficiently than humans, enabling businesses to gain more insight out of their data.It is a branch of computer science that aims to create intelligent machines. It has become an essential part of the technology industry.Research associated with artificial intelligence is highly technical and specialized. The core problems of artificial intelligence include programming computers.

Knowledge engineering is a core part of AI research. Machines can often act and react like humans only if they have abundant information relating to the world. Artificial intelligence must have access to objects, categories, properties and relations between all of them to implement knowledge engineering. Initiating common sense, reasoning and problem-solving power in machines is a difficult and tedious task.

Machine learning is also a core part of AI. Learning without any kind of supervision requires an ability to identify patterns in streams of inputs, whereas learning with adequate supervision involves classification and numerical regressions. Classification determines the category an object belongs to and regression deals with obtaining a set of numerical input or output examples, thereby discovering functions enabling the generation of suitable outputs from respective inputs. Mathematical analysis of machine learning algorithms and their performance is a well-defined branch of theoretical computer science often referred to as computational learning theory.Machine perception deals with the capability to use sensory inputs to deduce the different aspects of the world, while computer vision is the power to analyze visual inputs with a few sub-problems such as facial, object and gesture recognition. Robotics is also a major field related to AI. Robots require intelligence to handle tasks such as object manipulation and navigation, along with sub-problems of localization, motion planning and mapping.

3.3 HARDWARE AND SOFTWARE SPECIFICATION

3.3.1 Hardware Requirements

➢ Hard Disk:	80GB and Above
➢ RAM	: 4GB and Above
➢ Processor :	P IV and Above

3.3.2 Software Requirements

- Python 3.7.2
- Jupyter Notebook(IDE)

3.3.3 Machine Learning

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers. Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

Here in this thesis, we are providing basic info of the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbour algorithm, decision tree learning, and deep learning.

Supervised learning

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labeled by humans, and unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within its input data.Let's explore these methods in more detail.

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output Y = f(X) . The goal is to approximate the mapping function so well that when you have new input data .(x) that you can predict the output variables (Y) for that data. Techniques of Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, Decision Trees and support vector machines. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. For example, a classification algorithm will learn to identify animals after being trained on a dataset of images that are properly labeled with the species of the animal and some identifying characteristics. Supervised learning problems can be further grouped into Regression and Classificationproblems. Both problems have as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification.

Classification

As the name suggests, Classification is the task of "classifying things" into sub- categories.But, by a machine. If that doesn't sound like much, imagine your computer being able to differentiate between you and a stranger. Between a potato and a tomato. Between an A grade and a F.In Machine Learning and Statistics, Classification is the problem of identifying to which of a set of categories (sub populations), a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.

Types of Classification

Classification is of two types:

- Binary Classification : When we have to categorize given data into 2 distinct classes. Example – On the basis of given health conditions of a person, we have to determine whether the person has a certain disease or not.

- Multiclass Classification : The number of classes is more than 2. For Example – On the basis of data about different species of flowers, we have to determine which specie does our observation belong to Fig 2 : Binary and Multiclass Classification. Here x1 and x2 are our variables upon which the class is predicted.Suppose we have to predict whether a given patient has a certain disease or not, on the basis of 3 variables, called features. Which means there are two possible outcomes:

1. The patient has the said disease. Basically a result labelled “Yes” or “True”.

2. The patient is disease free. A result labelled “No” or “False”.

This is a binary classification problem.We have a set of observations called training data set, which comprises of sample data with actual classification results. We train a model, called Classifier on this data set, and use that model to predict whether a certain patient .

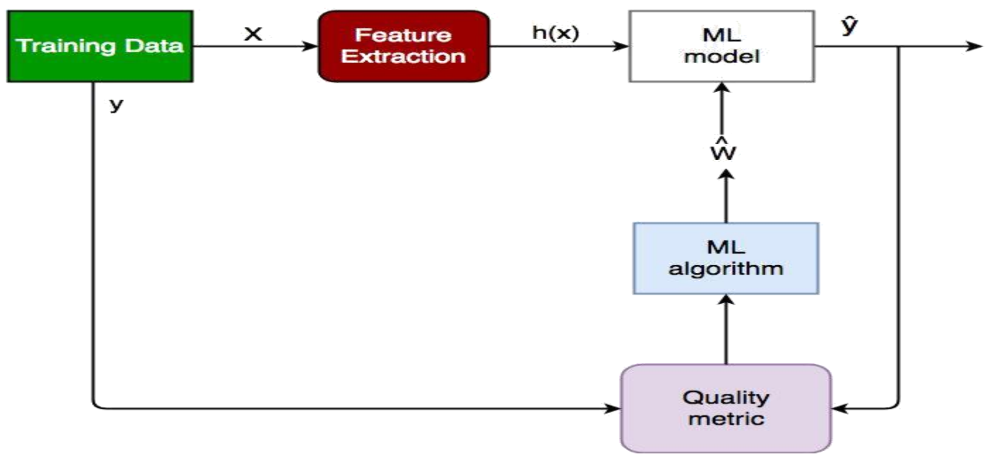


Fig 3.1 Generalized Classification Block Diagram.

1. X : pre-classified data, in the form of a N*M matrix. N is the no. of observations and M is the number of features

2. y : An N-d vector corresponding to predicted classes for each of the N observations.

3. Feature Extraction : Extracting valuable information from input X using a series of transforms.

4. ML Model : The “Classifier” we’ll train.

5. y’ : Labels predicted by the Classifier.

6. Quality Metric : Metric used for measuring the performance of the model.

7. ML Algorithm : The algorithm that is used to update weights w’, which update the model and “learns” iteratively.

Types of Classifiers (Algorithms)

There are various types of classifiers. Some of them are :

- Linear Classifiers : Logistic Regression
- Tree Based Classifiers : Decision Tree Classifier
- Support Vector Machines
- Artificial Neural Networks
- Bayesian Regression
- Gaussian Naive Bayes Classifiers
- Stochastic Gradient Descent (SGD) Classifier
- Ensemble Methods : Random Forests, AdaBoost, Bagging Classifier, Voting Classifier, ExtraTrees Classifier

Practical Applications of Classification

- Google's self driving car uses deep learning enabled classification techniques which enables it to detect and classify obstacles.
- Spam E-mail filtering is one of the most widespread and well recognized uses of Classification techniques.
- Detecting Health Problems, Facial Recognition, Speech Recognition, Object Detection, Sentiment Analysis all use Classification at their core.

Regression

A regression problem is when the output variable is a real or continuous value, such as “salary” or “weight”. Many different models can be used, the simplest is the linear regression. It tries to fit data with the best hyper-plane which goes through the points.

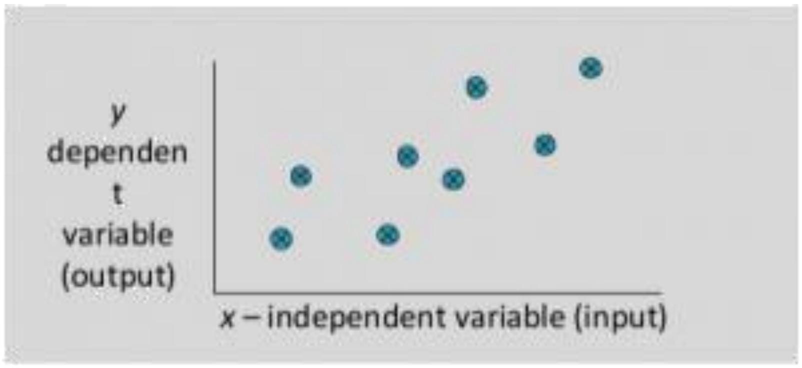


Fig 3.2 Linear Regression Model

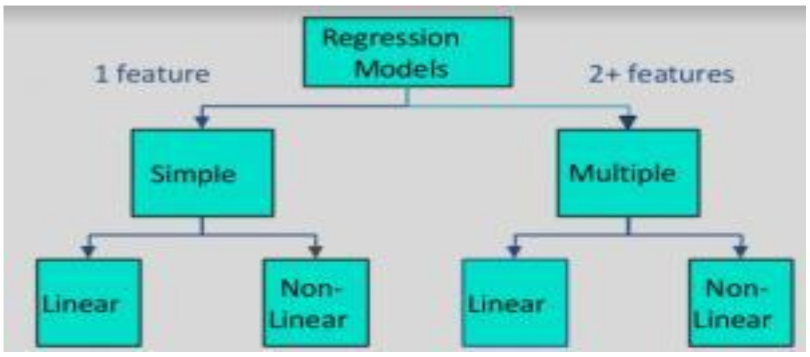


Fig 3.3 Types Of Regression Models

Unsupervised Learning

Unsupervised learning is where we only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data. Unsupervised learning problems can be further grouped into clustering and association problems.

Clustering

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples. Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.For example, the data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.

