

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi-590018, Karnataka, INDIA



PROJECT REPORT

On

**“ML Based Intrusion Detection System for Wireless
Network”**

Submitted in partial fulfillment of the requirements for the VIII Semester

Bachelor of Engineering

In

INFORMATION SCIENCE AND ENGINEERING

For the Academic year

2020-2021

BY

Nitesh Kalal

Arpitha S Bhat

Neel Mani

1PE15IS070

1PE17IS018

1PE17IS052

Under The Guidance Of

Dr. Annapurna D

Head of the Department

Department of Information Science & Engineering, PESIT-BSC



Department of Information Science and Engineering

PESIT Bangalore South Campus

Hosur Road, Bengaluru-560100

PESIT Bangalore South Campus

Hosur Road, Bengaluru-560100

Department of Information Science and Engineering



CERTIFICATE

This is to certify that the project work entitled “**ML Based Intrusion Detection System for Wireless network**” carried out by **Nitesh Kalal, Arpitha S Bhat and Neel Mani** bearing USNs **1PE15IS070, 1PE17IS018 and 1PE15IS052**, in partial fulfillment for the award of Degree of **Bachelors of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi** during the year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for said degree.

Signature of Guide
Dr. Annapurna D
HOD, ISE

Signature of HOD
Dr. Annapurna D
HOD, ISE, PESIT-BSC

Signature of Principal
Dr. Subhash Kulkarni
Principal, PESIT-BSC

External Viva

Name of the Examiners

Signature with date

1.

2.

DECLARATION

We, Nitesh Kalal (1PE15IS070), Arpitha S Bhat (1PE17IS018) and Neel Mani (1PE17IS052) hereby declare that the dissertation entitles, 'ML based Intrusion Detection System for wireless network', is an original work done by us under the guidance of Dr. Annapurna D, Head Of the Department, Department of Information Science and Engineering, PESIT-BSC, Bengaluru, is being submitted in partial fulfillment for the award of B.E. in Information Science and Engineering, VTU of the requirements for 8th semester.

Date: 18/07/2021
candidate

Signature of the

Place: Bangalore

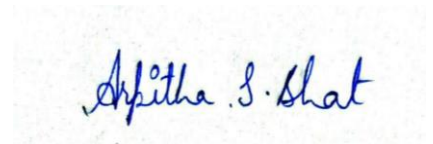
Nitesh Kalal



Neel Mani



Arpitha S Bhat



Acknowledgement

The sense of accomplishment and exhilaration that comes with completing a work would be incomplete if we didn't acknowledge the people who made it possible, whose persistent direction and support crowned our efforts as a success.

Dr. Subhash S Kulkarni, Principal, PESIT Bangalore South Campus, deserves special thanks for providing us with great infrastructure to accomplish our project.

We are grateful to our Project Guide, Dr. Prof. Annapurna D, Head of the Department, for their unwavering support, important contributions, capable guidance, advice, and assistance throughout the project.

At all times, we gladly recognise the assistance provided by the faculty members of the Department of Information Science and Engineering, PESIT Bangalore South Campus. We'd like to take this opportunity to thank our college administration for the resources they gave during the project. Furthermore, we acknowledge the support and feedback of my parents and friends.

Abstract

Threats and attacks against IoT infrastructure are increasing in lockstep with the increased use of IoT infrastructure across all domains. IoT devices transmit data across a wireless connection, making them an easier target for hackers.

Attacks and anomalies that might cause an IoT system failure include Denial of Service, Data Type Probing, Malicious Control, Malicious Operation, Scan, Spying, and Wrong Setup.

As a result, for the security of IoT devices from cybercrime, a secure IoT infrastructure is required. Designing an Intrusion Detection System employing Machine Learning Concepts could be a solution to this challenge.

So, in our project we are implementing an Intrusion Detection System using machine learning models to protect the IoT devices from cyber-attacks.

Contents

Acknowledgement	1
Abstract	2
Contents	3
List Of Figures	5
List Of Tables	6
Chapter 1	7
INTRODUCTION	7
1.1 PURPOSE OF THE PROJECT	7
1.2 SCOPE OF THE PROJECT	7
1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS	8
1.3.1 Definitions	8
1.3.2 ABBREVIATIONS	8
1.4 LITERATURE SURVEY	8
1.5 EXISTING SYSTEM	10
1.6 PROPOSED SYSTEM	11
1.7 PROBLEM STATEMENT	12
1.8 SUMMARY	12
SOFTWARE REQUIREMENTS SPECIFICATION	13
2.1 OPERATING ENVIRONMENT	13
2.1.1 Hardware Requirements	14
2.1.2 Software Requirements	14
2.3. FUNCTIONAL REQUIREMENTS	14
2.4. NONFUNCTIONAL REQUIREMENTS	15
2.5 Advantages of the IDS system	15
2.6 Summary	15
HIGH LEVEL DESIGN	16
3.1 HIGH LEVEL DESIGN	16
3.2 SYSTEM ARCHITECTURE	17
3.3 ALGORITHM ARCHITECTURE	18
3.4 DATA FLOW DIAGRAM	19
3.5 ACTIVITY DIAGRAM	20
3.6 USE CASE DIAGRAM	21
3.7 SEQUENCE DIAGRAM	22

3.8 SUMMARY	23
DETAILED DESIGN	24
4.1 PURPOSE	24
4.2 USER INTERFACE DESIGN	24
4.3 MODULE 1: DATA ACQUISITION	24
4.4 MODULE 2: STATISTICAL VISUALIZATION OF DATA AND FEATURE SELECTION	25
4.5 MODULE 3: DATA PREPROCESSING	26
4.6 MODULE 4: MODELS ASSEMBLED	27
4.7 MODULE 5: TRAIN MODELS	31
4.4 MODULE 6: PREDICTION AND EVALUATION	31
4.8 SUMMARY	32
IMPLEMENTATION	33
5.1.1 Overview of Python	33
5.1.2 Overview of Google Colab	34
5.2 PLATFORM SELECTION	34
5.3 GRAPHICAL USER INTERFACE	35
5.4 PYTHON LIBRARIES	36
5.6 IMPLEMENTATION STEPS	37
5.7 SUMMARY	37
TESTING	38
6.1 SOFTWARE TESTING	38
6.2 PURPOSE	38
6.3 LEVELS OF TESTING	38
6.4 UNIT TESTING	39
6.5 INTEGRATION TESTING	43
6.6 SUMMARY	44
RESULTS	45
7.1 Outcomes	45
7.2 Measurements tables and charts:-	45
CONCLUSION	49
8.1 LIMITATIONS OF THE SYSTEM	49
8.2 FUTURE ENHANCEMENTS	50
REFERENCES	51

List Of Figures

Figure No.	Name of the figure	Page No.
Fig 3.1	High level design	16
Fig 3.2	System architecture	17
Fig 3.3	Deep recurrent neural network algorithm	18
Fig 3.4	Data Flow diagram	19
Fig 3.5	Activity diagram	20
Fig 3.6	Use case diagram	21
Fig 3.7	Sequence diagram	22
Fig 7.1	Graph showing importance of each feature	45
Fig 7.2	Heatmap of Correlation matrix between features	46
Fig 7.3	Model Evaluation	48
Fig 7.4	Confusion matrix	48

List Of Tables

Table No.	Table Name	Page No.
Table 2.1	Hardware Requirements	14
Table 2.2	Software Requirements	14
Table 4.3.1	Dataset Features	25
Table 6.4.1	Unit testing for Data acquisition	39
Table 6.4.2	Unit testing for preprocessing	40
Table 6.4.3	Unit testing for classification	42
Table 6.4.4	Unit testing for model evaluation	43
Table 7.1	Accuracy table	47

Chapter 1

INTRODUCTION

1.1 PURPOSE OF THE PROJECT

Internet of Things (IoT) - IoT is the next evolution of the internet, where almost all the devices have the ability to connect to the internet. With the increasing demand, advancement in the technology and growth within the IoT automated network system, the IoT models have become more complex. IoT devices use a wireless medium to broadcast data which makes them become a major target for the attack..

Machine learning - As attack vectors and threats evolve, traditional security techniques are becoming inefficient and ineffective in addressing the problem of IoT security, as they are unable to update themselves to detect new threats, opening the door for new types of intrusion detection systems based on machine learning and artificial neural networks concepts.

1.2 SCOPE OF THE PROJECT

Security plays a key role in the field of networks so an Intrusion Detection system is used for ensuring security. It can detect a network intrusion, whether it's a current one or one that has already happened. It assists in determining if network traffic is typical or unusual. IDS can determine four types of attack such as Denial Of Service(DOS), User to Root(U2R), Probe(Probing) and Root to Local(R2L).

The main focus is on resolving the issue in a real-world network application setting. We're working on an artificially fully automated intrusion detection system for fog security against cyber-attacks as part of our research. In comparison to other existing models, the suggested model employs deep learning ideas that have a high accuracy rate.

1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

1.3.1 Definitions

IoT - The Internet of Things (IoT) is a network of physical objects that are implanted with sensors, software, and other technologies in order to connect and exchange data with other devices and systems over the internet.

Machine Learning - Machine learning is a branch of artificial intelligence (AI) that allows computers to learn and improve on their own without having to be explicitly programmed. Machine learning is concerned with the creation of computer programmes that can access data and learn on their own.

Intrusion Detection System - An intrusion detection system (IDS) is a type of security software that alerts administrators when someone or something is attempting to infiltrate an information system through malicious activity or security policy violations.

