

A Project Report

On

**Improving Intrusion Detection Systems With Machine Learning To
Strengthen Network Security**

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

P. Sridathu (20NN1A0541)

G. Triveni (20NN1A0513)

P. Divya Sai (20NN1A0539)

K. Madhavi (20NN1A0521)

Under the Esteemed Guidance of

Dr. V. Lakshman Narayana

Associate Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR
WOMEN**

PEDAPALAKALURU, GUNTUR-522005

(Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada.)

2020-2024

**VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR
WOMEN**

(Approved by AICTE, NEW DELHI and Affiliated to JNTUK,

Kakinada) PEDAPALAKALURU, GUNTUR-522005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that this project report entitled “**Improving Intrusion Detection Systems with Machine Learning to Strengthen Network Security**” is a bonafide record of work carried out by **Ganta Triveni (20NN1A0513)** under the guidance of and supervision of Dr. V Lakshman Narayana partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN, GUNTUR.

Project Guide
Dr. V. Lakshman Narayana
Associate Professor

Head of the Department
Dr. V. Lakshman Narayana

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the work described in this project work, entitled **“Improving Intrusion Detection Systems With Machine Learning To Strengthen Network Security”** which is submitted by us in partial fulfilment for the award of **Bachelor of Technology** in the Department of **Computer Science and Engineering** to the **Vignan’s Nirula Institute of Technology and Science for women**, affiliated to Jawaharlal Nehru Technological University Kakinada, Andhra Pradesh, is the result of work done by us under the guidance of **Dr. V. Lakshman Narayana**, Associate Professor.

The work is original and has not been submitted for any Degree/ Diploma of this or any other university.

Place:

Date:

P. Sridathu (20NN1A0541)

G. Triveni (20NN1A0513)

P. Divya Sai (20NN1A0539)

K. Madhavi (20NN1A0521)

ACKNOWLEDGEMENT

We express our heartfelt gratitude to our beloved principal **Dr. P. Radhika** for giving a chance to study in our esteemed institution and providing us all the required resources.

We would like to thank **Dr. V. Lakshman Narayana, Associate Professor, Head of the Department of Computer Science and Engineering**, for his extended and continuous support, valuable guidance and timely advices in the completion of this project thesis.

We wish to express our profound sense of sincere gratitude to our Project Guide **Dr. V. Lakshman Narayana, Associate Professor, Department of Computer Science and Engineering**, without whose help, guidance and motivation this project thesis could not have been completed the project successfully.

We also thank all the faculty of the Department of Computer Science and Engineering for their help and guidance of numerous occasions, which has given us the cogency to build-up adamant aspiration over the completion of our project thesis.

Finally, we thank one and all who directly or indirectly helped us to complete our project thesis successfully.

PROJECT ASSOCIATES

P. Sridathu (20NN1A0541)

G. Triveni (20NN1A0513)

P. Divya Sai (20NN1A0539)

K. Madhavi (20NN1A0521)

**IMPROVING INTRUSION DETECTION
SYSTEMS WITH MACHINE LEARNING TO
STRENGTHEN NETWORK SECURITY**

ABSTRACT

Machine Learning which is one of the most prominent applications of Artificial Intelligence, is doing wonders in the research field of study. Machine Learning is a rapidly growing field that has found its application in various industries across the world. Security measures to prevent malicious activity and unauthorized access to computer networks are mostly dependent on IDS. This work suggests a hybrid IDS strategy that combines the XGBoost and Random Forest algorithms. Because of its scalability and resilience, Random Forest can effectively handle big, highly dimensional datasets. By improving the detection accuracy of the IDS, XGBoost, which is renowned for its effectiveness in managing sparse data and understanding intricate patterns, enhances Random Forest. By combining the advantages of both methods, the hybrid model seeks to increase detection rates while lowering false positive rates. The outcomes of the experiments show how well the suggested method works to improve IDS performance, underscoring its potential to protect computer networks from a variety of online threats.

TABLE OF CONTENTS

Chapter	Content	PageNo
Chapter 1 Introduction		1
Chapter 2 Literature Survey		3
Chapter 3 System Analysis		9
3.1 System Analysis		9
3.2 Existing System		10
3.3 Proposed System		11
3.4 Feasibility Study		11
Chapter 4 Software Requirements Specifications		12
4.1 Purpose,Scope,definition		12
4.2 Requirement Analysis		12
4.3 System Requirements		14
Chapter 5 System Design		15
5.1 System Architecture		15
5.2 Flow Chart		16
5.3 Design Overview		17
5.4 UML Diagrams		18
5.4.1 Use case Diagram		20
5.4.2 Activity Diagram		21
5.4.3 Sequence Diagram		22
5.4.4 Component Diagram		23
5.5 Modules		25
5.5.1 Front-End Module		25
5.5.2 Connectivity Module(Flask)		25
5.5.3 Back-End Module		26
Chapter 6 Implementation		28
6.1 Steps for Implementation		28
6.2 Backend Code		39
6.3 Front end Code		44
Chapter 7 System Testing		48
7.1 Testing		48

7.2 Types Of Tests	48
7.2.1 Unit Testing	49
7.2.2 Integration Testing	50
7.2.3 Functional Testing	50
7.2.4 System Testing	51
7.2.5 White Box Testing	51
7.2.6 Black Box Testing	51
7.3 Test Cases	51
Chapter 8 Results	53
8.1 Results	53
8.2 Accuracy	55
Chapter 9 Conclusion And FutureWork	56
9.1 Conclusion And FutureWork	56
Bibliography	57
Certification	59

LIST OF FIGURES

Figure No	Description of Figures	Page No
5.1	Machine Learning Model	15
5.2	Flow Chart	16
5.3	Types and Categories of UML Diagrams	17
5.4.1	Use Case Diagram	20
5.4.2	Activity Diagram	21
5.4.3	Sequence Diagram	22
5.4.4	Component Diagram	23
5.4.4.1	Component Diagram-2	24
5.4.4.2	Component Diagram-3	24
5.4.5	Deployment Diagram	25
7.3	Alphabetic Misinterpretation	52
7.3.1	Threshold Value Failure	52
8.1.1	Input Representation for Normal Detection	53
8.1.2	Output Representation for Normal Detection	53
8.1.3	Input Representation for Anamoly Detection	54
8.1.4	Output Representation for Anamoly Detection	54
8.1.5	Comparison of Actual and predicted Labels	55

LIST OF TABLES

Table No	Table Description	Page No
2.1	Literature Survey	5
8.2	Accuracy	55

LIST OF ACRONYMS

IDS	Intrusion Detection System
ML	Machine Learning
OS	Operating Systems
UML	Unified Modelling Language
RF	Random Forest
GB	Gradient Boosting
XGB	Extreme Gradient Boosting

CHAPTER 1

INTRODUCTION

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

The security of computer networks is critical in today's globally interconnected world. Traditional security measures by themselves are no longer adequate to fend off intrusions and harmful activity due to the growing sophistication of cyber threats[1]. By keeping an eye out for unusual activity in system logs and network traffic, intrusion detection systems, or IDSs, are essential in spotting and eliminating these threats. Traditional IDSs struggle to keep up with new attack strategies and zero-day vulnerabilities because they rely on predetermined rules or signatures to identify known threats[2]. This constraint has prompted research into machine learning (ML) methods as a way to improve IDS performance. Using ML, IDS can go beyond strict adherence to rules by using historical data to identify anomalies that may be signs of an impending intrusion[3].