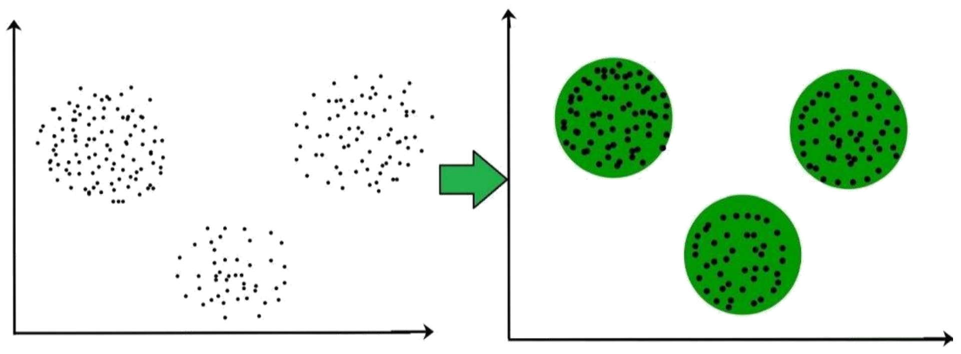


Fig 3.4 Clustering overview

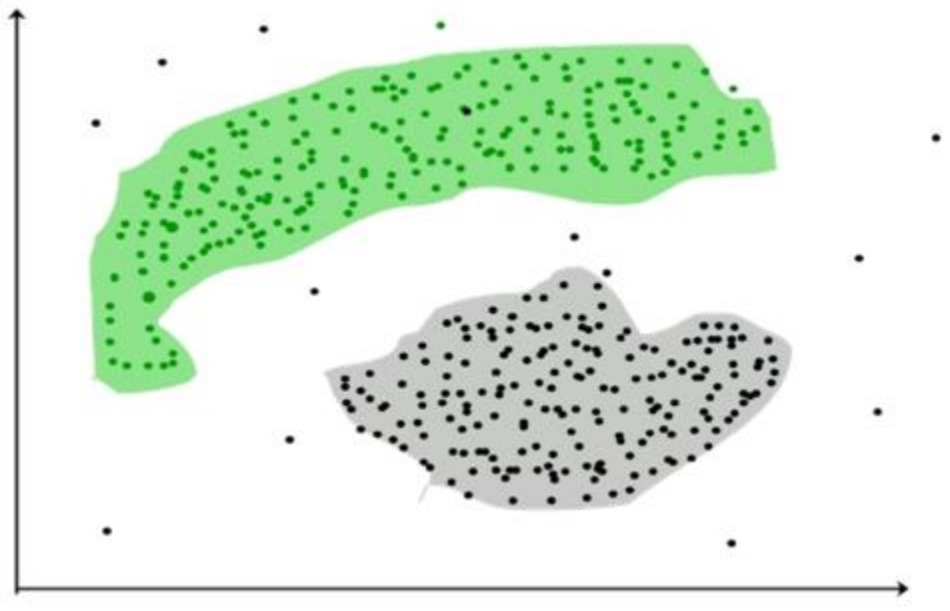


Fig 3.5 Clustering Example

These data points are clustered by using the basic concept that the data point lies within the given constraint from the cluster center. Various distance methods and techniques are used for calculation of the outliers.

Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters.

Clustering Methods :

1. Density-Based Methods : These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space. These methods have good accuracy and ability to merge two clusters.Example DBSCAN (Density-Based Spatial Clustering of Applications with Noise) , OPTICS (Ordering Points to Identify Clustering Structure) etc.

2. Hierarchical Based Methods : The clusters formed in this method forms a tree type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two category

   - Agglomerative (bottom up approach)

   - Divisive (top down approach) .

3. Partitioning Methods : These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter example K-means, CLARANS (Clustering Large Applications based upon randomized Search) etc.

4. Grid-based Methods : In this method the data space are formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects example STING (Statistical Information Grid), wave cluster, CLIQUE (CLustering In Quest) etc.

Clustering Algorithms:

- K-Means Clustering.

- Mean-Shift Clustering for a single sliding window.

- The entire process of Mean-Shift Clustering.

- DBSCAN Smiley Face Clustering.

- EM Clustering using GMMs.

- Agglomerative Hierarchical Clustering.

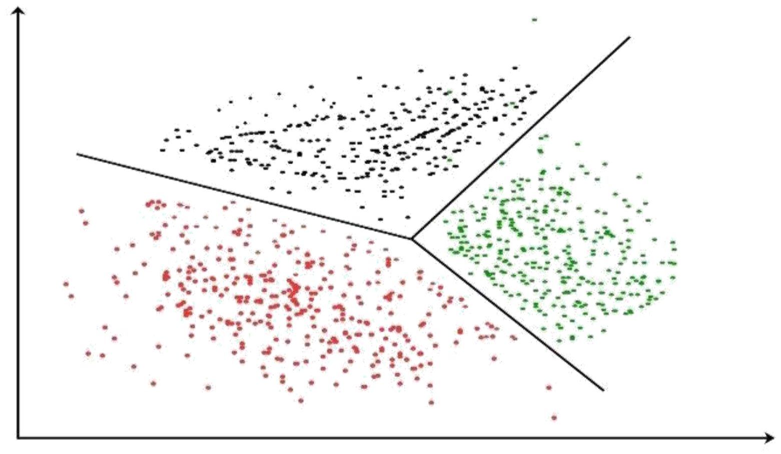


Fig 3.6 k-means clustering

CHAPTER 4

SYSTEM DESIGN

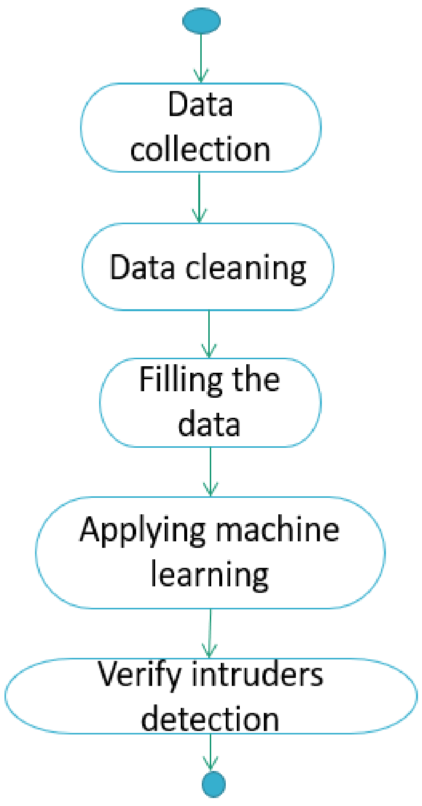
4.1 UML DIAGRAMS

The elements are like components which can be associated in different ways to make a complete UML picture, which is known as diagram. Thus, it is very important to understand the different diagrams to implement the knowledge in real-life systems.

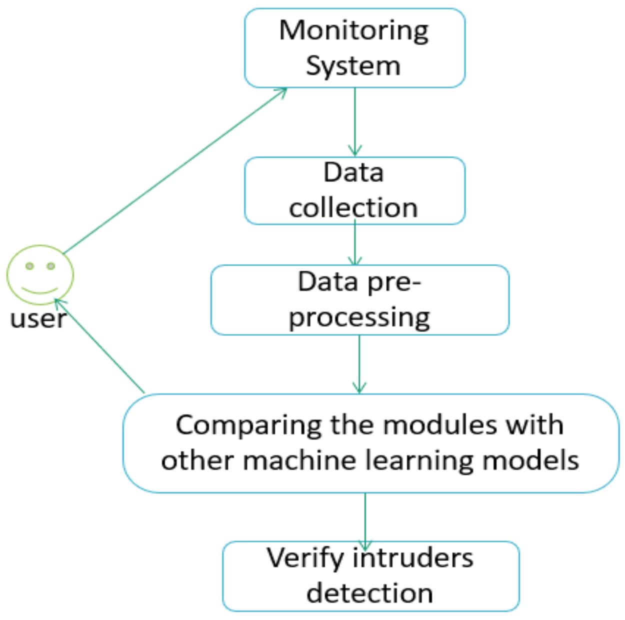
Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. If we look around, we will realize that the diagrams are not a new concept but it is used widely in different forms in different industries.

We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system.