1.3.2 ABBREVIATIONS

IoT - Internet of Things.

ML - Machine Learning.

IDS - Intrusion Detection System

SRS - Software Requirements Specifications

NN - Neural Network

1.4 LITERATURE SURVEY

- DEEP RECURRENT NEURAL NETWORK FOR IOT INTRUSION DETECTION SYSTEM

MuderAlimiani, Alia AbuGhazleh, Amer Al-Rahayfeh, Saleh Atiewi, AbdulRazaque (2020)

This paper first talks about the existing approach used for developing the intrusion detection system along with their disadvantages. Then the paper presents its approach which uses recurrent neural networks. These are designed to detect the specific type of attacks such as DoS and probe attacks.

This paper provides a comparative result with other previous works. Their results show the effectiveness of their approach.

- A REVIEW OF INTRUSION DETECTION SYSTEMS USING MACHINE AND DEEP LEARNING IN THE INTERNET OF THINGS: CHALLENGES, SOLUTIONS AND FUTURE DIRECTIONS

Javed Asharf , Nour Moustafa, Hasnat Khurshid, Essam Debie, Waqas Haider and Abdul Wahab (2020)

In this paper the author has described six important topics step by step. In this paper there is a detailed explanation of IoT architecture and different types of threats that exist. After this the author talks about IDS architecture then there is an introduction to ML and DL approach to IDS design and advantages of using these technologies.

As ML approaches require a dataset to perform operations, the paper presents the various datasets that are available. At last the future research challenges are mentioned such as building a real time IDS, increasing the number of stages increases the computational cost etc.,.

- A SURVEY ON INTRUSION DETECTION SYSTEM FOR WIRELESS NETWORK

Ajita Mishra and Ashish Kumar Srivastava

This paper describes the types of wireless networks such as ad hoc networks, sensor networks and types of attacks in wireless networks such as probing attacks, surveillance, DoS attack and man in the middle attack. Then the paper talks about the requirement of intrusion detection systems for wireless networks. It also introduces us to the Intrusion Prevention System.

In the end it presents the intrusion detection probability based on the intrusion distance and network parameters and prevention of that problem. Their approach jams the whole RF medium when the intruder enters the system but disadvantage of using this is it will block all the network traffic.

- A MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM FOR MOBILE INTERNET OF THINGS

Amar Amouri, Vishwa T. Alaparthi and Salvatore D. Morgera (2020)

This paper first talks about the previous system proposed by them which uses a cross layer-based IDS system which has two layers of detection and it uses a heuristic approach. In this paper the proposed system consists of two stages, in the first stage the dedicated sniffers are used to collect the data that is then converted to correctly classified instances and then sent to the super node which uses linear regression to perform analysis.

The proposed system is trained to detect blackhole and distributed denial of service attacks. According to the paper the detection rate increases for high velocity nodes as compared to low velocity nodes whose detection rate is 90%.

The limitation related to this IDS is the detection of the false positive rate. In this paper the authors have suggested a solution to use the filtering techniques at the early stages.

- DEEP LEARNING APPROACH FOR INTRUSION DETECTION SYSTEM (IDS) IN THE INTERNET OF THINGS (IOT) NETWORK USING RECURRENT NEURAL NETWORKS

Manoj Kumar Putchala(2017)

This paper holds a vast volume of data, describing in detail about Literature review on the Internet of Things and the applications of Machine Learning and Deep Learning for IoT. It also presents a variety of approaches for network security in an IoT system and compares the results with the results of other machine learning and deep learning research on IDS.

1.5 EXISTING SYSTEM

The existing system employs some traditional methods as well as various machine learning techniques to design an intrusion detection system, including fuzzy logic, random forest, neural networks, and support vector machines, among others, but their ability to classify incoming packets is inferior to that of neural networks. When existing systems are employed,

we can see more skewed results, and low frequency packets may be missed in some circumstances, resulting in poor performance and classification accuracy.

1.6 PROPOSED SYSTEM

We will provide the description of the concepts such as IoT, fog computing, growth of IoT, security importance with advancement in IoT technologies, security challenges that exist and other related security concerns and common attacks found in it. We also present a description of different machine learning based approaches and their usage in security and how its usage will increase the efficiency and accuracy.

We want to show an intrusion detection system that uses fog security to protect against cyber-attacks, and our major goal is to construct a model that uses neural networks (NN). To do so, we've implemented a number of machine learning models alongside the neural network to compare their performance. To train the dataset, the model employs the weight update technique and the forward propagation algorithm. The use of RNN is that it classifies the incoming packet not only based on the current characteristics that are present, but also takes into account the history or the classification result of the previous input and learns to classify the packet on its own.

We create an intrusion detection system with cascaded filtering stages, where artificial neural networks and recursive neural networks are utilised for each filter, and they are configured to catch specific types of assaults common in IoT contexts, such as DoS, Probe, R2L, and U2R. As a result, our project primarily distinguishes four categories of attacks, which is accomplished by pre-processing the dataset and removing the undesirable features. As a result, we're comparing our system's performance against that of other systems to show that ours is superior.

1.7 PROBLEM STATEMENT

The Internet of Things (IoT) is the next step in the internet's evolution. Because there are so many devices connected to the internet and also connected to each other, massive volumes of traffic and digital data are generated and transferred. Attacks and penetrations on the system are becoming more common as IOT technologies improve. As threats evolve, traditional security solutions become inefficient and inadequate in addressing IoT security, a new generation of intrusion detection systems based on machine learning and artificial neural networks emerges. So our approach is to use various machine learning models for the classification of the packet as malicious/normal packets and compare their performances.

1.8 SUMMARY

This chapter included a brief introduction to the project's aim and scope, as well as a brief overview of the suggested methodology and problem statement. It also included a literature survey to gather all of the material needed for the project.

Chapter 2

SOFTWARE REQUIREMENTS SPECIFICATION

A Software Requirements Specification (SRS) describes how the software will perform and what it will do and includes the functional and nonfunctional requirements for the software to be developed. The functional requirements involve what the software should do and non-functional requirements involve how the system should implement the designed software.. Requirements must be measurable, testable, related to identified needs or opportunities, and defined to a level of detail sufficient for system design.

Software requirement specification will state what the software will do. When the software must be directly perceived by its users – either human users or other software systems. The common understanding between the user and the developer is captured in the requirements document. By writing down the software requirement specifications we can reduce the development effort as careful review of the document happens before starting the implementation which can reveal omissions, misunderstandings, and inconsistencies early in the development cycle which is easier to correct at the early stage. The SRS mostly talks about the product and not the project that is used to develop it so it helps for later enhancement of the finished product. The SRS may need to be altered, but it does provide a foundation for continued production evaluation.

2.1 OPERATING ENVIRONMENT

Our proposed system will work on any system with the latest version of the Google Chrome Browser or Mozilla Firefox and should have an Internet Connection to run the program.

2.1.1 Hardware Requirements

Table 2.1: Hardware Requirements

HARDWARE	DESCRIPTION
Processor	Intel(R) Core(TM) i3and above
RAM	4.00 GB and above
Hard Disk	128GB and above

2.1.2 Software Requirements

Table 2.2: Software Requirements

SOFTWARE	DESCRIPTION
Operating System	Ubuntu 14.04 and above or newer Windows 10
Programming Languages Used	Python, Pandas, Scikit-learn, Keras

2.3. FUNCTIONAL REQUIREMENTS

These are the requirements which specify how the system should react for the given inputs and what functionality it should provide and behavior of the system.

They are: -

- Process data since data has abnormalities
- Implements dynamic machine learning classification to check for attacks.
- Detect different categories of attacks as attacks can be of any type.

2.4. NONFUNCTIONAL REQUIREMENTS

Availability: It shows how well the system operates when it is subjected to use. It also includes how the system handles all the types of inputs given to the system.

Reliability: The system's capacity to retain functionality over time, as well as how frequently it experiences critical failures.

Robustness: The system correctly handles mistakes and does not crash in the middle of them.

Maintainability: When the user requirements change over time the system should be able to handle the updates made on the system without hindering its performance.

Safety/Security Requirements: Security plays a very important role and the system should be such that it shouldn't be easily manipulated or hacked.

Performance: It shows how the system responds to a piece of input given by the user under a certain workload. System should be able to consistently perform better every time.

2.5 Advantages of the IDS system

- It efficiently prevents any network damage.
- Any intrusion or attack on the network or machine is constantly monitored.
- IDS guarantees that known anomalies are detected quickly and effectively, with a low risk of false alarms.
- It analyses various forms of attacks, finds dangerous content patterns, and assists administrators in tuning, organising, and implementing effective restrictions.

2.6 Summary

This chapter covered both the software and hardware requirements for the project's proper operation. Then we go over the functional and non-functional requirements, which give us an idea of what to expect from the final project after it's completed. The system's users, benefits, and applications are also discussed.

Chapter 3

HIGH LEVEL DESIGN

3.1 HIGH LEVEL DESIGN

The architecture that will be utilised to construct this project is described in high-level design (HLD). The architecture diagram shows the overall system and identifies the primary components that will be built as well as their interfaces. The HLD employs non-technical to somewhat technical terminology that are easily understood by the user. For programmers or developers using our API, low-level design, on the other hand, reveals the logical precise design of each of these pieces. A high-level architectural diagram describing the system's major components and interfaces are generally included in high-level design.