The life cycle of the data that is currently available thanks to new technologies involves numerous stages, such as generation, transfer, storage, and deletion[4]. At any point in the data cycle, the portable information is quite valuable, particularly when it comes to financial transactions, governments, or the armed forces. As such, information security and data privacy were crucial concerns for minimizing damages brought about by ignoring them. Hackers attempt to steal, alter, or destroy data due to system weaknesses, frequently causing harm to the systems themselves[5]. The purpose of this study is to use machine learning algorithms to give a thorough investigation of intrusion detection systems. It will go over the difficulties faced by conventional IDSs, the advantages of using ML algorithms, and the many ML techniques that are frequently applied to IDSs[6].

The introduction lays the groundwork for comprehending the role that ML-based IDSs play in contemporary cybersecurity procedures. It emphasizes how effective defense against changing cyber threats requires flexible and intelligent security solutions[7]. Therefore, the goal of this research is to add to the body of knowledge about machine learning-based IDSs and how they protect computer networks against intrusion attempts. The proposed architecture is random forest known for its efficiency in managing huge datasets with high dimensionality, the Random Forest method is used in the suggested Intrusion Detection System (IDS) approach. Random Forest is an effective method for identifying both

legitimate and invasive network activity, offering a reliable way to find anomalies. Nevertheless, Random Forest has many shortcomings in spite of its advantages[8]. One of its primary disadvantages is that it tends to overfit when the forest has an excessive number of trees, which might result in poorer generalization performance on data that hasn't been seen before[9]. Furthermore, because Random Forest can bias the model in favor of the majority class, it may not perform well on datasets that are severely imbalanced and have one class with a considerable population advantage over the other[10]. Using this, the output is accurate. But not faster. So, for this, we used a hybrid approach which consists of random forest and xgboost algorithms to predict IDS.

CHAPTER 2
LITERATURE SURVEY

CHAPTER-2

LITERATURE SURVEY

Yuansheng Dong, Rong Wang and Juan He et al. [1] proposed a real-time network intrusion detection system based on deep learning, which through the system Flume collects log information and network information, uses Flink to perform real-time cleaning and feature extraction on the original data, and then transmits the extracted high-order features to the neural network for training and judgment.

Nadeem et al. [2] proposed a neural network with semi-supervised learning, and uses a small number of labeled samples to obtain high accuracy by using small amount of data advantage is high efficiency when compared with deep learning technique.

Staudemeyer et al. [3] proposed a intrusion detection based on LSTM regression neural network. The results show that the LSTM classifier has certain advantages over other strong static classifiers. These advantages lie in detecting DoS and Rube attacks, both of which can produce unique time series characteristics. In order to compensate for the high false alarm rate.

Kim et al. [4] proposed a call language modeling method to improve the host intrusion detection system based on LSTM. By integrating call language modeling, the system can better analyze and understand the sequence of system calls, thereby enabling it to detect anomalous behavior indicative of potential intrusions with higher precision and efficiency.

Agarap et al. [5] proposed a softmax by introducing linear support vector machine (SVM) into the final output layer of GRU model, and applied the model to the second classification of intrusion detection.

Kolosnjai et al. [6] proposed a neural network consisting of convolution and feedforward neural structures. This paper presents a hierarchical feature extraction method, which vectorizes n-gram instruction features and convolution features.

Shun. Jimmy et al. [7] proposed a network intrusion detection system using neural networks anomaly detection methods based on neural network aims to enhance network security through the implementation of neural network-based anomaly detection methods within a network intrusion detection system.

Chung-Ming Ou et al. [8] proposed a artificial immune systems (AIS) and intrusion detection systems are similar in many ways. We may apply mechanisms inspired by AIS to improve the intelligence of IDS. Researches on AIS focus on the development of specialized algorithms inspired by theories such as the negative selection theory or the danger theory.

Rafath Samrin et al. [9] proposed the various Anomaly based IDS technique concluded that single technique is not able to provide accurate detection rate. For boosting the anomaly detection method, an efficient automatic hybrid technique is suggested to achieve accurate detection rate. Also, helps to reduce the false prediction rate as well as decrease the time complexity. Additionally, machine learning techniques reduce the network traffic.

Mukherjee et al. [10] proposed a naive Bayes classifier coupled with feature reduction techniques, such as Principal Component Analysis (PCA) or Information Gain, to enhance the efficiency and effectiveness of intrusion detection systems by reducing dimensionality and improving classification accuracy.

Table 2.1. Literature Study

S.No	Title	Author	Year published	Algorithms used	Advantages	Accuracy	Limitations
1.	Real-Time Network Intrusion Detection System Based on Deep Learning	Yuansheng Dong, Rong Wang and Juan He	2019	Deep Learning Techniques such as hierarchical abstraction.	The learning of neural networks has obvious advantages in identifying threats.	The experimental results of the intrusion detection dataset KDD 99 show that the accuracy of the AE-AlexNet model is as high as 94.32%.	There are some shortcomings in the actual test, mainly in the long training period and poor portability.
2.	Automatic salt deposits segmentation: A deep learning approach	Nadeth, Muthur,	2016	Semi-supervised deep neural network for network intrusion detection.	Efficiency is very high	The efficiency of this method is high because of small amount of data 96%	high false alarm rate.
3.	Applying long short-term memory recurrent neural networks to intrusion detection	Staudemeyer, Ralf C.	2015	LSTM regression neural network.	advantages lie in detecting DoS and RDoS attacks.	compensate for the high false alarm rate.	LSTM regression is time taken process for analysis.

4.	LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems	Kim, Gyuwan et al.	2016	A call language modeling method to improve the host intrusion detection system based on LSTM.	Its advantage lies in its ability to capture sequential patterns using LSTM networks, enhancing the detection accuracy of the host intrusion detection system.	Improve the host intrusion detection system	One limitation is its reliance on labeled training data, which can be challenging to obtain for real-world intrusion scenarios and may introduce biases or inaccuracies.
5.	A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data	Agarwal, Abhinav, and Fred M.	2018	Linear support vector machine (SVM) into the final output layer of GRU model, and applied the model to the second classification of intrusion detection.	More advanced Technique for intrusion detection.	By combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection reduce network traffic.	If the number of Techniques used, the computational time also increases.