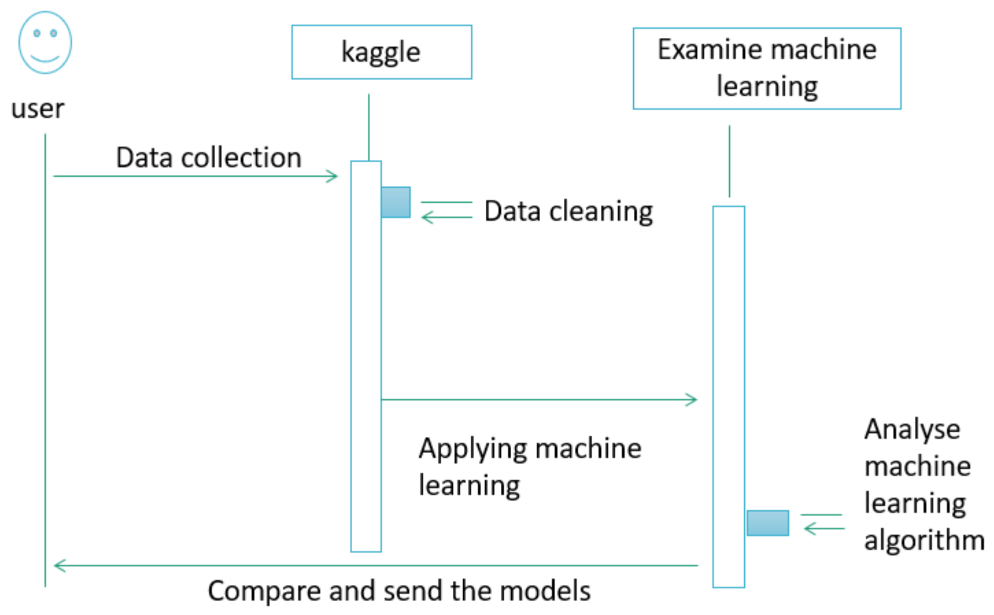
4.1.1 Activity Diagram



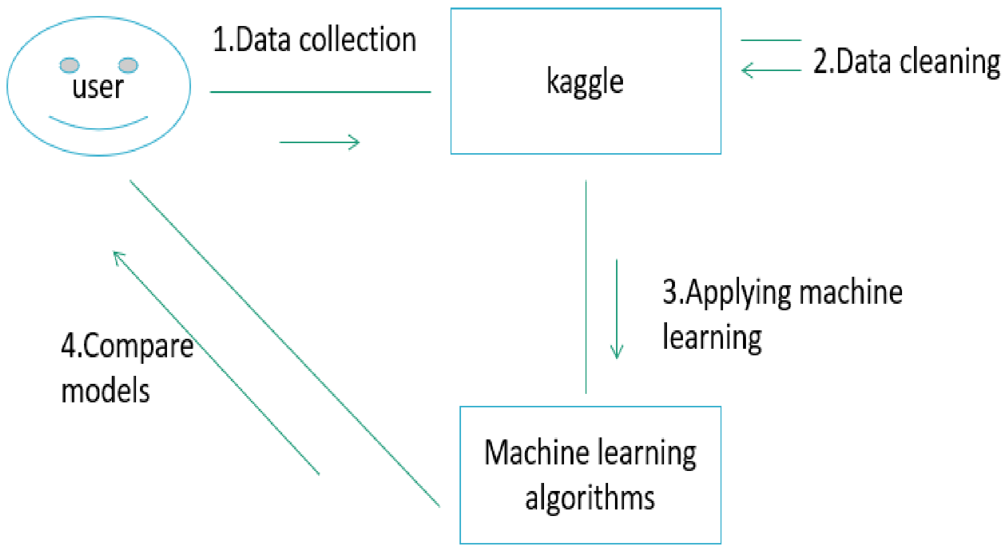
4.1.2 Usecase Diagram


Monitoring System
Data collection
Data pre-processing
user
Comparing the modules with other machine learning models
Verify intruders detection

4.1.3 Sequence Diagram


user
kaggle
Examine machine learning
Data collection
Data cleaning
Applying machine learning
Analyse machine learning algorithm
Compare and send the models

4.1.4 Collaboration Diagram



4.2 DATA

1. Data for Network intruder detection.
2. Data preprocessing.
3. Mean Normalization
4. Results and Discussions.

1. Data collection.

Attacks can be described as •Dos attack – It is a kind of attack where the attacker makes processing time of the resources and memory busy so as to avoid legitimate user from accessing those resources.

- U2R attack – Here the attacker sniffs the password or makes some kind of attack to access the particular host in a network as a legitimate user. They can even promote some vulnerability to gain the root access of the system.
- R2L attack – Here the attacker sends a message to the host in a network over remote system and makes some vulnerability.
- Probe attack – Attacker will scan the network to gather information and would make some violation in the future.

KDD Cup 99 data set contains 23 attack types and their names are shown in Table and its features are grouped as,

1. Basic features

It encompasses all the attributes of TCP/IP connection and leads to delay in detection.

2. Traffic features

It is evaluated in accordance with window interval & two features as same host and same service.

(a) Same host feature

It examines the number of connections for the past 2 s that too from the same destination host. In other words, the probability of connections will be done in a specific time interval.

(b) Same service feature

It examines the number of connections in a particular time interval that too posses same service.

3. Content features

Dos & probe attack have frequent intrusion sequential patterns than the R2L & U2R. Because these two attacks include many connections to several hosts at a particular time period whereas R2L and U2R perform only a single connection. To detect these types of attacks, domain knowledge is important to access the data portion of the TCP packets. Ex. Failed login, etc. these features are called as content features.

We are using NSL-KDD dataset divided into two parts, one for training and other for testing.

There are total 42 features in the dataset including Protocol_type, Service, Flag, Source bytes, Destination bytes, Num_failed_logins, Root shells etc.

The last column contains the output denoting a normal activity or an attack.

Different types of attacks are included like DoS, U2R, R2L, Probing.

2. Data preprocessing.

Collecting the data is one task and making that data useful is an-other vital task. Data collected from various means will be in an unorganized format and there may be lot of null values, in-valid data values and unwanted data. Cleaning all these data and replacing them with appropriate or approximate data and removing null and missing data and replacing them with some fixed alternate values are the basic steps in pre processing of data. Even data collected may contain completely garbage values. It may not be in exact format or way that is meant to be. All such cases must be verified and replaced with alternate values to make data meaning meaningful and useful for further processing. Data must be kept in a organized format.

3. Mean normalisation

The next step is algorithms are applied to data and results are noted and observed. The algorithms are applied in the fashion mention in the diagram so as to improve accuracy at each stage.

Finally after processing of data and training the very next task is obviously testing. This is where performance of the algorithm, quality of data, and required output all appears out. From the huge data set collected 80 percent of the data is utilized for train-ing and 20 percent of the data is reserved for testing. Training as discussed before is the process of making the machine to learn and giving it the capability to make further predictions based on the training it took. Whereas testing means already having a predefined data set with output also previously labeled and the model is tested whether it is working properly or not and is giving the right prediction or not. If maximum number of predictions is right then model will have a good accuracy percentage and is reliable to continue with otherwise better to change the model.

Normalization refers to the creation of shifted and scaled versions of statistics, where the intention is that these normalized values allow the comparison of corresponding normalized values for different datasets in a way that eliminates the effects of certain gross influences, as in an anomaly time series. Some types of normalization involve only a rescaling, to arrive at values relative to some size variable. In terms of levels of measurement, such ratios only make sense for ratio measurements (where ratios of measurements are meaningful), not interval measurements (where only distances are meaningful, but not ratios).

Normalization by Mean

Assume that there are n rows with seven variables (columns), A, B, C, D, E, F and G, in the data. We use variable E as an example in the calculations below. The remaining variables in the rows are normalized in the same way.

Without rescaling (Baseline variable = None)

The normalized value of ei for variable E in the ith row is calculated as:

Where,

p = the number of records used to calculate the mean

Rescaling by a baseline variable

If we select variable A as baseline variable, the normalized value of ei for variable E in the ith row is calculated as:

where

p = the number of rows used to calculate the mean

aj = the value for variable A in the jth record

4. **Results and Discussions**

KDD Cup 99 data set has been used in this research of which 60% is treated as training data and 40% is considered as testing data. The proposed framework has been implemented in MatLab10 and Java using data mining techniques. Performance of four proposed methods such as,

- Classification of network data using EDADT algorithm.
- Proposed Hybrid IDS.
- Performance of Semi-Supervised Approach for IDS and,
- Mitigating DDoS attacks using Varying Clock Drift Mechanism.

LINEAR REGRESSION

Study of linear relationship between an output variable and one or more input features.

For a linear model, our hypothesis of the form,

h_θ(x) = θ_0 + θ_1x or h_θ(x) = θ^Tx.

We have to adjust θ_0 and θ_1 so that $h_\theta(x)$ is close to y. For that we define an error function,

$$J(\theta_0,\theta_1) = (1 / 2m) * \sum_{i=1}^{m} \left(h_\theta (x^i) - y^i \right)^2$$

To minimize the error, we use Gradient Descent Algorithm,

Repeat until convergence {

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial\theta_0} J(\theta_0,\theta_1) \quad \text{for } j=0 \text{ and } j=1$$

CHAPTER 5

CODING AND TESTING

5.1 CODING

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

5.2 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand.

- Naming conventions
- Value conventions
- Script and comment procedure
- Message box format
- Exception and error handling

5.2.1 Naming Conventions

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy**

understanding of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

Class names

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

Member Function and Data Member name

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

5.2.2 Value Conventions

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.
- Proper documentation of flag values.

5.2.3 Script writing and commenting standard

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

5.2.4 Message Box Format

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

5.3 TEST PROCEDURE

System Testing

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

5.4 TEST DATA AND OUTPUT

5.4.1 Unit Testing

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

5.4.2 Functional Tests

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

5.4.3 Performance Test

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

5.4.4 Stress Test

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

5.4.5 Structured Test

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

5.4.6 Integration Testing

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

5.5 TESTING TECHNIQUES / TESTING STRATERGIES

5.5.1 Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet – undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

White box testing

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- ➢ Flow graph notation
- ➢ Cyclometric complexity
- ➢ Deriving test cases
- ➢ Graph matrices Control

Black box testing

In this testing by knowing the internal operation of a product, test can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

5.5.2 Software Testing Strategies

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

Integration testing:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

Program testing:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

Security testing:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

Validation Testing

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- The function or performance characteristics confirm to specifications and are accepted.

- A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

User acceptance testing

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.

- Output screen design.