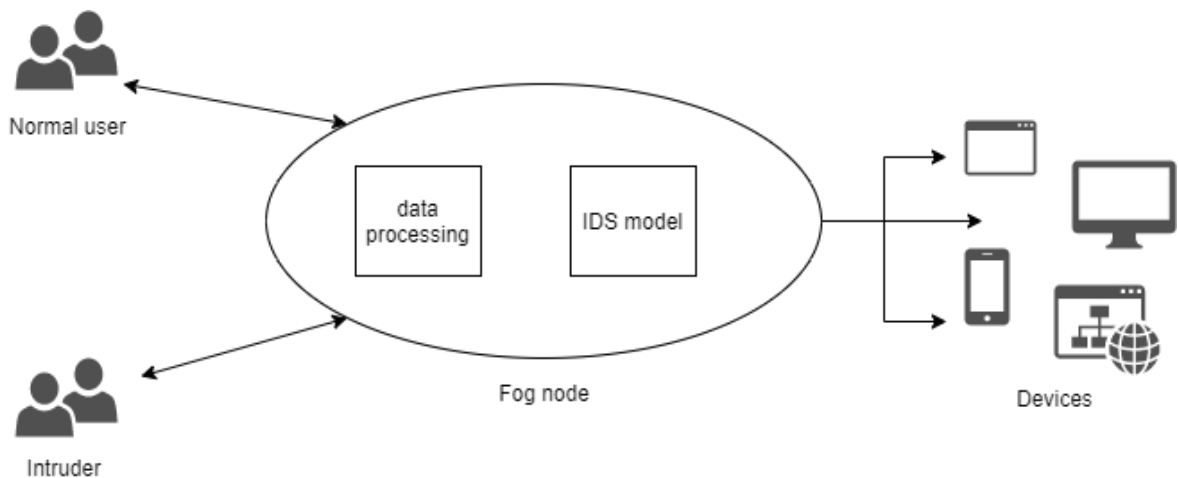


Fig 3.1: High level design

3.2 SYSTEM ARCHITECTURE

The architecture diagram depicts the overall system by identifying the major components that could be produced for the product as well as their interfaces.

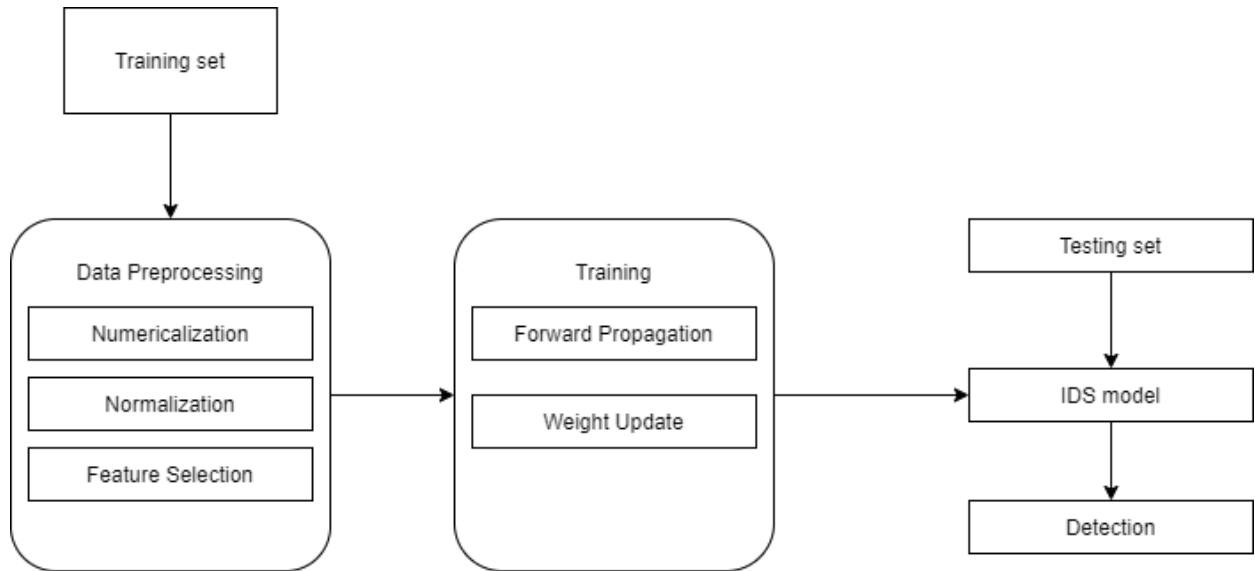


Fig 3.2: System Architecture

Description about the diagram:

The system architecture depicted in the above figure is made up of three layers: the Data Processing Layer, Training, and Testing.

The process consists of the following stages-

1. The data is preprocessed before being separated into two groups. One is used for training and the other is used for testing.
2. The processed data, which has been split for training reasons, is put into the Algorithm, which then learns the classifications.
3. The data that was previously isolated for testing purposes is now being used for evaluation and generating results.

4. When packets are classified as normal, the model is tested for performance, and if any anomalies are discovered, the system notifies the administrator.

3.3 ALGORITHM ARCHITECTURE

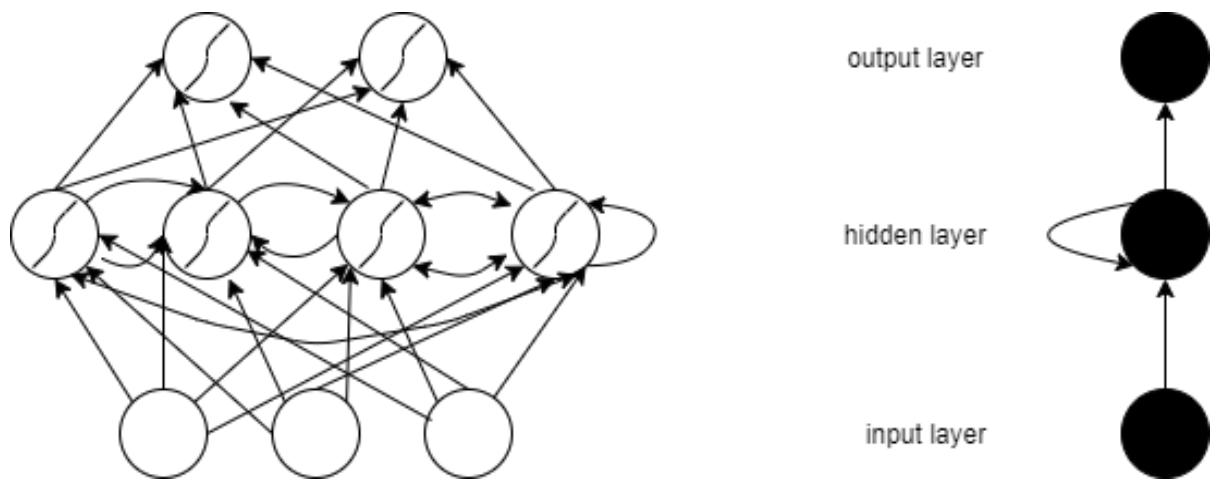


Fig 3.3: Deep recurrent neural network algorithm diagram

3.4 DATA FLOW DIAGRAM

Data flow diagrams depict the flow of data in a corporate information system graphically. A data flow diagram (DFD) depicts the steps in a system's data flow from input through processing to output production. There are two types of data flow diagrams: logical and physical. The logical data flow diagram depicts the movement of data through a system in order to fulfil specific business functions. The physical data flow graphic depicts how the logical data flow is implemented.

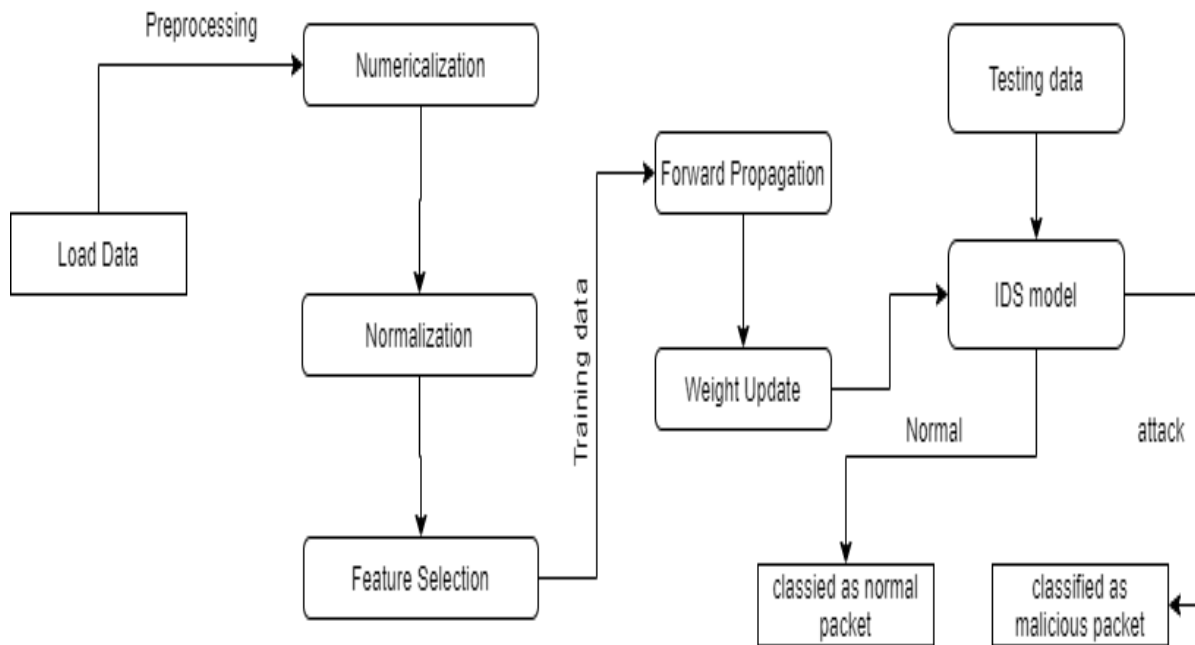


Fig 3.4:Data Flow Diagram.