6.	Deep learning for classification of malware system call sequences.	Kolosnjaji . Bojan, et al.	2016	This paper presents a deep learning hierarchical feature extraction method.	which vectorizesn-gram instruction features and convolution features.	Achieves high classification accuracy by leveraging deep learning techniques to analyze malware system call sequences.	Requires large amounts of labeled data and computational resources for training, which can be challenging to obtain and costly to implement in practice.
7.	Network intrusion detection system using neural networks.	Shun. Jimmy, and Heidar A. Malki.	2008	Utilizes neural network-based anomaly detection methods, such as self-organizing maps (SOMs).	Enhancing the system's ability to identify sophisticated cyber threats.	Demonstrates improved detection rates through the utilization of neural network-based anomaly detection methods .	Susceptible to false positives and requires substantial computational resources for real-time monitoring due to the complexity of neural networks

8.	Host-based Intrusion Detection Systems inspired by Machine Learning of Agent-based Artificial Immune Systems	Chung - Ming Ou	2019	Adaptable Agent-based IDS (AAIDS)	existence of both danger and PAMP signals show the learning mechanism of AAIDS rely on host computers which correctly classify system calls	Achieves effective intrusion detection by leveraging adaptable agent-based mechanisms inspired by artificial immune systems and danger theory.	complex.
9.	Review on Anomaly based Network Intrusion Detection System	Rafath Samrin and, D Vasumathi	2017	For boosting the anomaly detection method, an efficient automatic hybrid technique is suggested to achieve accurate detection rate	Helps to reduce the false prediction rate as well as decrease the time complexity. Additionally, machine learning techniques reduce the network traffic.	Provides a comprehensive overview of intrusion detection systems, advocating for hybrid techniques to enhance detection accuracy	Require training data from many different datasets

10.	Intrusion detection using naive Bayes classifier with feature reduction,	S.Mukherjee and N.Sharma	2012	Naïve Bayes Classification(NBC) and Feature Vitality Based Reduction Method (FVBRM)	FVBRM method recognize the significant input features. After that apply NBC on decrease the datasets for intrusion detection.	This feature reduction method improves the classification accuracy.	The training and testing speed is slow
-----	--	--------------------------	------	---	---	---	--

CHAPTER 3

SYSTEM ANALYSIS

CHAPTER-3

SYSTEM ANALYSIS

3.1 SYSTEM ANALYSIS:

The intrusion detection system analysis that employs Random Forest and XGBoost comprises a thorough investigation of its design, training procedure, performance metrics, and possible improvements. XGBoost, a gradient boosting technique, and Random Forest, renowned for its ensemble learning methodology, are used to efficiently categorize network traffic as malicious or benign. While XGBoost uses gradient boosting to improve decision tree performance, Random Forest combines forecasts from several decision trees. During the training phase, methods such as grid search and cross-validation are used to optimize the models' hyperparameters in order to minimize a loss function and raise prediction accuracy. Accuracy measurements are used in performance evaluation to gauge how successful intrusion detection is. While XGBoost and Random Forest provide benefits in ensemble learning and classification, respectively, feature selection, class imbalance, and model interpretability remain obstacles. Future steps can include incorporating sophisticated feature selection strategies, using sampling techniques to solve class imbalance, and improving the interpretability of the model for increased reliability and understanding.

Problem Definition

The topic of automatic intrusion detection, which aims to categorize network traffic as benign or malicious based on numerous network parameters, is the focus of this study. The system uses XGBoost and Random Forest for classification in order to identify and stop possible cyberthreats in network environments, like intrusions or illegal access. Managing high-dimensional feature spaces, responding to changing attack tactics, and guaranteeing the effectiveness and scalability of detection techniques are major obstacles. The goal is to create a strong intrusion detection system that can quickly and effectively detect and neutralize a variety of online threats. The study aims to improve network security and defend vital information infrastructures from cyberattacks by tackling these issues.

3.2 EXISTING SYSTEM:

A viable method of real-time deep learning-based network intrusion detection is presented by the current system that is the subject of the research. The system efficiently handles both log and network information by utilizing technologies such as Flume for data collecting and Flink for real-time cleaning and feature extraction. Nevertheless, a number of shortcomings limit its usefulness and efficacy despite its improvements. The lengthy training period that the neural network needs to undergo is one major restriction. Long training periods can make it more difficult for the system to recognize and react to threats in real-time intrusion detection settings, where speed is of the essence. Transfer learning is one technique that could be investigated to break through this bottleneck and speed up the training process without sacrificing accuracy.

Furthermore, the system's weak portability limits its application to other network topologies and contexts. Improving portability entails making sure that the architecture and software dependencies are optimized for different platforms. The accessibility and adaptability of the system could be enhanced by containerization technologies like Docker or frameworks like TensorFlow Serving, which could enable smooth deployment across various environments. Enhancing the efficiency of feature extraction is an additional area of focus. Improving the efficiency of the pipeline could lower latency and resource consumption, even while Flink handles feature extraction and cleaning in real-time. Feature extraction for large-scale networks can be efficiently scaled using methods like data parallelism or distributed processing, which will improve the system's overall performance and responsiveness.

The fundamental components of intrusion detection systems are security and scalability. It is essential to have robustness against attacks and horizontal scaling. Decision-making is aided by improved interpretability. These improvements provide effective, trustworthy threat identification in dynamic networks. Enhancing trust and streamlining security analysts' decision-making also depends on how well the deep learning models interpret and explain.

Model interpretability techniques can shed light on the reasoning behind the system's choices, empowering analysts to successfully validate and comprehend its outputs. A more effective, scalable, and reliable intrusion detection system must be developed by addressing

these shortcomings and potential areas for development. Through the removal of obstacles to training, increased efficiency and portability, increased security and scalability, and improved interpretability, the system may provide prompt and trustworthy threat detection in dynamic network environments. The existing system consists the 92% accuracy according to the research.

3.3 PROPOSED SYSTEM

To improve network security, a hybrid approach called Random Forest and XGBoost is suggested for the IDS. The robust and scalable Random Forest algorithm is used to efficiently categorize both normal and intrusive network activity. By increasing the IDS's detection accuracy, XGBoost, which is well-known for its effectiveness in managing sparse data and understanding intricate patterns, enhances Random Forest. The IDS seeks to improve network security by combining both algorithms into a hybrid model that will increase detection rate while preserving low false positive rate. The hybrid approach is a complete and user-friendly solution because it incorporates front-end connection with Flask, enabling users to communicate with the IDS and view results in real-time. The proposed model gives the accuracy of 99% which will be the best fit for IDS prediction.

3.4 FEASIBILITY STUDY:

A Feasibility Study is a preliminary study undertaken before the real work of a project starts to ascertain the likely hood of the project's success. It is an analysis of possible alternative solutions to a problem and a recommendation on the best alternative.

- 3.3.1 Economic Feasibility
- 3.3.2 Technical Feasibility
- 3.3.3 Operational Feasibility

3.3.1. Economic Feasibility:

It is defined as the process of assessing the benefits and costs associated with the development of project. A proposed system, which is both operationally and technically feasible, must be a good investment for the organization. With the proposed system the users are greatly benefited as the users are able to detect water quality that is useful for salmon fish basing on the physical, chemical and biological parameters of water. This proposed system does not need any additional software and high system configuration.

3.3.2 Technical Feasibility

The technical feasibility infers whether the proposed system can be developed considering the technical issues like availability of the necessary technology, technical capacity, adequate response and extensibility. The project is decided to build using Python.Jupyter Note Book is designed for use in distributed environment of the internet and for the professional programmer it is easy to learn and use effectively. As the developing organization has all the resources available to build the system therefore the proposed system is technically feasible.

3.3.3 Operational Feasibility:

Operational feasibility is defined as the process of assessing the degree to which a proposed system solves business problems or takes advantage of business opportunities. The system is self-explanatory and doesn't need any extra sophisticated training. The system has built in methods and classes which are required to produce the result. The overall time that a user needs to get trained is less than one hour. As the software that is used for developing this application is very economical and is readily available in the market. Therefore, the proposed system is operationally feasible.