CODING

TCP-checkpoint.ipynb

```
import numpy as np

import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt


df = pd.read_csv("kdd.csv",index_col=0)


df.head()


tcp_syn_df = df[df.loc[:,"protocol_type"] == "tcp"]


tcp_syn_df = tcp_syn_df[tcp_syn_df.loc[:,"srv_serror_rate"] > 0.7]


tcp_syn_df.to_csv("tcp.csv")


tcp_syn_df.head()


#there are no null values from priori knowledge
```

```
service_values = np.unique(tcp_syn_df.loc[:,"service"])

mid = (len(service_values)+1)/2

for i in range(len(service_values)):

    tcp_syn_df = tcp_syn_df.replace(service_values[i], (i-mid)/10)



tcp_syn_df.head()



#I will be extracting all the important features as a "priori" for
preprocessing

features =
["service","src_bytes","wrong_fragment","count","num_compromis
ed","srv_count","srv_serror_rate","serror_rate"]

target = "result"



X = tcp_syn_df.loc[:,features]

y =tcp_syn_df.loc[:,target]



classes = np.unique(y)

print(classes)



#replacing all classes of attack with 1 and normal result with 0 in our
icmp_df

for i in range(len(classes)):
    if i == 2:
        tcp_syn_df = tcp_syn_df.replace(classes[i], 0)
    else:
        tcp_syn_df = tcp_syn_df.replace(classes[i], 1)


#turning the service attribute to categorical values
tcp_syn_df=tcp_syn_df.replace("eco_i",-0.1)
tcp_syn_df=tcp_syn_df.replace("ecr_i",0.0)
tcp_syn_df=tcp_syn_df.replace("tim_i",0.1)
tcp_syn_df=tcp_syn_df.replace("urp_i",0.2)


tcp_syn_df.head()


y =tcp_syn_df.loc[:,target]


#I selected certain features but I will have to find some covariance
between them so I will plot a covariance heatmap
sns.heatmap(X.corr(), annot=True,cmap="RdBu")
plt.plot()
#the data as seen is highly uncorrelated as most of it is one valued
such as the duration one.
```

```
from sklearn.ensemble import RandomForestClassifier



rs = RandomForestClassifier()

rs.fit(X,y)

print(pd.Series(rs.feature_importances_,index=features).sort_values(
ascending=False))



#updated feature selection to release overfitting and accuracy
improvement

#in generalised prediction

X =
X.loc[:,["service","count","srv_count","src_bytes","serror_rate"]]

for i in X.index:

    if y[i] == 0:

        print(i,":",X.loc[i,:])




print(list(X.loc[136565 ,:]))

print(y[136565 ])



from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=42, test_size=0.3)


from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neural_network import MLPClassifier

from sklearn.tree import DecisionTreeClassifier


from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score


models = [LogisticRegression(),
KNeighborsClassifier(n_neighbors=3),MLPClassifier(alpha=0.005),
DecisionTreeClassifier()]

classifiers = ["LR", "KNN","MLP","ID3"]

scores = []


for model in models:

    model.fit(X_train,y_train)

    y_pred = model.predict(X_test)

    score = accuracy_score(y_test, y_pred)*100

    scores.append(score)
    print("Accuracy of the model is: ", score)
    conf_matrix = confusion_matrix(y_test,y_pred)
    report = classification_report(y_test,y_pred)
    print("Confusion Matrix:\n",conf_matrix)
    print("Report:\n",report)
    print("\n=============***===============")

plt.plot(classifiers,scores)
plt.title("TCP Sync Attack")
plt.ylim(99.9,100)
plt.show()
```

UDP-checkpoint.ipynb

```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("kdd.csv",index_col=0)

df.head(50)
```

```
udp_df = df[df.loc[:,"protocol_type"] == "udp"]


udp_df.to_csv("udp.csv")


udp_df.head()


#there are no null values from priori knowledge


service_values = np.unique(udp_df.loc[:,"service"])

mid = (len(service_values)+1)/2

for i in range(len(service_values)):

    udp_df = udp_df.replace(service_values[i], (i-mid)/10)


udp_df.head(50)


#I will be extracting all the important features as a "priori" for preprocessing

features =
["service","src_bytes","dst_bytes","wrong_fragment","count","num_compromi
sed","srv_count","dst_host_srv_count","dst_host_diff_srv_rate"]

target = "result"


X = udp_df.loc[:,features]

y = udp_df.loc[:,target]
```

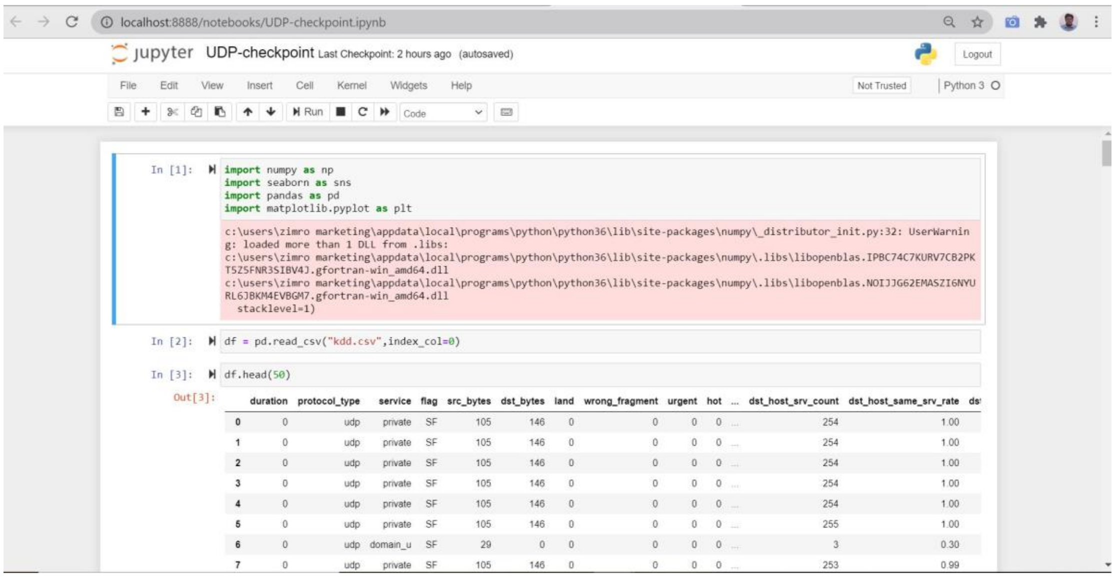
```
classes = np.unique(y)

print(classes)


#replacing all classes of attack with 1 and normal result with 0 in our icmp_df

for i in range(len(classes)):

    if i == 1:

        udp_df = udp_df.replace(classes[i], 0)

    else:

        udp_df = udp_df.replace(classes[i], 1)


#turning the service attribute to categorical values

udp_df=udp_df.replace("eco_i",-0.1)

udp_df=udp_df.replace("ecr_i",0.0)

udp_df=udp_df.replace("tim_i",0.1)

udp_df=udp_df.replace("urp_i",0.2)


udp_df.head(50)


#I selected certain features but I will have to find some covariance between
them so I will plot a covariance heatmap

sns.heatmap(X.corr(), annot=True,cmap="RdBu")

plt.plot()

#the data as seen is highly uncorrelated as most of it is one valued such as the
duration one.
```

```
from sklearn.ensemble import RandomForestClassifier


y = udp_df.loc[:,target]


rs = RandomForestClassifier()

rs.fit(X,y)

print(pd.Series(rs.feature_importances_,index=features).sort_values(ascending
=False))


#updated feature selection to release overfitting and accuracy improvement

#in generalised prediction

X = X.loc[:,["dst_bytes","service","src_bytes","dst_host_srv_count","count"]]

X.head(20)


print(list(X.loc[32,:])) #5 input


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42,
test_size=0.3)


from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier


from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score


models = [LogisticRegression(),
KNeighborsClassifier(n_neighbors=3),MLPClassifier(alpha=0.005),DecisionTr
eeClassifier()]

classifiers = ["LR", "KNN","MLP","ID3"]

scores = []


for model in models:

    model.fit(X_train,y_train)

    y_pred = model.predict(X_test)

    score = accuracy_score(y_test, y_pred)*100

    scores.append(score)

    print("Accuracy of the model is: ", score)

    conf_matrix = confusion_matrix(y_test,y_pred)

    report = classification_report(y_test,y_pred)

    print("Confusion Matrix:\n",conf_matrix)

    print("Report:\n",report)

    print("\n============***==============")
```

```
plt.plot(classifiers,scores)

plt.title("UDP ATTACK")

plt.ylim(50,80)

plt.show()
```