3.5 ACTIVITY DIAGRAM

An activity diagram is a diagram that emphasises the execution and flow of a system's behaviour rather than its implementation. Behavioral modelling technology uses activity diagrams, which are made up of activities made up of actions. Processes and workflows are modelled using them. A useful activity diagram's essential is to communicate a certain component of a system's dynamic behaviour. They encapsulate a system's dynamic elements.

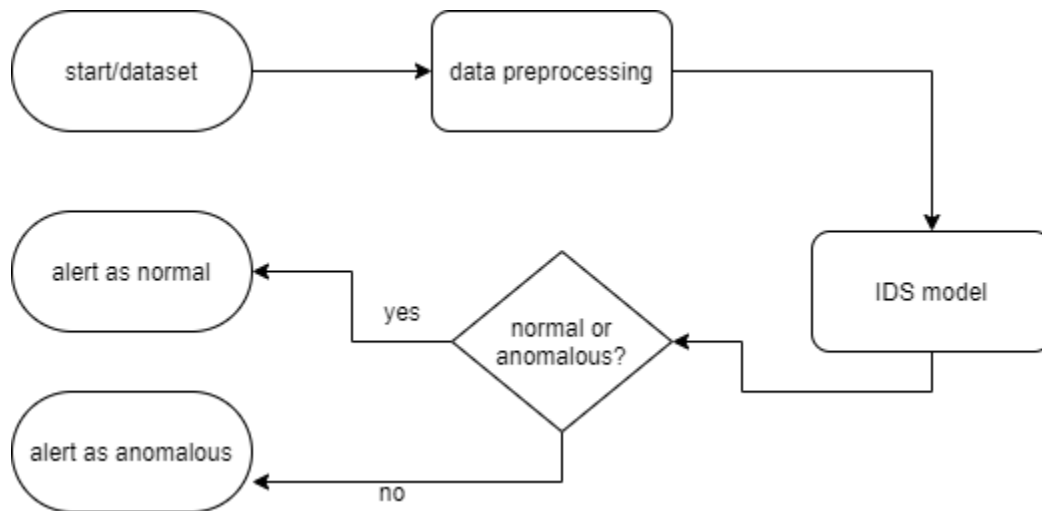


Fig 3.5: Activity diagram

3.6 USE CASE DIAGRAM

At its most basic level, a use case diagram depicts a user's interaction with the system by illustrating the relationship between the user and the many use cases in which the user is involved.

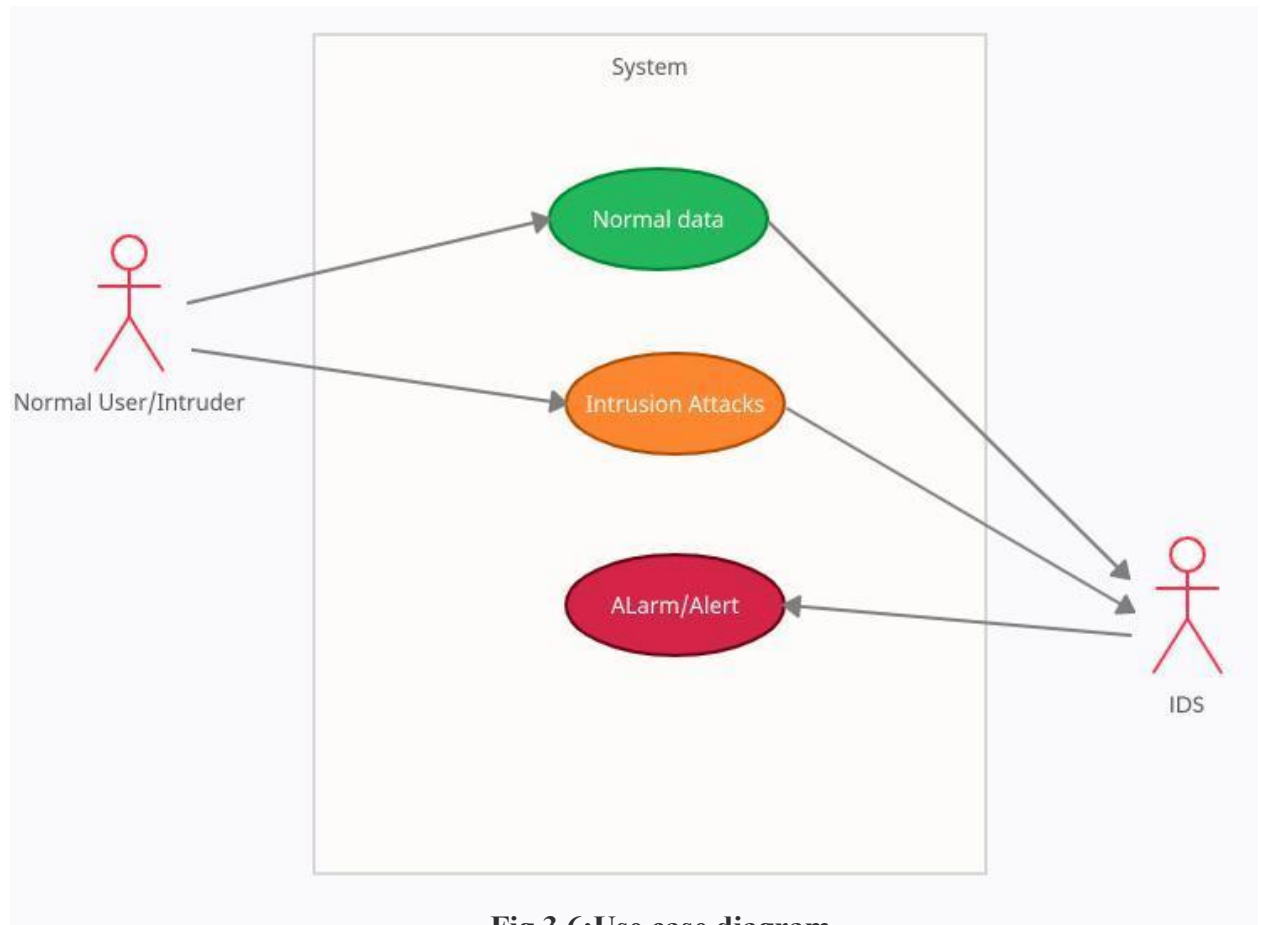


Fig 3.6:Use case diagram

3.7 SEQUENCE DIAGRAM

A sequence diagram simply displays the order in which things interact, or the order in which these interactions occur. A sequence diagram can also be referred to as an event diagram or an event scenario. Sequence diagrams show how and in what order the components of a system work together.

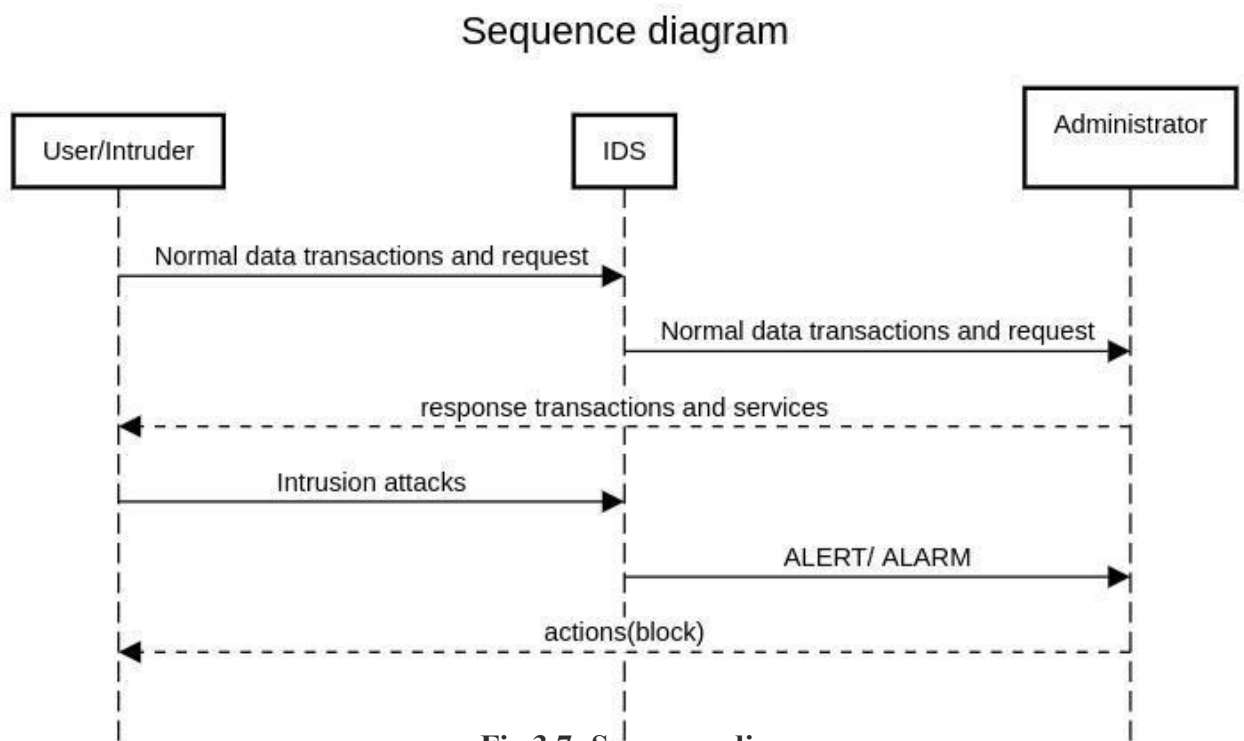


Fig 3.7: Sequence diagram

3.8 SUMMARY

This chapter describes the user's interaction with the system in great detail. We used interaction models including use case diagrams, data flow diagrams, and activity diagrams to describe this. It also includes the general system design, as well as the assumptions and limits to be made while constructing the system.