CHAPTER 4

REQUIREMENTS SPECIFICATION

CHAPTER-4

REQUIREMENTS SPECIFICATION

4.1 PURPOSE, SCOPE, DEFINITION:

4.1.1 Purpose:

The purpose of the software requirements Specification is the basis for the entire project .It lays the foundation and also framework that every team involved in the development will follow .It is used to provide the critical information to multiple teams like development, quality assurance , operations ,and maintenance . Software requirements specification is a rigorous assessment of requirements before the more specific system designs and its goal is to reduce later the redesign. It should also provide the realistic basis for estimating product costs, risks and also schedules.

4.1.2 Scope:

The scope is the part of project planning that involves determining and documenting a list of specific project goals, deliveries, features, functions, tasks, deadlines, and ultimately costs. In other words, it is what needs to be achieved and the work that must be done to deliver a project .The Software scope is well defined boundary which encompasses all the activities that are done to develop and deliver the software product. The software scope clearly defines the all functionalities and artifacts to be delivered as a part of the software.

4.1.3 Definition:

The Software Requirement Specification is a description of a software system to be developed. It is modeled of a software system to be developed. It is modeled after business requirements specification, also known as the stake holder requirements specification.

4.2 REQUIREMENT ANALYSIS

The process to gather the software requirements from clients, analyze and document them is known as requirements engineering or requirements analysis. The goal of requirement engineering is to develop and maintain sophisticated and descriptive ‘System/Software Requirements Specification’ documents.

It is a four step process generally, which includes—

- Feasibility Study
- Requirements Gathering
- Software Requirements Specification
- Software Requirements Validation

The basic requirements of our project are:

- Research Papers
- Camera

4.2.1 FUNCTIONAL REQUIREMENT ANALYSIS

Functional requirements explain what has to be done by identifying the necessary task, action or activity that must be accomplished. Functional requirements analysis will be used as the top –level functions for functional analysis.

4.2.2 USER REQUIREMENTS ANALYSIS

User Requirements Analysis is the process of determining user expectations for a new or modified product. These features must be Quantifiable, relevant and detailed.

4.2.3 NONFUNCTIONAL REQUIREMENT ANALYSIS

Non-functional requirements describe the general characteristics of a system .They are also known as quality attributes. Some typical non-functional requirements are Performance, Response Time, Throughput, Utilization, and Scalability.

Performance:

The performance of a device is essentially estimated in terms of efficiency, effectiveness and speed.

- Short response time for a given piece of work.
- High throughput (rate of processing work)
- Short data transmission time.

Response Time:

Response is the time a system or functional unit takes to react to a given input.

4.3 SYSTEM REQUIREMENTS

4.3.1 Software Requirements

Operating System: Windows 7 or 10 Ultimate, Linux, Mac.

Coding Language: Python

Software Environment: Python IDLE

Packages: numpy,tensorflow,keras,pandas,matplotlib

4.3.2 Hardware Requirements

Processor: Intel i3, i5, i7

RAM:4GB or Higher

Hard Disk: 500GB or Higher

CHAPTER 5

SYSTEM DESIGN

CHAPTER-5 SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

An IDS using machine learning architecture usually consists of multiple phases. During this phase, information is gathered from a variety of sources, including system logs, network traffic, and other pertinent sources. Preprocessing is done on the gathered data to clean, standardize, and format it appropriately for analysis. Data normalization, dimensionality reduction, and feature selection might be involved in this. Preprocessed data is used to extract pertinent features. These characteristics could be system call sequences, network packet headers, or other elements useful for identifying intrusions. Labeled or unlabeled data is used to train machine learning models, such as supervised learning algorithms (e.g., random forest, xg boost) or unsupervised learning algorithms (e.g., clustering, anomaly detection), to identify patterns of typical and malevolent actions. Performance indicators including accuracy, precision, recall, and F1-score are used to evaluate the trained models in order to determine how well they identify incursions. A model can be used to monitor network traffic or system activity for possible intrusions in real-time after it has been trained and assessed.

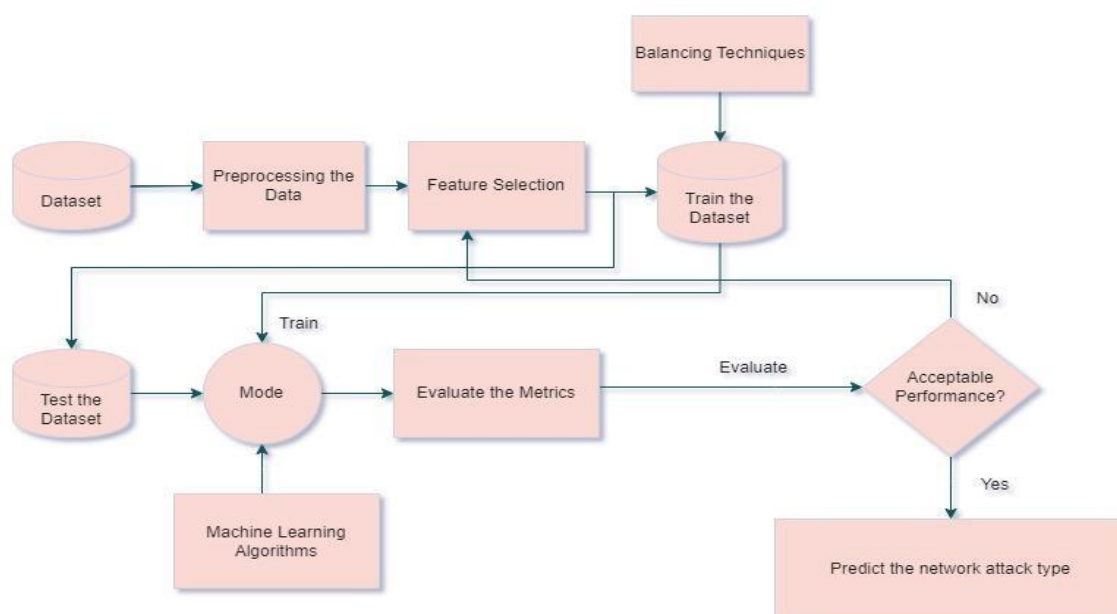


Figure 5.1. Machine Learning Model

5.2 FLOW CHART:

The procedure of an IDS that uses machine learning is described in the flowchart. The first step is to continuously monitor system activity or network traffic. Various sources of data are gathered, and before analysis, preprocessing is done. The processed data is converted into an intrusion dataset that contains annotated instances of both harmful and normal behavior. Different activities are characterized by extracting relevant features. The intrusion dataset is then used to build a machine learning model that will identify intrusions. The trained model is then used to anticipate possible attacks.