OUTPUT

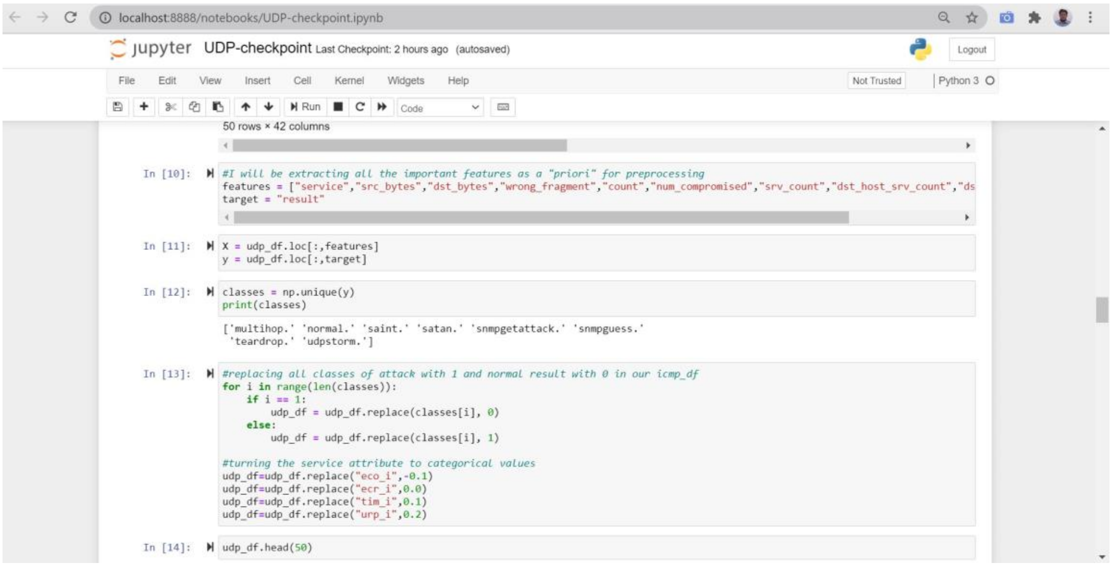
1.Importing Modules



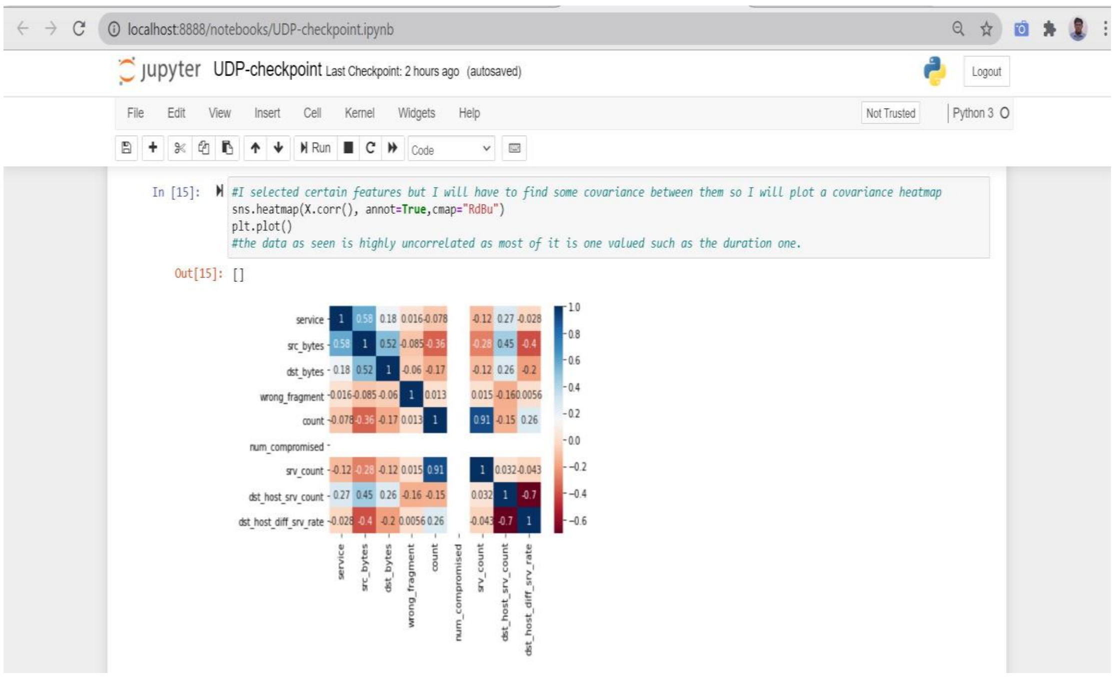
```
In [1]:  import numpy as np
         import seaborn as sns
         import pandas as pd
         import matplotlib.pyplot as plt

         c:\users\zimro marketing\appdata\local\programs\python\python36\lib\site-packages\numpy\_distributor_init.py:32: UserWarnin
         g: loaded more than 1 DLL from .libs:
         c:\users\zimro marketing\appdata\local\programs\python\python36\lib\site-packages\numpy\.libs\libopenblas.IPBC74C7KURV7CB2PK
         T5Z5FNR3SIBV4J.gfortran-win_amd64.dll
         c:\users\zimro marketing\appdata\local\programs\python\python36\lib\site-packages\numpy\.libs\libopenblas.NOIJJG62EMASZI6NYU
         RL6JBKM4EVBGM7.gfortran-win_amd64.dll
           stacklevel=1)

In [2]:  df = pd.read_csv("kdd.csv",index_col=0)

In [3]:  df.head(50)

Out[3]:
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	ds
0	0	udp	private	SF	105	146	0	0	0	0	...	254	1.00	
1	0	udp	private	SF	105	146	0	0	0	0	...	254	1.00	
2	0	udp	private	SF	105	146	0	0	0	0	...	254	1.00	
3	0	udp	private	SF	105	146	0	0	0	0	...	254	1.00	
4	0	udp	private	SF	105	146	0	0	0	0	...	254	1.00	
5	0	udp	private	SF	105	146	0	0	0	0	...	255	1.00	
6	0	udp	domain_u	SF	29	0	0	0	0	0	...	3	0.30	
7	0	udp	private	SF	105	146	0	0	0	0	...	253	0.99	

2Data Preprocessing



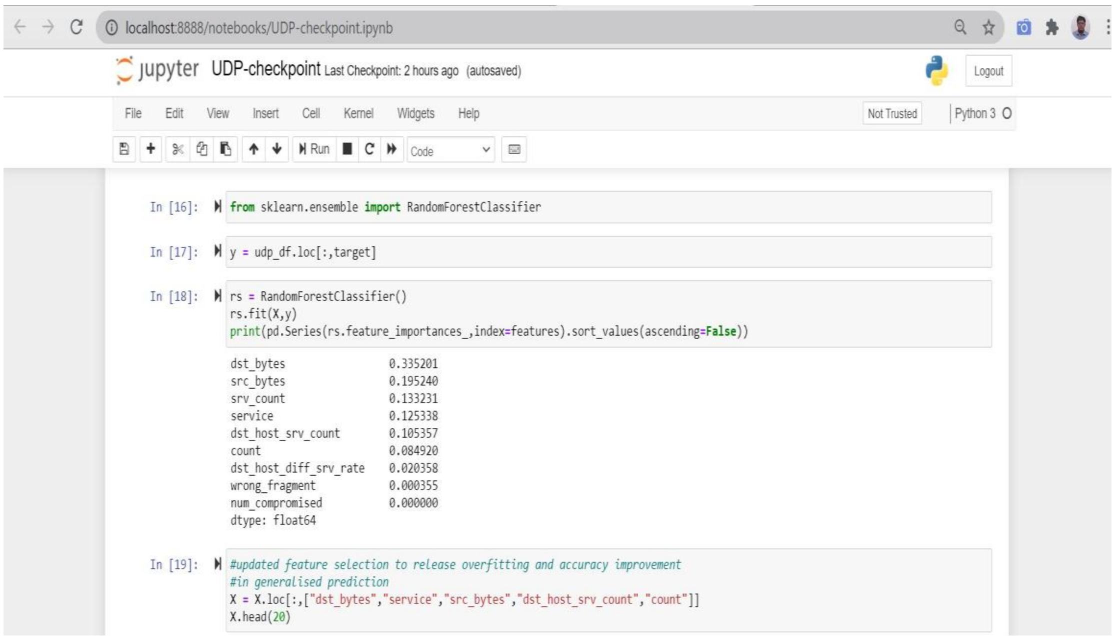
```
50 rows × 42 columns

In [10]: #I will be extracting all the important features as a "priori" for preprocessing
         features = ["service","src_bytes","dst_bytes","wrong_fragment","count","num_compromised","srv_count","dst_host_srv_count","ds
         target = "result"

In [11]: X = udp_df.loc[:,features]
         y = udp_df.loc[:,target]

In [12]: classes = np.unique(y)
         print(classes)

         ['multihop.' 'normal.' 'saint.' 'satan.' 'snmpgetattack.' 'snmpguess.'
          'teardrop.' 'udpstorm.']

In [13]: #replacing all classes of attack with 1 and normal result with 0 in our icmp_df
         for i in range(len(classes)):
             if i == 1:
                 udp_df = udp_df.replace(classes[i], 0)
             else:
                 udp_df = udp_df.replace(classes[i], 1)

         #turning the service attribute to categorical values
         udp_df=udp_df.replace("eco_i",-0.1)
         udp_df=udp_df.replace("ecr_i",0.0)
         udp_df=udp_df.replace("tim_i",0.1)
         udp_df=udp_df.replace("urp_i",0.2)