Chapter 4

DETAILED DESIGN

4.1 PURPOSE

The goal of this chapter is to go over the project's control flow in depth, describing how each module works.

4.2 USER INTERFACE DESIGN

The user interface is created to benefit the end users who don't have too much domain knowledge. The notebook should have a well-defined code structure as well as a straightforward design. The system is made more readable by using Markdown, Code cells, and comments.

4.3 MODULE 1: DATA ACQUISITION

In intrusion detection experiments, the NSL-KDD dataset, which was created in 2009, is widely used. In the most recent literature, all researchers use the NSL-KDD dataset as the benchmark dataset, which not only effectively solves the KDD Cup1999 dataset's inherent redundant records problems, but also keeps the number of records reasonable in the training and testing sets, so that the classifier does not favor more frequent records. The KDDTrain+ dataset serves as the training set, while the KDDTest+ and KDDTest-21 datasets serve as the testing set, containing various normal records and four different types of attack records. The KDDTest-21 dataset is a subset of the KDDTest+ that makes classification more challenging.

Every traffic record has 41 features and one class label, as shown in Table 2. The features comprise basic features (No.1-No.10), content features (No.11 - No.22), and traffic features (No.23 - No.41). The attacks in the dataset are classified into four groups based on their characteristics: DoS (Denial of Service), R2L (Root to Local assaults), U2R (User to Root attack), and Probe (Probing attacks). Some specific attack types are present in the testing set but do not appear in the training set, allowing it to give a more realistic theoretical basis for intrusion detection.

No.	Features	Types	No.	Features	Types
1	duration	Continuous	22	is_guest_login	Symbolic
2	protocol_type	Symbolic	23	count	Continuous
3	service	Symbolic	24	srv_count	Continuous
4	flag	Symbolic	25	serror_rate	Continuous
5	src_bytes	Continuous	26	srv_serror_rate	Continuous
6	dst_bytes	Continuous	27	rerror_rate	Continuous
7	land	Symbolic	28	srv_rerror_rate	Continuous
8	wrong_fragment	Continuous	29	same_srv_rate	Continuous
9	urgent	Continuous	30	diff_srv_rate	Continuous
10	hot	Continuous	31	srv_diff_host_rate	Continuous
11	num_failed_logins	Continuous	32	dst_host_count	Continuous
12	logged_in	Symbolic	33	dst_host_srv_count	Continuous
13	num_compromised	Continuous	34	dst_host_same_srv_rate	Continuous
14	root_shell	Continuous	35	dst_host_diff_srv_rate	Continuous
15	su_attempted	Continuous	36	dst_host_same_src_port_ra	Continuous
16	num_root	Continuous	37	dst_host_srv_diff_host_rat	Continuous
17	num_file_creations	Continuous	38	dst_host_serror_rate	Continuous
18	num_shells	Continuous	39	dst_host_srv_serror_rate	Continuous
19	num_access_files	Continuous	40	dst_host_rerror_rate	Continuous
20	num_outbound_cmds	Continuous	41	dst_host_srv_rerror_rate	Continuous
21	is_host_login	Symbolic			

Table 4.3.1: Dataset features

4.4 MODULE 2: STATISTICAL VISUALIZATION OF DATA AND FEATURE SELECTION

The CSV file is used to feed the dataset. The dataset is statistically evaluated to determine the mean, mode, standard deviation, and other characteristics of each feature. The following are some of the techniques used in this stage:

Exploratory Data Analysis: With the use of summary statistics and graphical representations, exploratory data analysis refers to the crucial process of doing initial investigations on data in order to uncover patterns, spot anomalies, test hypotheses, and check assumptions.

Histograms: The frequency charts of the datasets are called histograms. They are used to visualise data graphically. The frequency distributions are plotted for each attribute, giving a basic indication of the probable distributions.

Feature Elimination: This is a method of repeatedly constructing a classifier on a dataset in order to find the most essential features. Given that we had a two-class problem, we chose Logistic Regression as our model. The model is developed on the dataset over and over again, with the least significant features being discarded in each iteration. This procedure is repeated until the desired number of features has been achieved. Each iteration uses a smaller collection of features as input.

Continuing in this way, 41-dimensional features map into 122-dimensional features after transformation.

4.5 MODULE 3: DATA PREPROCESSING

The data attributes of the chosen dataset that represent networking system input traffic are inherently inconsistent. As a result, pre-treatment of traffic data is a necessary gate for the classification engine.

Traffic Preprocessor applies these pre-processing steps on raw traffic data:

1) NUMERICALIZATION

The NSL-KDD dataset contains 38 numeric characteristics and three non-numerical features. We must convert several non-numerical properties, such as protocol type, service, and flag, into numeric form because RNN-IDS input value should be a numeric matrix. The feature protocol type, for example, includes three types of attributes: 'tcp', 'udp', and 'icmp', and its numeric values are encoded as binary vectors (1,0,0), (0,1,0), and (1,0,1). (0,0,1). The feature 'service' has 70 different types of attributes, while the feature 'flag' has 11 different types of attributes. Continuing in this way, 41-dimensional features map into 122-dimensional features after transformation.

2) MIN-MAX NORMALIZATION

The purpose of normalisation is to equalise the scale of all data points such that each attribute is equally important. The data from the same house has been normalised using min-max normalisation .

One of the most prevalent methods of data normalisation is min-max normalisation. The minimum value of each feature is converted to a 0, the highest value is converted to a 1, and all other values are converted to a decimal between 0 and 1.

4.6 MODULE 4: MODELS ASSEMBLED

The term "classification" refers to the process of determining which category a certain sample data point belongs to. This necessitates training the dataset with classification algorithms in order for it to predict classes for new datasets in the future. Our topic is classified as a two-class problem, which means that there are only two types of networks: attacked and non-attacked.

For classification, ten different algorithms were used, each of which is briefly discussed below:

Logistic Regression

The chance of a given class or event existing, such as pass/fail, win/lose, alive/dead, or healthy/sick, is modelled using a logistic model. This can be used to represent a variety of occurrences, such as determining whether an image contains a cat, dog, lion, or other animal. Each detected object in the image would be assigned a probability ranging from 0 to 1, with a total of one.

Binomial, ordinal, and multinomial logistic regressions are all possible. Binomial or binary logistic regression is used when the observed outcome for a dependent variable can only be one of two types: "0" or "1."

Naive Bayes Classifier

Naive Bayes is a straightforward method for building classifiers, which are models that give class labels to problem cases represented as vectors of feature values, with the class labels selected from a limited set. For training such classifiers, there is no one algorithm, but rather a variety of algorithms based on

the same principle: all naive Bayes classifiers assume that the value of one feature is independent of the value of any other feature, given the class variable.

For example, if a fruit is red, round, and around 10 cm in diameter, it is termed an apple. Regardless of any possible relationships between the colour, roundness, and diameter features, a naive Bayes classifier considers each of these features to contribute independently to the likelihood that this fruit is an apple

Decision Tree Classifier

One of the predictive modelling methodologies used in statistics, data mining, and machine learning is decision tree learning, also known as induction of decision trees. It goes from observations about an item (represented in the branches) to inferences about the item's goal value using a decision tree (as a predictive model) (represented in the leaves). Classification trees are tree models in which the goal variable can take a discrete set of values; in these tree structures, leaves indicate class labels and branches represent feature combinations that lead to those class labels. Regression trees are decision trees in which the target variable can take continuous values (usually real numbers).

Random Forest Classifier

Random forests, also known as random decision forests, are an ensemble learning method for classification, regression, and other problems that works by training a large number of decision trees. For classification tasks, the random forest's output is the class chosen by the majority of trees. The mean or average prediction of the individual trees is returned for regression tasks. Random decision forests address the problem of decision trees overfitting their training set. Random forests outperform decision trees in most cases, but they are less

accurate than gradient enhanced trees. Data features, on the other hand, can have an impact on their performance.

KNeighborsClassifier

The k-nearest neighbours algorithm (k-NN) is a non-parametric classification method invented by Evelyn Fix and Joseph Hodges in 1951 and expanded by Thomas Cover. It is employed in the categorization and regression of data. The input in both situations is the k closest training examples in the data set. Depending on whether k-NN is used for classification or regression, the following is the result: Depending on whether k-NN is used for classification or regression, the following is the result:

The outcome of k-NN classification is class membership. An object is categorised by a majority vote of its neighbours, with the object allocated to the most common class among its k closest neighbours (k is a positive integer, typically small). If $k = 1$, the item is simply assigned to that single nearest neighbor's class.

The output of k-NN regression is the object's property value. This number is the average of the values of the k closest neighbours.