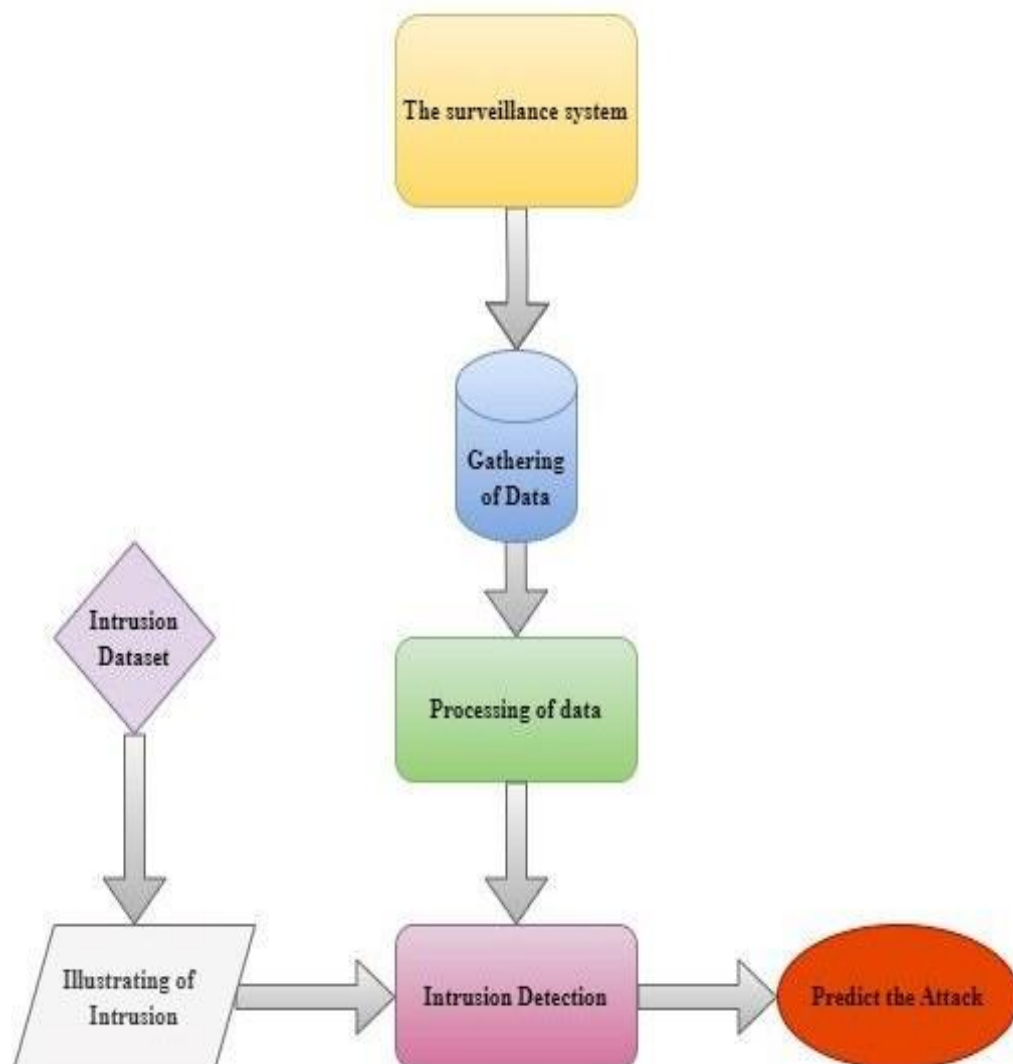


Figure 5.2. Flow Chart

5.3 DESIGN OVERVIEW

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object modeling technique (OMT) and object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

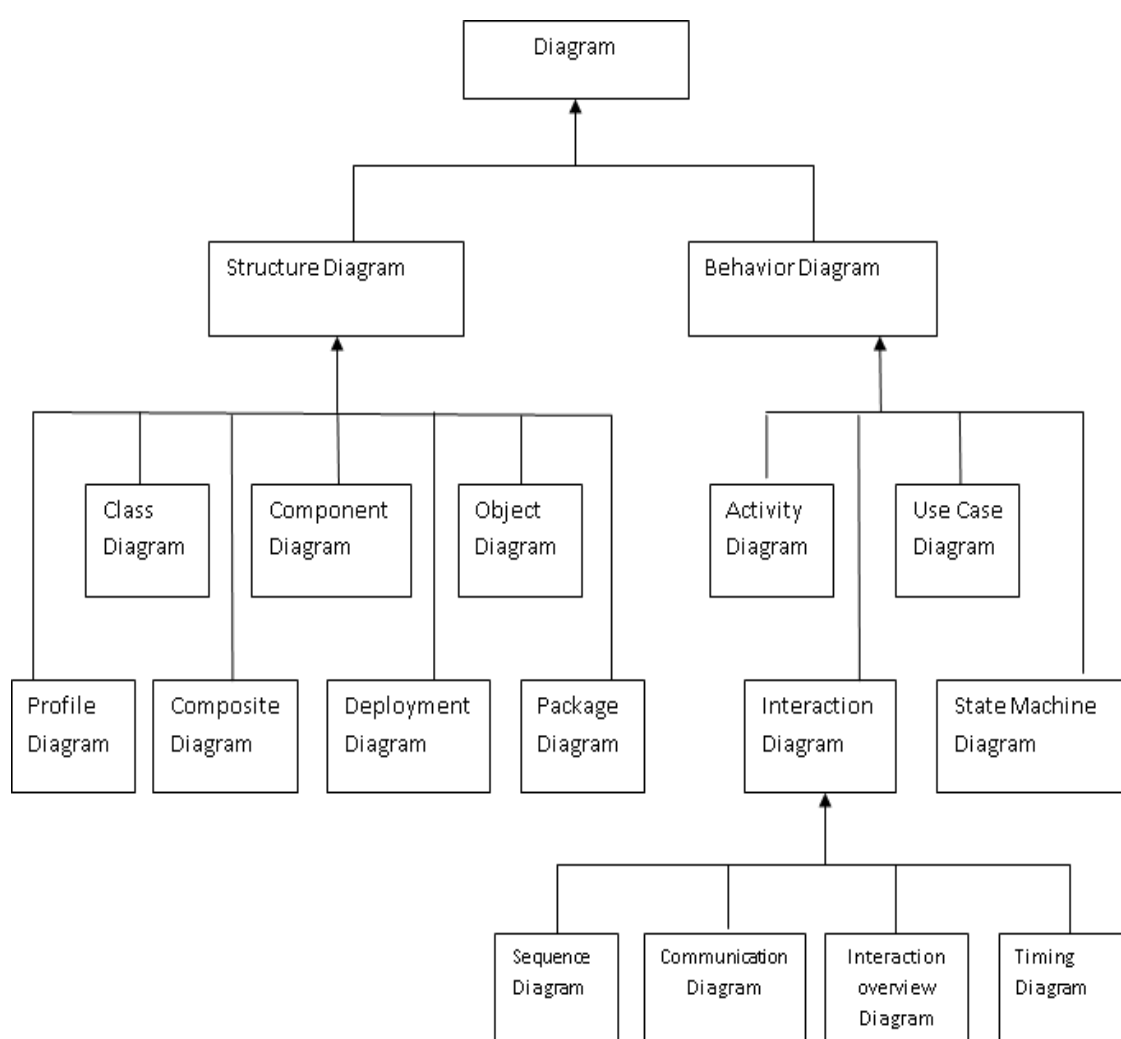


Figure 5.3. Types and categories of UML diagrams

5.4 UML Diagrams

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blue prints, including elements such as:

- Actors
- Business process
- (Logical) Components
- Activities
- Programming language statements
- Database schemas, and
- Reusable software components.