In [14]: udp_df.head(50)
```


localhost:8888/notebooks/UDP-checkpoint.ipynb
jupyter UDP-checkpoint Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

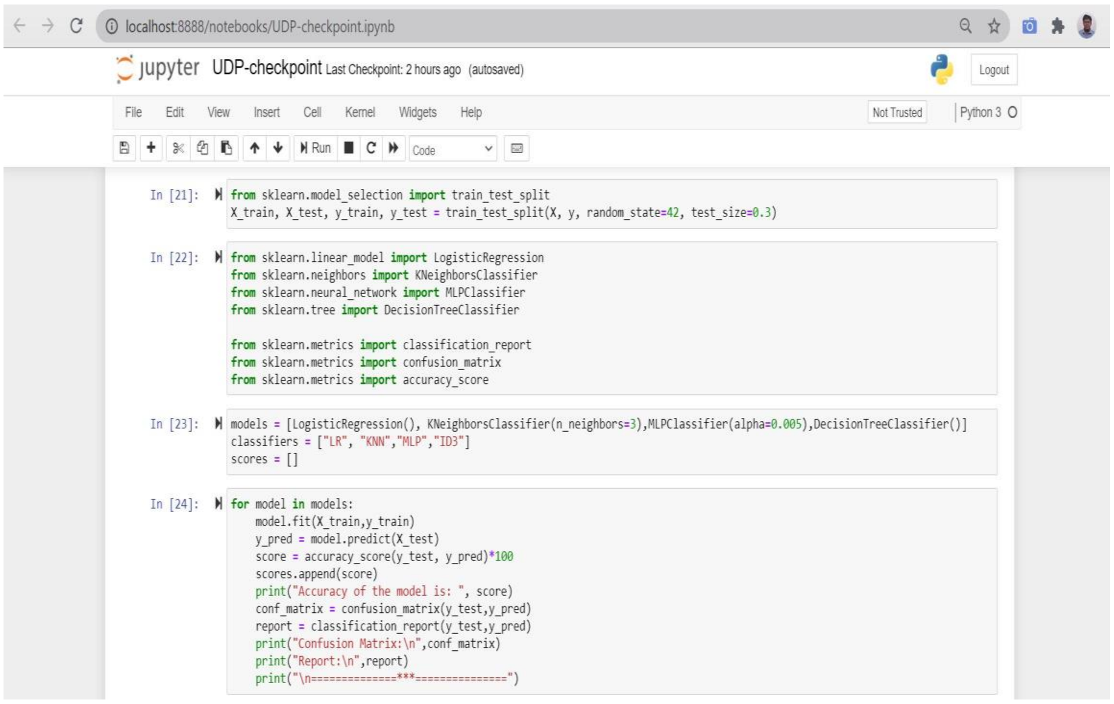
```
In [15]: #I selected certain features but I will have to find some covariance between them so I will plot a covariance heatmap
         sns.heatmap(X.corr(), annot=True,cmap="RdBu")
         plt.plot()
         #the data as seen is highly uncorrelated as most of it is one valued such as the duration one.

Out[15]: []
```

	service	src_bytes	dst_bytes	wrong_fragment	count	num_compromised	srv_count	dst_host_srv_count	dst_host_diff_srv_rate
service	1	0.58	0.18	0.016	-0.078		-0.12	0.27	-0.028
src_bytes	0.58	1	0.52	-0.085	-0.36		-0.28	0.45	-0.4
dst_bytes	0.18	0.52	1	-0.06	-0.17		-0.12	0.26	-0.2
wrong_fragment	-0.016	-0.085	-0.06	1	0.013		0.015	-0.16	0.0056
count	-0.078	-0.36	-0.17	0.013	1		0.91	-0.15	0.26
num_compromised									
srv_count	-0.12	-0.28	-0.12	0.015	0.91		1	0.032	-0.043
dst_host_srv_count	0.27	0.45	0.26	-0.16	-0.15		0.032	1	-0.7
dst_host_diff_srv_rate	-0.028	-0.4	-0.2	0.0056	0.26		-0.043	-0.7	1

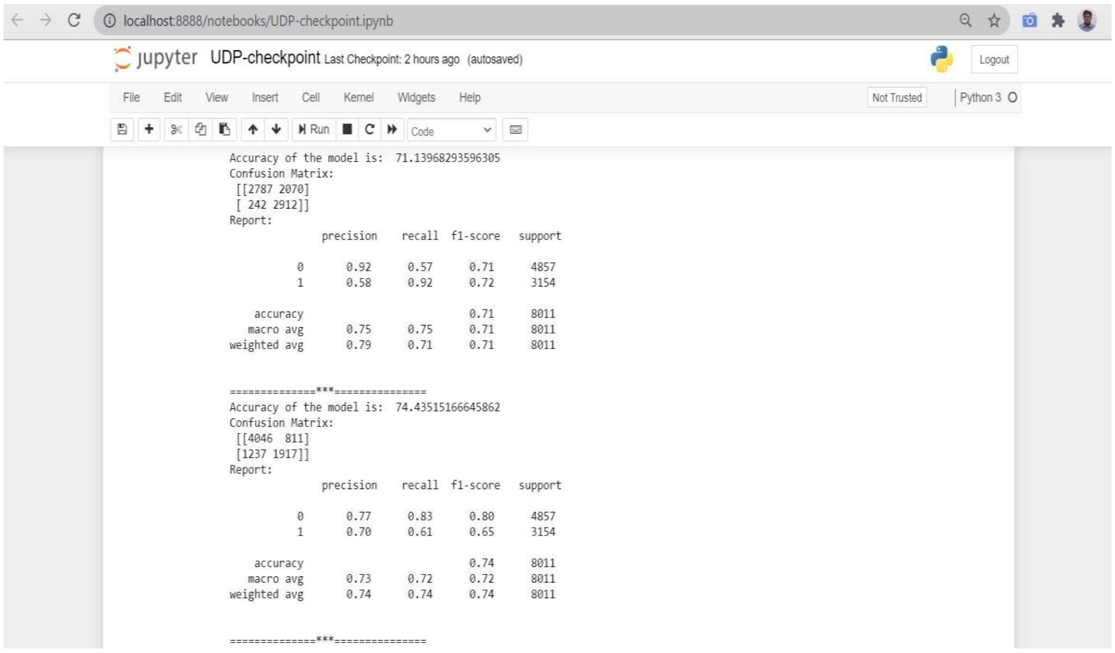
3.Machine Learning Algorithms


localhost:8888/notebooks/UDP-checkpoint.ipynb
jupyter UDP-checkpoint Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

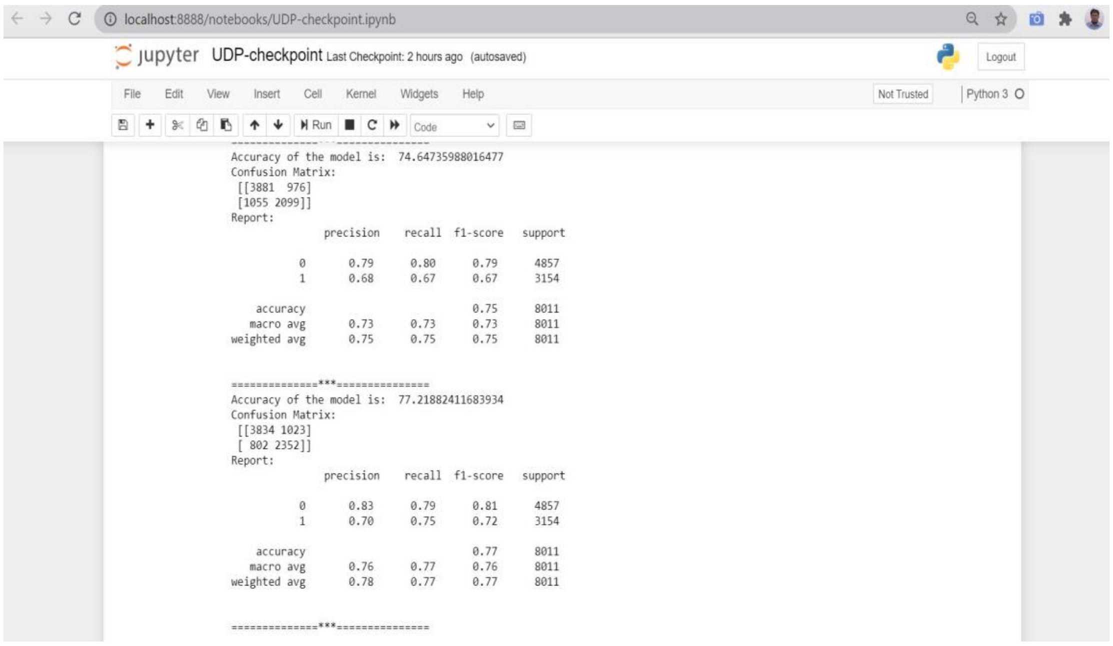
```
In [16]: from sklearn.ensemble import RandomForestClassifier

In [17]: y = udp_df.loc[:,target]

In [18]: rs = RandomForestClassifier()
         rs.fit(X,y)
         print(pd.Series(rs.feature_importances_,index=features).sort_values(ascending=False))

         dst_bytes               0.335201
         src_bytes               0.195240
         srv_count               0.133231
         service                 0.125338
         dst_host_srv_count      0.105357
         count                   0.084920
         dst_host_diff_srv_rate  0.020358
         wrong_fragment          0.000355
         num_compromised         0.000000
         dtype: float64