Support Vector Classifier

Support-vector machines are supervised learning models that examine data for classification and regression analysis using related learning techniques. An SVM training algorithm creates a model that assigns new examples to one of two categories, making it a non-probabilistic binary linear classifier, given a series of training examples, each marked as belonging to one of two categories. SVM maps training examples to points in space in order to widen the distance between the two categories as much as possible. New examples are then mapped into the same space and classified according to which side of the gap they fall on.

ANN Classifier :

Artificial neurons are a collection of connected units or nodes in an artificial neural network (ANN) that loosely mimic the neurons in a biological brain. Each link, like synapses in the human brain, can transmit a signal to other neurons. An artificial neuron that receives a

signal, processes it, and sends it to other neurons. The output of each neuron is determined by a nonlinear function of its inputs, and the "signal" at a connection is a real number. Connections are referred to as edges.

As learning develops, the weight of neurons and edges is often modified. Neurons may have a threshold above which they can only transmit a signal if the total signal surpasses it. Neurons are commonly arranged in layers. Separate layers can apply distinct transformations to their inputs. After crossing the layers numerous times, signals move from the first (input) layer to the last (output) layer.

Keras Classifier:

Keras includes a scikit-learn wrapper called KerasClassifier that allows us to apply K-fold cross validation in our Keras code. We must develop a function because the KerasClassifier requires a function as one of its inputs. The goal of this function is to build the architecture of our ANN.

RNN Classifier :

A recurrent neural network (RNN) is a sort of artificial neural network in which nodes are connected in a directed graph in a temporal order. This allows it to respond in a time-dependent manner. RNNs, which are built from feedforward neural networks, can use their internal state to process variable length sequences of inputs (memory). As a result, unsegmented, connected handwriting recognition and speech recognition are now conceivable.

MLP Classifier :

MLP Classifier is an acronym for Multi-layer Perceptron Classifier, which is a type of Neural Network. Unlike other classification methods such as Support Vectors or Naive Bayes Classifier, MLPClassifier uses an underlying Neural Network to do classification.

A multilayer perceptron (MLP) is a feedforward artificial neural network that generates a set of outputs from a set of inputs. Between the input and output layers of an MLP, a directed graph is formed by several layers of input nodes. MLP uses backpropagation to train the network.

4.7 MODULE 5: TRAIN MODELS

Learning (determining) good values for all the weights and the bias from labelled samples is all that training a model involves. A machine learning algorithm generates a model by studying numerous examples and tries to find a model that minimises loss; this process is known as empirical risk minimization.

Train-Test Split: When machine learning algorithms are used to make predictions on data that was not used to train the model, the train-test split process is used to measure their performance. It's a quick and simple technique that allows you to compare the performance of different machine learning algorithms for your predictive modelling problem.

Loss: An incorrect prediction results in a loss. To put it another way, loss is a statistic that indicates how inaccurate the model's forecast was for a single case. The loss is 0 if the model's forecast is perfect; otherwise, the loss is bigger. The purpose of training a model is to discover a set of weights and biases that have a low loss across all cases on average.

Epoch: The term epoch is used in machine learning to describe the number of passes the machine learning algorithm has made through the full training dataset.

Learning Rate: A tuning parameter in an optimization algorithm that affects the step size at each iteration while advancing toward a loss function minimum is called learning rate.

4.4 MODULE 6: PREDICTION AND EVALUATION

When forecasting the likelihood of a specific result, “prediction” refers to the output of an algorithm after it has been trained on a previous dataset and applied to new data.

The performance of the IDS model is measured using the most important performance indicator (Accuracy, AC) for intrusion detection in our model. We also include the detection rate and false positive rate in addition to the accuracy. The True Positive (TP) reflects the number of anomaly records that are detected as anomalies and is equivalent to those

appropriately rejected. The number of normal records that are labelled as abnormal is known as the False Positive (FP). It is the equivalent of mistakenly rejected records. The True Negative (TN) is the number of normal records that are classified as normal, and it is comparable to those correctly admitted. The False Negative (FN) reflects the number of anomaly records that are classified as normal, and it is equivalent to those wrongly allowed. Formulas for the above mentioned metrics are given below:

Accuracy(AC):

$$AC = \frac{TP + TN}{TP + TN + FP + FN}$$

True Positive Rate (TPR):

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + TN}$$

4.8 SUMMARY

We went over the comprehensive design of our project in this chapter, discussing each step that went into executing it. This chapter gives a good idea of the design technique used and is broken into sections to make the procedure easier to understand.

Chapter 5

IMPLEMENTATION

The most crucial stage of the project development life cycle is implementation. This step entails converting the specified system requirements and specifications into a fully functional model that can be used to provide real-time services. The design stage's key functionality is built into functions that may be executed using the appropriate programming languages.

As a result, the implementation phase is always preceded by critical decisions such as platform selection, language selection, and so on. These selections are based on a variety of considerations, including the desired reaction time, security concerns, and data management concerns, among others. These choices have an impact on how well the finished system works.

5.1 PROGRAMMING LANGUAGE SELECTION

In the implementation phase, the programming language plays an important role. The requirements must be taken into account when choosing a programming language. Given the simplicity of the syntax and the ease with which apps can be constructed in python, it was an easy choice for our project. Python programming is ideal for machine learning applications since it has a large library of libraries that make duplicate work much easier. Python was used to build the backend of the project, which included the logic for the predictive tool. To deliver a user-friendly experience to patients or doctors, a front end is required. This was built with a flask framework using Google colab and its environment.

5.1.1 Overview of Python

Python is a multi-purpose, dynamic, object-oriented programming language. It comes with a large and diverse standard library. It allows users to express concepts in fewer lines of code, something that languages like C++ and Java cannot do. It can also be used to construct web applications. Because the project we're working on is mostly concerned with machine learning and data analysis, python is one of the greatest programming languages we might

utilise due to its dynamic nature.

Features of Python:

- Python is simple to learn because it has few keywords, clear syntax, and a straightforward structure.
- Because it can run on a variety of hardware systems, this language is portable.
- The Python code is simple to maintain.
- Python supports graphical user interfaces (GUIs).
- It has type checking that is dynamic.

5.1.2 Overview of Google Colab

Google Research's Colaboratory, or "Colab" for short, is a product. Colab is a web-based Python editor that allows anyone to write and run arbitrary Python code. It's notably useful for machine learning, data analysis, and education. Colab is a hosted Jupyter notebook service that doesn't require any setup and offers free access to computational resources, including GPUs.

Features of Google Colab:

- It allows you to execute and also write code in Python.
- It helps in documenting the code which supports the mathematical equations.
- Easy to create new notebooks.
- Open the existing notebooks by uploading them.
- With the google link notebooks can be shared easily.
- Data can be imported from Google Drive.
- Notebooks can be saved from/to Google Drive.
- Notebooks can be Imported/published from GitHub.

5.2 PLATFORM SELECTION

Windows 10 was used to create our project. This OS was chosen since our data required a lot of fine editing and visualisations, which MS Excel software could easily offer. Windows

provides a graphical user interface (GUI) for its PCs. By implementing a mouse that navigates among menus, tabs, dialogue boxes, and icons, Windows reduces the need for commands to control the OS. Because of its dynamic nature, which allows numerous jobs to execute at the same time, Windows was given this term. We chose this platform because it is user-friendly, and our project necessitated an emphasis on analytics rather than a robust development environment.

5.3 GRAPHICAL USER INTERFACE

Any project's success is determined by how well it satisfies the end consumers. Our project will primarily be used as a backend for other apps or platforms. As a result, we don't need a user interface with text boxes, buttons, labels, and so on. As a result, we use a Google Colab Notebook to demonstrate IDS. To make the results more thorough, users are given outputs in the form of data, tables, and graphs.

5.4 PYTHON LIBRARIES

Scikit-learn: Scikit-learn is a large Python package for machine learning, data cross-validation, and data preprocessing. NumPy and SciPy are used to create it.

Pandas: Pandas is a python package. It offers a well-designed data structure with quick expression and flexibility. It simplifies and intuitively handles data. Pandas is ideally suited for a variety of data types, including tabular, matrix, ordered, and unordered data.

NumPy: The fundamental object in NumPy is a multidimensional array, which contains an array data structure. NumPy is a Python core package that includes a number of tools and approaches. Multidimensional objects are one of these tools.

Matplotlib: Matplotlib is a data visualisation library that is widely used.

SciPy: SciPy is a library that includes routines for performing more sophisticated calculations, especially scientific calculations.

The developer may not be able to sustain the developed product for the rest of its life. As a result, it's critical to make the code understandable to every new reader. Making the code more understandable makes it easier to change and maintain in the future. Two further methods are used in addition to creating the code in python that is relatively straightforward to grasp.

Naming variables and constants: The naming is done in such a way that the name clearly reflects the variable's/evident constant's meaning. However, some of the dataset's features are network words, which may necessitate some additional research. Flags and services, for example, necessitate network expertise.

Comments: Appropriate comments are included in the code to help you follow the most difficult sections. To avoid any ambiguity, comments are given as complete sentences.