The unified modelling language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using 5 different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows:

User Model View:

- This view represents the system from the user's perspective.
- The analysis representation describes a usage scenario from the end-user's perspective.
- The UML user model view encompasses the models which define a solution to a problem as understood by the client stakeholders.

Structural Model View:

- In this model the and functionality are arrived from inside the system.
- This model view models the static structures.

Behavioral Model View:

- It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

Implementation Model View:

- Implementation view is also known as Architectural view which typically captures the enumeration of all the subsystems in the implementation model, the component diagrams illustrating how subsystems are organized in layers and hierarchies and illustrations of import dependencies between subsystems.

Environmental Model View:

- These UML model describe both structural and behavioural dimensions of the domain or environment in which the solution is implemented. This view is often also referred to as the deployment or physical view.

5.4.1 Use case Diagram:

A flow of events is a sequence of transaction performed by the system. They typically contain very detailed information, written in terms of what the system should do not how the system accomplishes the task flow of events are created as separate files or documents in your favourite text editor and then attached or linked to use case using the files or documents in your favourite text editor and then attached or linked to a use case using the files tab of a model element.

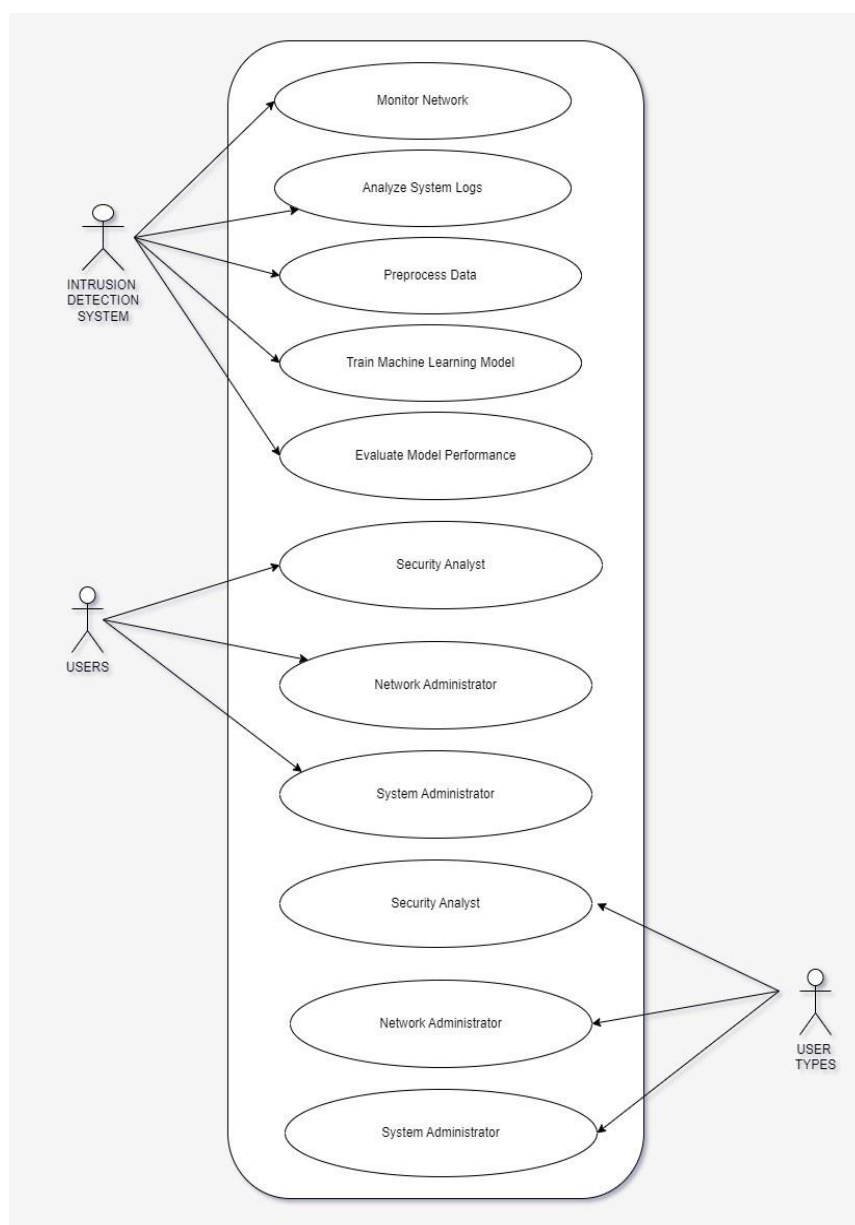


Figure 5.4.1. Use case Diagram

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some systems should or can perform in collaboration with one or more external users of the system(actors).

5.4.2 Activity Diagram:

Activity Diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

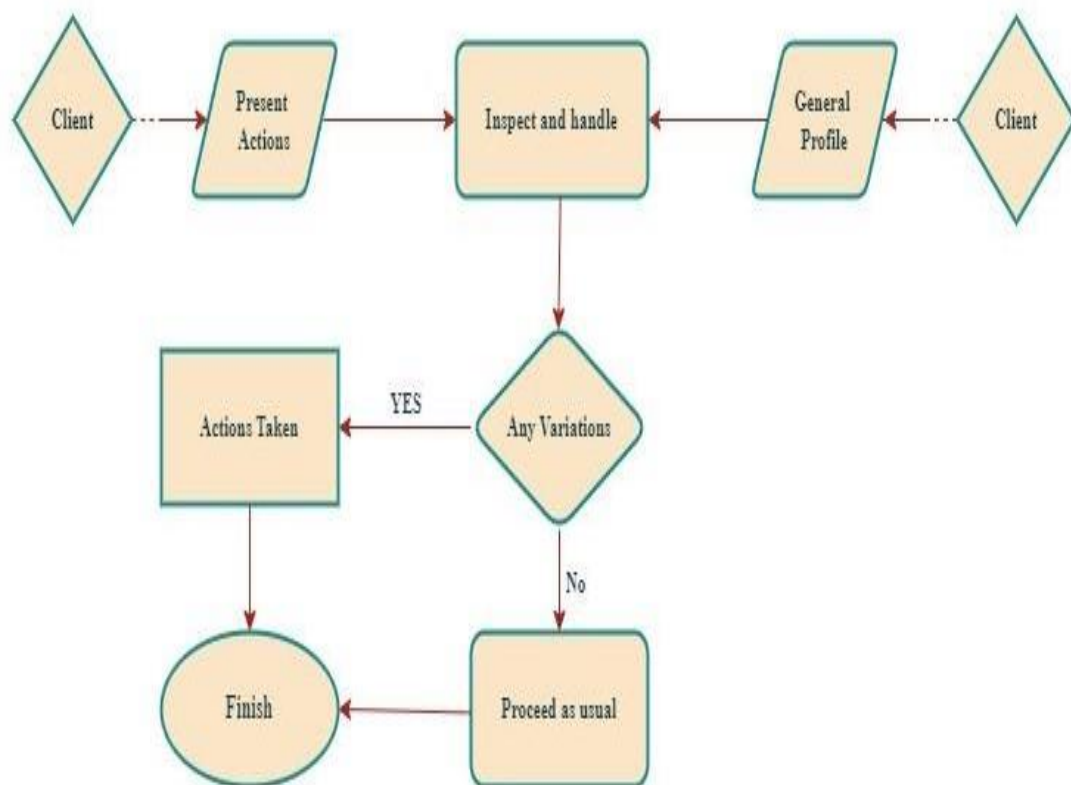


Figure 5.4.2. Activity Diagram

5.4.3 Sequence Diagram:

A sequence diagram in UML is a kind of interaction diagram that shows how processes operates with one another and in what order. It is a construct of Message Sequence Chart. A sequence diagram shows, as parallel vertical lines(—lifelines‖), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in graphical manner.