In [19]: #updated feature selection to release overfitting and accuracy improvement
         #in generalised prediction
         X = X.loc[:,["dst_bytes","service","src_bytes","dst_host_srv_count","count"]]
         X.head(20)
```


localhost:8888/notebooks/UDP-checkpoint.ipynb
jupyter UDP-checkpoint Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

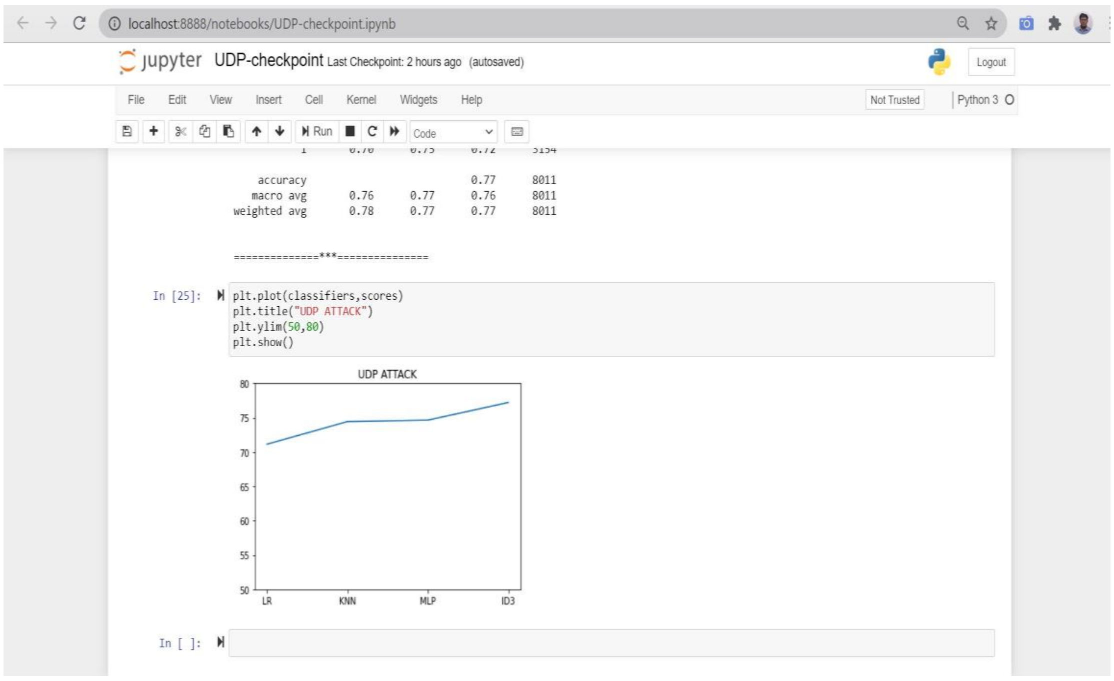
```
In [21]:  from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.3)

In [22]:  from sklearn.linear_model import LogisticRegression
          from sklearn.neighbors import KNeighborsClassifier
          from sklearn.neural_network import MLPClassifier
          from sklearn.tree import DecisionTreeClassifier

          from sklearn.metrics import classification_report
          from sklearn.metrics import confusion_matrix
          from sklearn.metrics import accuracy_score

In [23]:  models = [LogisticRegression(), KNeighborsClassifier(n_neighbors=3),MLPClassifier(alpha=0.005),DecisionTreeClassifier()]
          classifiers = ["LR", "KNN","MLP","ID3"]
          scores = []

In [24]:  for model in models:
              model.fit(X_train,y_train)
              y_pred = model.predict(X_test)
              score = accuracy_score(y_test, y_pred)*100
              scores.append(score)
              print("Accuracy of the model is: ", score)
              conf_matrix = confusion_matrix(y_test,y_pred)
              report = classification_report(y_test,y_pred)
              print("Confusion Matrix:\n",conf_matrix)
              print("Report:\n",report)
              print("\n==============***===============")
```


localhost:8888/notebooks/UDP-checkpoint.ipynb
jupyter UDP-checkpoint Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
Accuracy of the model is:  71.13968293596305
Confusion Matrix:
 [[2787 2070]
 [ 242 2912]]
Report:
              precision    recall  f1-score   support

           0       0.92      0.57      0.71      4857
           1       0.58      0.92      0.72      3154

    accuracy                           0.71      8011
   macro avg       0.75      0.75      0.71      8011
weighted avg       0.79      0.71      0.71      8011


==============***===============
Accuracy of the model is:  74.43515166645862
Confusion Matrix:
 [[4046  811]
 [1237 1917]]
Report:
              precision    recall  f1-score   support

           0       0.77      0.83      0.80      4857
           1       0.70      0.61      0.65      3154

    accuracy                           0.74      8011
   macro avg       0.73      0.72      0.72      8011
weighted avg       0.74      0.74      0.74      8011


==============***===============
```


localhost:8888/notebooks/UDP-checkpoint.ipynb
jupyter UDP-checkpoint Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
Accuracy of the model is:  74.64735988016477
Confusion Matrix:
 [[3881  976]
 [1055 2099]]
Report:
              precision    recall  f1-score   support

           0       0.79      0.80      0.79      4857
           1       0.68      0.67      0.67      3154

    accuracy                           0.75      8011
   macro avg       0.73      0.73      0.73      8011
weighted avg       0.75      0.75      0.75      8011


==============***===============
Accuracy of the model is:  77.21882411683934
Confusion Matrix:
 [[3834 1023]
 [ 802 2352]]
Report:
              precision    recall  f1-score   support

           0       0.83      0.79      0.81      4857
           1       0.70      0.75      0.72      3154

    accuracy                           0.77      8011
   macro avg       0.76      0.77      0.76      8011
weighted avg       0.78      0.77      0.77      8011


==============***===============
```


localhost:8888/notebooks/UDP-checkpoint.ipynb
jupyter UDP-checkpoint Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
UDP ATTACK
80
75
70
65
60
55
50
LR
KNN
MLP
ID3

```
    accuracy                           0.77      8011
   macro avg       0.76      0.77      0.76      8011
weighted avg       0.78      0.77      0.77      8011


==============***===============
```

```
In [25]: plt.plot(classifiers,scores)
         plt.title("UDP ATTACK")
         plt.ylim(50,80)
         plt.show()
```