5.6 IMPLEMENTATION STEPS

1. Data Collection. We have chosen NSL-KDD datasets as our dataset to train and test the model. We have splitted the data dataset in 2:8 ratio for test to train purpose.
2. Data Prepossessing :Data features in the chosen dataset that represent input traffic of the networking system are naturally inconsistent. Thus, traffic data pre-treatment is a necessary gate for the classification engine.
3. Choose a Model. We have presented an ensemble model which consists of 10 different algorithms. They are:-
ANN Classifier , Keras Classifier, RNN Classifier, MLP Classifier, Logistic Regression, Naive Bayes Classifier, Decision Tree Classifier, Random Forest Classifier, KNeighborsClassifier, Support Vector Classifier
4. Training the model using the dataset.
5. Prediction of results and Evaluating the Model based on Accuracy , TPR and FPR.
6. Parameter Tuning to avoid the condition of overfitting and underfitting of the model.

5.7 SUMMARY

This chapter covered the many techniques used in the project's development, beginning with language and platform selection and ending with an explanation of the whole implementation process.

Chapter 6

TESTING

6.1 SOFTWARE TESTING

It's critical to guarantee the software's quality before putting it to use. Software testing is the process of examining a system and its separate components in order to detect flaws or problems. The primary goal of software testing is to determine whether or not the product meets the requirements and standards. When compared to the predicted requirements, it can be utilised to identify any gaps or missing needs.

Starting the testing process concurrently with the implementation is a good strategy to reduce the cost of fixing errors. Testing can begin after the specifications are clearly laid out in the software test cycle and continue until the product is deployed. Each phase of the life cycle can be tested with the goal of enhancing that phase. Testing, in addition to debugging, entails product verification and validation to ensure that both functional and non-functional criteria are met.

6.2 PURPOSE

The goal of this chapter is to make sure that the system meets all of the requirements, both at the unit and system levels.

6.3 LEVELS OF TESTING

We employed functional testing, which is a type of black box testing that is focused on the system's needs. At the unit and integration testing levels, the system is given inputs, and the outcomes are reviewed to confirm that they satisfy the functionality that was intended.

Unit testing: This type of testing is carried out on all of the system's primary components. Each component is separated from the rest of the system and tested for proper operation. The smallest bits of the software are tested for correctness. This technique is carried out during the system's development to avoid the system's failure as a whole after it is integrated. Unit

testing was performed on all important modules in our project, including data retrieval, preprocessing, classification, classifier assessment, and the screening tool mechanism.

Integration testing: Individual components may function well, but it's also crucial to figure out how they work together. This testing is done after all of the component modules have been merged to guarantee that the system as a whole operates without faults and fulfils its promises.

The subsections below go over each of these testing levels.

6.4 UNIT TESTING

Major modules are evaluated for their individual performance and correctness in this test.

Table 6.4.1 Unit testing for Data acquisition

Test case ID	Unit test case I
purpose	To check if csv file is loaded
input	<ul style="list-style-type: none">• CSV file containing data of network
procedure	<ul style="list-style-type: none">• Store the CSV file in the same folder as program <p>Import the CSV file in program using pandas</p>
expected result	<ul style="list-style-type: none">• should get all the data in 2D table
actual result	<ul style="list-style-type: none">• Dataset was imported successfully

	as expected
remark	PASS

Table 6.4.2 Unit testing for Data Preprocessing

Test case ID	Unit test case I
purpose	Preprocess the data
procedure	<ul style="list-style-type: none">• Take the raw data of CSV file• Check for null values• Convert categorical data to numerical data• Normalize the data with min-max normalization
input	<ul style="list-style-type: none">• Imported raw dataset
expected result	<ul style="list-style-type: none">• Preprocessed data without NULL, categorical data.
actual result	<ul style="list-style-type: none">• Preprocessed data without NULL, categorical data.
remark	PASS

Table 6.4.3 Units testing for classification

Test case ID	Unit test case I
purpose	To check if classifier is able to classify into expected classes
procedure	<ul style="list-style-type: none">• Divide the data into a Training and testing set.• Run classification algorithm
input	<ul style="list-style-type: none">• input the preprocessed data
expected result	<ul style="list-style-type: none">• algorithm classifies based on data• classes are attacked or not attacked
actual result	<ul style="list-style-type: none">• The predicted class of instance of all classifiers.• Algorithms classifies weather network is attacked or not
remark	PASS

Table 6.4.4 Unit testing for model evaluation

Test case ID	Unit test case I
purpose	Evaluate each model based on accuracy score
procedure	<ul style="list-style-type: none">• Compare the predicted results with the test dataset to get an accuracy score.
input	<ul style="list-style-type: none">• Testing data to predict values
expected result	<ul style="list-style-type: none">• correct accuracy score of each model
actual result	<ul style="list-style-type: none">• Accuracy score of each model
remark	PASS

6.5 INTEGRATION TESTING

We did integration testing once the unit testing was completed to ensure that all of the project's components were working together. When the system is working as a whole, this testing helps discover any faults or blunders. This assured that all of the modules, even as a group, worked properly. Integration testing was carried out in stages.

The modules were put together in order to see if the inputs and outputs of the combined modules were working properly. After the statistical analysis, the unstructured data with missing values was given to the preprocessing unit, where the outputs were validated for

missing values before the data was sent to the classification unit. At this point, the data was reviewed for the most important features, which were then ranked after the classification procedure. The Intrusion detection system was created by combining this with the prediction module and pooling the recommendations. The complete IDS's results were then put to the test in conjunction with the user interface. This ensured the overall process of receiving user inputs, operating the system, and returning the results to the user, which is a black box approach to the testing mechanism.

6.6 SUMMARY

This chapter describes the testing carried out at the module level as well as the testing of the entire project. The project appears to effectively meet all of the requirements. The various levels of testing have also been briefly explained.

Chapter 7

RESULTS

7.1 Outcomes

The intrusion detection system was built with the goal of detecting if the network has been attacked or not. We have compiled the results obtained for training model and model evaluation. IDS is successfully able to recognise all four types of attacks. The classification model is able to classify the networks and recognise detection with high accuracy. Different models have different accuracy based on how well a problem is suited to an algorithm. Out of all the models RNN has the highest accuracy of 98 percent. The accuracy scores of all the models are put in a table for comparison and the same is also plotted in a bar graph for easier understanding. At last confusion matrix gives us idea of how well the model has performed compared to testing dataset.