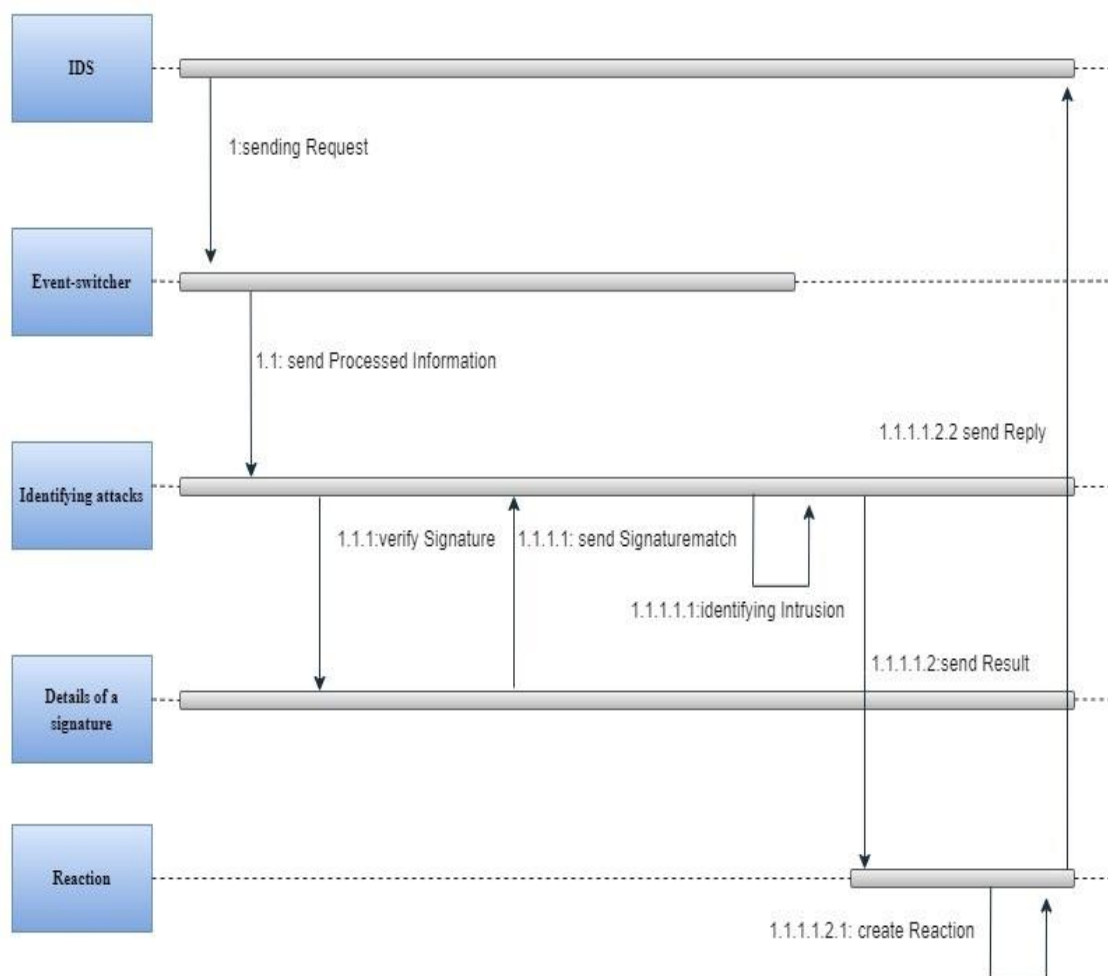


Figure 5.4.3. Sequence Diagram

5.4.4 Component Diagram:

Component diagrams are used to display various components of a software system as well as subsystems of a single system. They are used to represent physical things or components of a system. It generally visualizes the structure and an organization of a system.

An Intrusion Detection System (IDS) has two main phases:

1. **Learning Phase:** The IDS learns normal behavior patterns from collected data.
2. **Detection Phase:** It detects anomalies or intrusions based on deviations from the learned normal behavior.