```
In [ ]:
```

CHAPTER 6

CONCLUSION

Algorithms based on Machine Learning were implemented successfully showing different accuracies. NSL-KDD dataset was preprocessed using mean normalization method. Linear regression, surprisingly, proved to be very effective in detecting network attacks with a 80% accuracy. K-Means Clustering being a semi-supervised approach showed decent results with a 67.5 % accuracy. Neural Networks was implemented with one hidden layer one time and two hidden layers other time. With two hidden layers, it proved to be the best among all the approaches above. But there is always a trade-off between the accuracy and time an algorithm takes. Neural Network took the most time to get trained while K-Means Clustering took lowest amount of time.

6.1 FUTURE WORKS

Network Intrusion Detection can be improved using latest machine learning techniques like deep neural network, dbscan etc.Here we have not used feature selection to select only relevant features. Using feature selection time performance can be improved as well as accuracy. Dimensionality Reduction using principal component analysis can be used to improve the time and visualization.

The NSL-KDD dataset is very old and there are some bugs which can be tackled using a well derived dataset.

REFERENCES

[1] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, ‘‘A deep learning approach to network intrusion detection,’’ IEEE Trans. Emerg. Topics Comput. Intell.,vol. 2, no. 1, pp. 41–50, Feb. 2018.

[2] C. Yin, Y. Zhu, J. Fei, and X. He, ‘‘A deep learning approach for intrusion detection using recurrent neural networks,’’ IEEE Access, vol. 5,pp. 21954–21961, 2017.

[3] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, ‘‘A deep learning approach for network intrusion detection system,’’ in Proc. 9th EAI Int. Conf. BioInspired Inf. Commun. Technol. (BICT), 2015, pp. 21–26.

[4] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho,‘‘Deep learning approach for network intrusion detection in software defined networking,’’ in Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM), Oct. 2016, pp. 258–263.

[5] Y. LeCun et al., ‘‘Backpropagation applied to handwritten zip code recognition,’’ Neural Comput., vol. 1, no. 4, pp. 541–551, 1989.

[6] C. Szegedy et al. (2013). ‘‘Intriguing properties of neural networks.’’ [Online]. Available: https://arxiv.org/abs/1312.6199

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy. (2014). ‘‘Explaining and harnessing adversarial examples.’’ [Online]. Available: https://arxiv.org/abs/1412.6572

[8] A. Kurakin, I. Goodfellow, and S. Bengio. (2016). ‘‘Adversarial examples in the physical world.’’ [Online]. Available: https://arxiv.org/abs/1607.02533

[9] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, ‘‘The limitations of deep learning in adversarial settings,’’ in Proc. IEEE Eur. Symp. Secur. Privacy, Nov. 2015, pp. 372–387.

[10] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, ‘‘DeepFool: A simple and accurate method to fool deep neural networks,’’ in Proc. IEEE Conf.Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 2574–2582.

[11] N. Carlini and D. Wagner, ‘‘Towards evaluating the robustness of neural networks,’’ in Proc. IEEE Symp. Secur. Privacy, Mar. 2017, pp. 39–57.

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, ‘‘A detailed analysis of the KDD CUP 99 data set,’’ in Proc. IEEE Symp. Comput. Intell. Secur.Defense Appl. (CISDA), Jul. 2009, pp. 1–6.

[13] KDD Cup 99 Dataset. Accessed: Apr. 21, 2018. [Online]. Available:http://kdd.ics.uci.edu/ databases/kddcup99/kddcup99.html [14] NSL-KDD Dataset. Accessed: Apr. 21, 2018. [Online]. Available:http://www.unb.ca/cic/research/datasets/nsl.html

[15] TensorFlow. Accessed: Apr. 21, 2018. [Online]. Available: https://www. tensorflow.org

PUBLICATION

INTRUSION DETECTION SYSTEM FOR CLOUD ENVIRONMENTS BASED SUPERVISED MACHINE LEARNING ALGORITHMS was published in International Journal of Scientific Research in Engineering and Management (IJSREM) Volume 05, Issue 03, March 2021.

INTRUSION DETECTION SYSTEM FOR CLOUD ENVIRONMENTS BASED SUPERVISED MACHINE LEARNING ALGORITHMS

Mr. M.Krishnamoorthy
Asst Professor
Panimalar Engineering College
Chennai,TamilNadu

Hritish S
Computer Science and Engineering
PanimalarEngineering College,
Chennai,TamilNadu

Gokulakrishnan P
Computer Science and Engineering
PanimalarEngineering College,
Chennai,TamilNadu

Jebinesh E H
Computer Science and Engineering College
Panimalar Engineering College
Chennai,TamilNadu

***Abstract*—There are many risks of network attacks under the cloud environment. Internet Security may be a vital issue and thus , the intrusion detection is one major research problem for business and private networks to resist external attacks. A Network Intrusion Detection System (NIDS) may be a software application that monitors the network or system activities for malicious activities and unauthorized access to devices. The goal of designing a NIDS is to guard data's confidentiality and integrity.Our paper focuses on these issues with the assistance of supervised Machine Learning algorithm to find out the patterns of the attacks classfication , NSL-KDD dataset has been used.**

***Keywords*—Intrusion, attack, confidentiality, network, unauthorized**

I. INTRODUCTION

The appealing features of Cloud computing still fuel its integration in many sectors including industry, governments, education, entertainment, to call few Cloud computing aims to supply convenient, on-demand, network access to a shared pool of configurable computing resources, which may be rapidly provisioned and released with minimal management effort or service provider Interactions.The pay-as-you-go and therefore the on-demand elastic operation Cloud characteristics are changing the enterprise computing model, shifting on-premises infrastructures to office premises data centers, accessed over the web and managed by cloud hosting providers. However, many security issues arise with the transition to the present computing paradigm including intrusions detection.Regardless the important evolution of the knowledge security technologies in recent years, intrusions and attacks still defeat existing intrusion detection systems in Cloud environments . Attackers developed new sophisticated techniques ready to bring down a whole Cloud platform or maybe many within minutes. New records are reached annually by attackers. Intrusion and attack tools became more sophisticated, challenging existing Cloud IDS by large volumes of network traffic data, dynamic and sophisticated behaviors and new sorts of attacks.
Jebinesh E H
Computer Science and Engineering College

It is clear that an IDS for Cloud should analyze large volumes of network traffic data, detect efficient the new attack behaviors and reach high accuracy with low false. However preprocessing, analyzing and detecting intrusions in Cloud environments using traditional techniques became very costly in terms of computation, time and budgets.

II. OBJECTIVE

The Main Objective of the paper is designing a NIDS is to protect data confidentiality and integrity. Our paper focuses on these issues with the help of Machine Learning.

III. TYPE OF ATTACK

A. Eavesdropping

In general, the bulk of network communications occurin an unsecured or "clear text" format, which allows an attacker who has gained access to data paths in your network to "listen in" or interpret (read) the traffic..

B. Data Modification

After an attacker has read your data, subsequent logical step is to change it. An attacker can modify the info within the packet without the knowledge of the sender or receiver. albeit you are doing not require confidentiality for all communications, you are doing not want any of your messages to be modified in transit. for instance , if you're exchanging purchase requisitions, you are doing not want the things , amounts, or billing information
to be modified.

C. Identity Spoofing (IP Address Spoofing)

Most networks and operating systems use the IP address of a computer to spot a legitimate entity. In certain cases, it's possible for an IP address to be falsely assumed— identity spoofing

D. Sniffer Attack

A sniffer is an application or device which will read, monitor, and capture network data exchanges and skim network packets. If the packets aren't encrypted, a sniffer provides a full view of the info inside the packet.

E. Application-Layer Attack

An application-layer attack targets application servers by deliberately causing a fault in during a server's OS or applications.

F. Password-Based Attacks

A common denominator of most OS and network security plans is password-based access control. This suggests your access rights to a computer and network resources are determined by who you're, that is, your user name and your password.

G. Man-in-the-Middle Attack

As the name indicates, a man-in-the-middle attack occurs when someone between you and therefore the person with whom you're communicating is actively monitoring, capturing, and controlling your communication transparently.

IV. THE EXISTING SYSTEM

- A false positive may be a situation where something abnormal (as defined by the IDS) happens, but it's not an intrusion.
- Users will quit monitoring IDS because of noise.
- No Confidentiality. Not real time network data implemented.

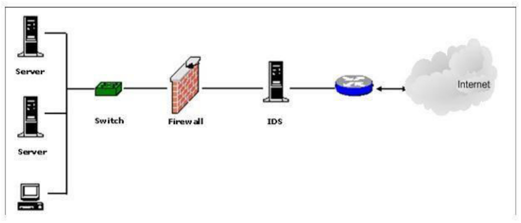
V. EXISTING DISADVANTAGES

- No Data confidentiality
- No Data integrity
- Less prediction accuracy.
- Not a real time analysis.
- Existing system not used for a long dataset.
- The prediction of identity malicious activity is not very accurate.

VI. MACHINE LEARNING IN IDS

The pattern of the normal activities and malicious activities can be learned and distinguished.Machine learning can result in higher detection rates, lower false alarm rates.
On dynamic data, the model can be updated and maintained.

VII. BLOCK DIAGRAM OF IDS



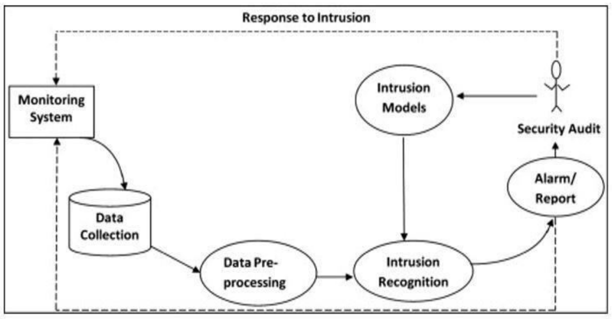
VIII. PROPOSED SYSTEM

- This research focuses on solving the problems in intrusion detection communities which will help the administrator to form pre-processing, classification, labeling of knowledge and to mitigate the result of Distributed Denial of Service Attacks.
- Since, the network administrator feels difficult to pre-process the info . thanks to the overwhelming growth of attacks which makes the task hard, attacks are often identified only after it happens. to beat this example , frequent updating of profiles is required .
- Reduced workload of administrators increases the detection of attacks.

Advantages
- When designing a IDS, the mission is to guard the data's

Confidentiality – read
Integrity – read/write
- Real time data analytics.
- High accuracy.

IX. ARCHITECTURE DIAGRAM



X. MODULES

- Data collection.
- Data preprocessing.
- Machine learning algorithm
- Training and testing

A.MODULES DESCRIPTION

1. DATA COLLECTION

Real time data collected from Twitter ,kaggle, UCI , Data.gov,NSL-KDD dataset

Collection of knowledge is one among most vital the keythe foremostthe main and most important tasks of any machine learning papers. Because the input we feed to the algorithms is data. So, the algorithms efficiency and accuracy depends upon the correctness and quality of knowledge collected. therefore the data are going to be the output.

2. Data preprocessing

As you'll see, the dataset contains nominal values also and to coach a model we'd like all numerical values.
Here is that the transformation table that we used. Dataset is extremely large and there's an outsized variation between values, Data Normalization is additionally required for better performance..

3. Machine learning algorithm

The next step is algorithms are applied to data and results are noted and observed. the choice tree and random forest algorithm applied for accuracy at each stage.

4. Training and Testing

Finally after processing of knowledge and training the very next task is clearly testing. this is often where performance of the algorithm, quality of knowledge , and required output all appears out. From the large data set collected 80 percent of the info is employed for training and 20 percent of the info is reserved for testing. Training as discussed before is that the process of creating the machine to find out and giving it the potential to form further predictions supported the training it took.

XI. CONCLUSION

- Algorithms based on Machine Learning were implemented successfully showing different accuracies.
- The NSL-KDD dataset was preprocessed using mean normalization method.
- Linear regression, surprisingly, proved to be very effective in detecting network attacks with a high accuracy.
- Neural Networks were implemented with one hidden layer one time and two hidden layers another time. With two hidden layers, it proved to be the best among all the approaches above.
- But there is always a trade-off between the accuracy and time an algorithm takes. Neural Network took the most time to get trained while K-Means Clustering took the lowest amount of time.

XII. REFERENCE

- [1]A Dynamical Growing Self-Organizing Tree (DGSOT) for Hierarchical Clustering Gene Expression Profiles," Feng Luo, Latifur Khan , Farokh Bastani, I-Ling Yen and J. Zhou, the Bioinformatics Journal, Oxford University Press, UK, 20 16, 2605-2617.
- [2]"Automatic Image Annotation and Retrieval using Weighted Feature Selection" Lei Wang and Latifur Khan to appear in a special issue in Multimedia Tools and Applications, Kluwer Publisher.
- [3]"Hierarchical Clustering for Complex Data" Latifur Khan and Feng Luo, to appear in International Journal on Artificial Intelligence Tools, World Scientific publishers.
- [4]"A New Intrusion Detection System using Support Vector Machines and Hierarchical Clustering" Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham, to appear in VLDB Journal: The International Journal on Very Large Databases, ACM/Springer-Verlag Publishing.
- [5]R. Lippman J. Haines, D. Fried., J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation" , Computer Networks, 34, pp. 579-595, 2000.