7.2 Measurements tables and charts:-

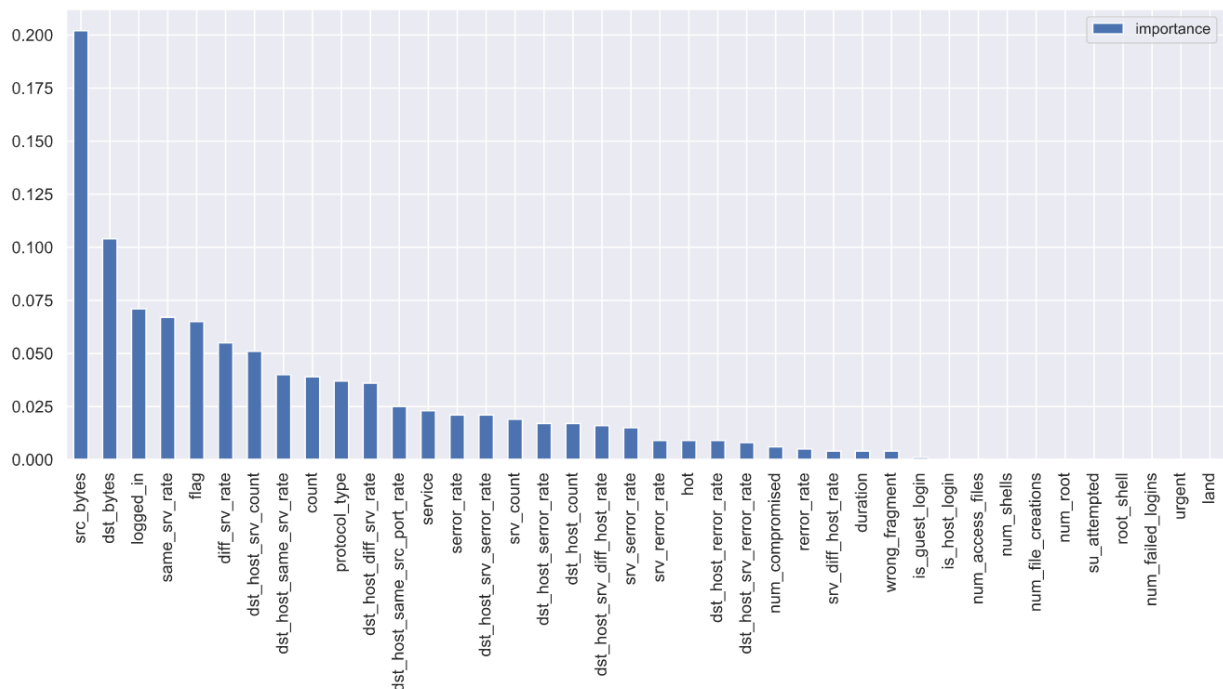


Fig: 7.1 Graph showing importance of each feature

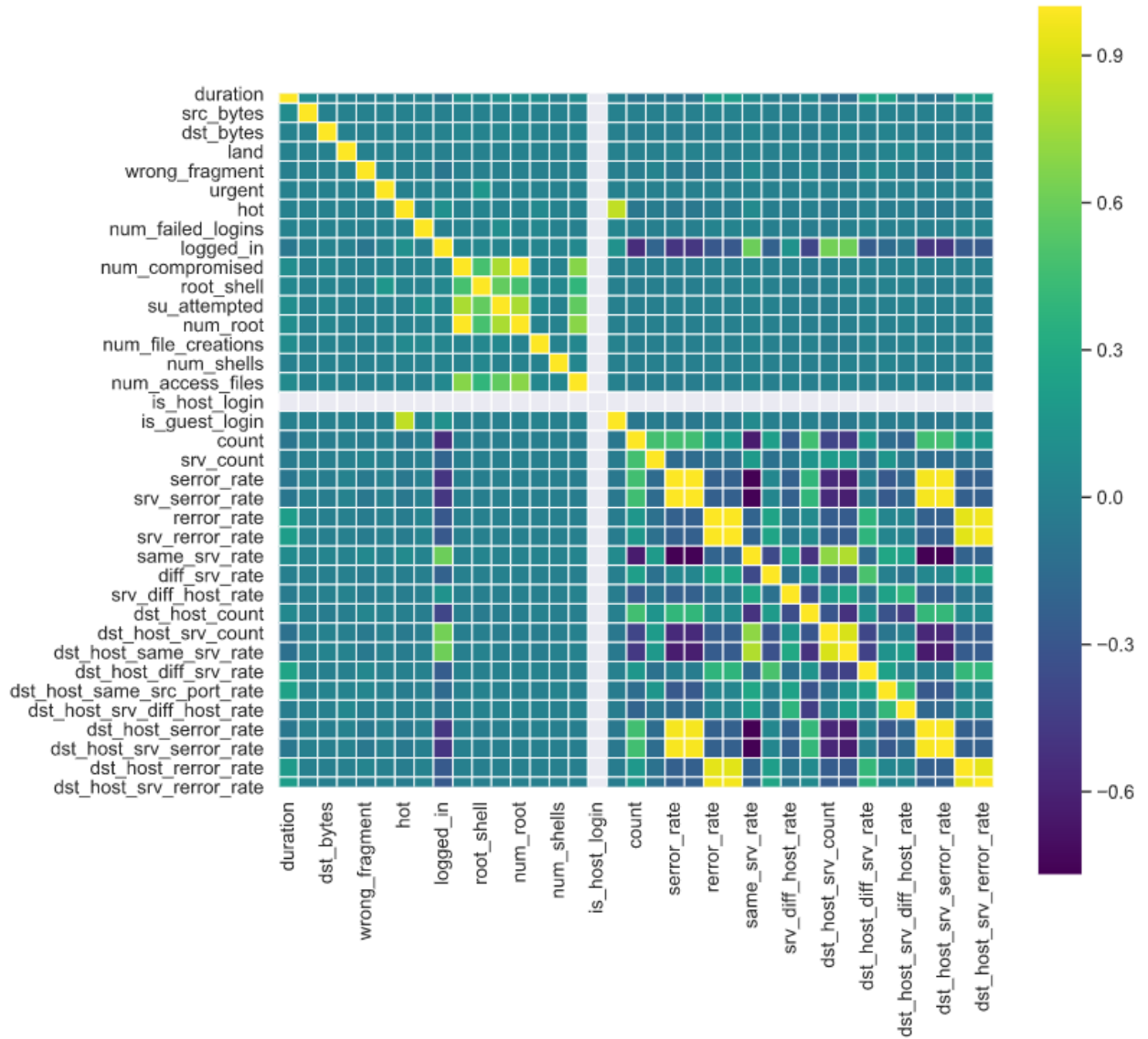


Fig: 7.2 Heatmap of Correlation matrix between features

Accuracy Table:-

Model	Accuracy	
ANN Classifier	0.978	
Keras Classifier	0.978	
RNN Classifier	0.988	
MLP Classifier	0.985	
Logistic Regression	0.945	
Naive Bayes Classifier	0.895	
Decision Tree Classifier	0.994	
Random Forest Classifier	0.998	
KNeighbors Classifier	0.986	
Support Vector Classifier	0.978	

Table 7.1: Accuracy table

Histogram Representation:-

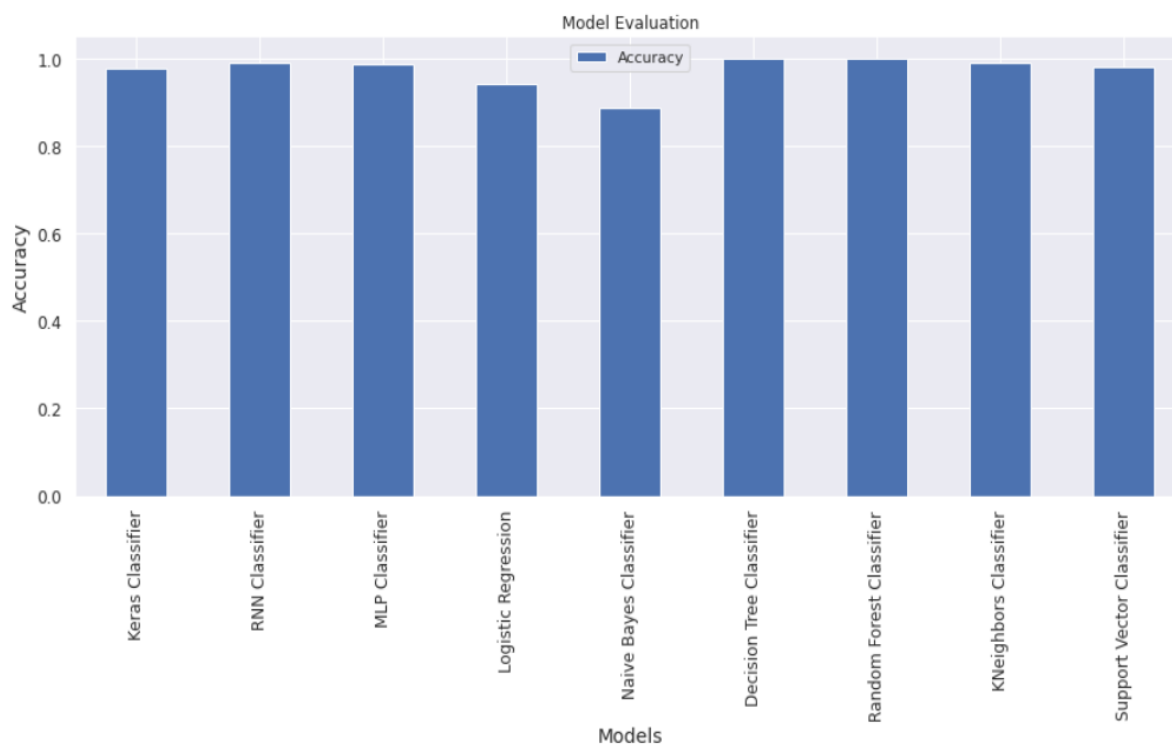


Fig 7.3: Model Evaluation

Confusion Matrix:-

<div> <div></div> <div>Predicted Class</div> </div> <div>Actual Class</div>	anomaly	normal
anomaly	9362	3471
normal	298	9413

Fig 7.4: Confusion matrix

Chapter 8

CONCLUSION

The RNN-IDS model offers high accuracy in binary and multiclass classification, as well as great modelling capacity for intrusion detection. When compared to traditional classification methods such as logistic regression, naive bayesian, and random forest, the performance achieves a higher accuracy rate and detection rate with a low false positive rate, particularly when performing multiclass classification on the NSL-KDD dataset. Both the accuracy of intrusion detection and the capacity to determine the type of intrusion can be improved with the model.

Of course, in future research, we'll continue to focus on reducing training time with GPU acceleration, avoiding exploding and vanishing gradients, and investigating the classification performance of the LSTM and Bidirectional RNNs algorithms in the field of intrusion detection.

8.1 LIMITATIONS OF THE SYSTEM

- The intrusion detection system has not been deployed, which means that while the technology has been successfully proven, the IDS has not been placed into production. This entails processes such as creating a graphical user interface (GUI) for software.
- This IDS was created using only one dataset, NSL-KDD, and subsequent versions.
- This project does not go into great detail on parameter tuning or model boosting to improve accuracy.

8.2 FUTURE ENHANCEMENTS

- When applications connected to the network are being targeted, intrusion detection functions best in the background. IDS should be capable of detecting attacks and alerting the user.
- Because intruders will continue to devise new ways to attack, intrusion detection systems must evolve to keep up.
- We will work on increasing accuracy so that it becomes more reliable and can be deployed.

REFERENCES

- DEEP RECURRENT NEURAL NETWORK FOR IOT INTRUSION DETECTION SYSTEM
MuderAlimiani, Alia AbuGhazleh, Amer Al-Rahayfeh, Saleh Atiewi, AbdulRazaque
<https://www.sciencedirect.com/science/article/pii/S1569190X19301625#!>
- DEEP LEARNING APPROACH FOR INTRUSION DETECTION SYSTEM (IDS) IN THE INTERNET OF THINGS (IOT) NETWORK USING RECURRENT NEURAL NETWORKS
Manoj Kumar Putchala(2017)
- A REVIEW OF INTRUSION DETECTION SYSTEMS USING MACHINE AND DEEP LEARNING IN INTERNET OF THINGS: CHALLENGES, SOLUTIONS AND FUTURE DIRECTIONS
Javed Asharf , Nour Moustafa, Hasnat Khurshid, Essam Debie, Waqas Haider and Abdul Wahab (2020)
<https://sciprofiles.com/profile/705539>
- A MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM FOR MOBILE INTERNET OF THINGS
Amar Amouri, Vishwa T. Alaparthi and Salvatore D. Morgera (2020)
- Online Editing tools:
<https://creately.com/>
<https://www.lucidchart.com/>
<https://www.draw.io/>
<https://research.google.com/colaboratory/>