There are three component diagrams for intrusion detection system. They are:-

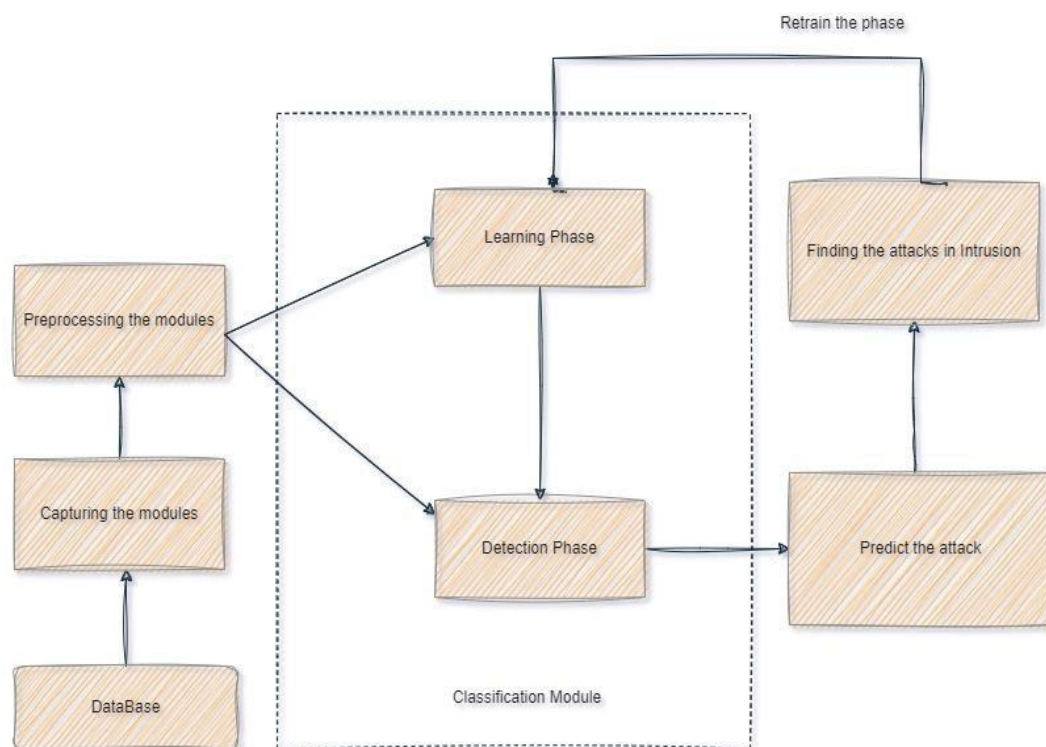


Figure 5.4.4. Component Diagram

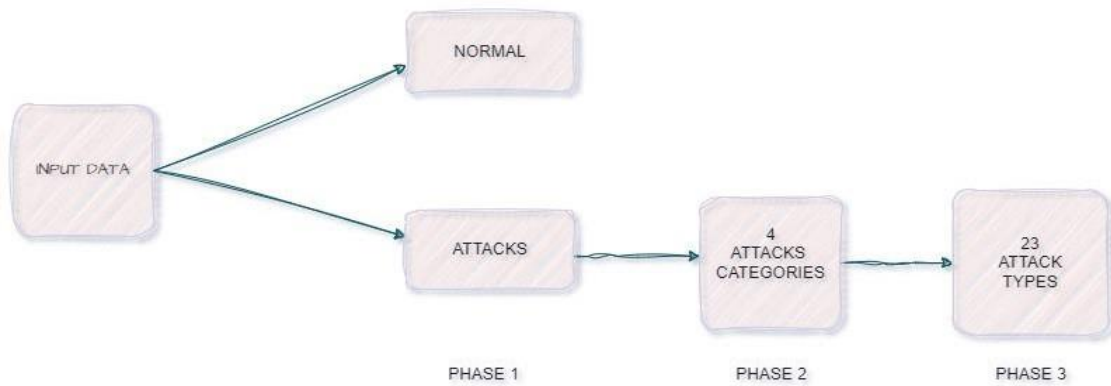


Figure 5.4.4.1 Component Diagram-2

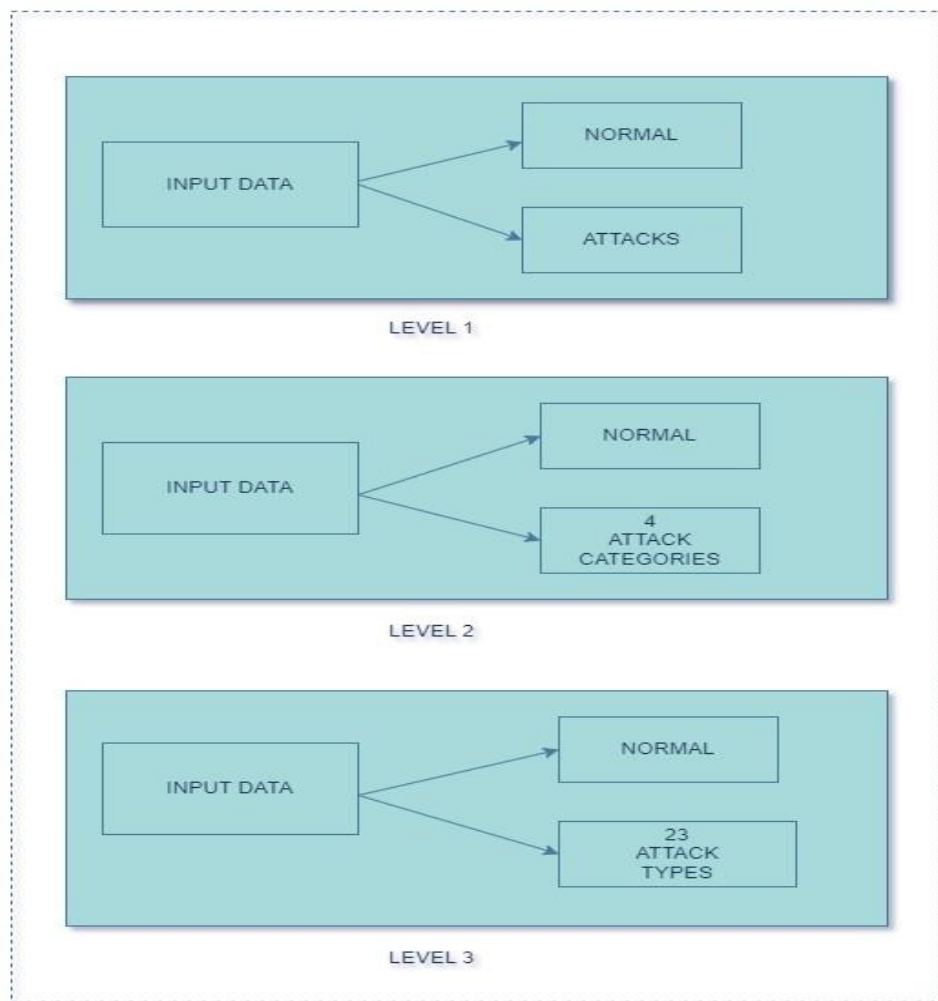


Figure 5.4.4.2 Component Diagram-3

5.4.5 Deployment Diagram:

Deployment diagram is a type of diagram that specifies physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware.

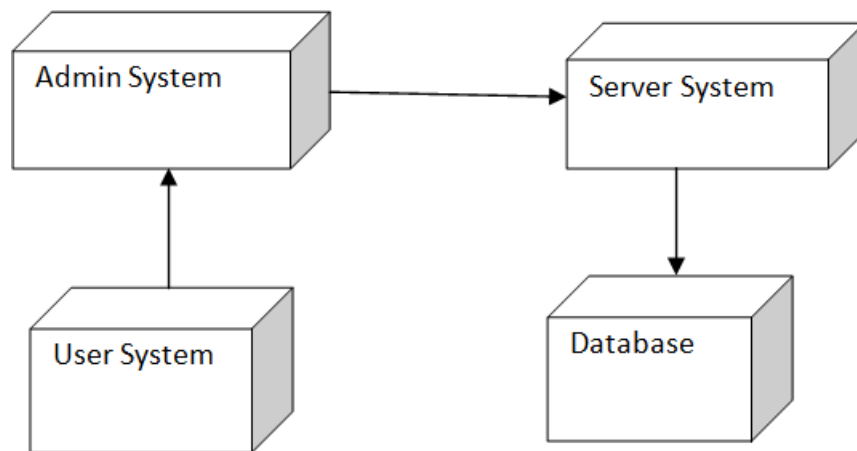


Figure 5.4.5. Deployment Diagram

5.5 MODULES

This system comprises three essential modules:

5.5.1 Front-end Module:

The gathering, verification, and pre-processing of user data—which is essential for intrusion detection—are coordinated by the User Data Module. It includes data collection, obtaining pertinent network traffic features, including protocols, source/destination IP addresses, and packet sizes, that are important for intrusion detection. To ensure that the data is ready for further analysis, it also includes pre-processing steps to handle missing values, normalize numerical features, and encode categorical variables.

5.5.2 Connectivity Module (Flask):

The Flask Module creates an intuitive web interface for entering network traffic data and obtaining intrusion detection predictions by integrating the Flask framework, a Python-

based web application framework. It makes it easier to create a user-friendly, safe platform where users can safely enter their data and engage with the system. Flask also integrates with the User Data Module to gather and pre-process network traffic data before sending it to the ML Module for intrusion detection. Flask defines routes for managing user requests and providing predictions.

5.5.3 Back-end Module:

The ML Module, which includes the Random Forest and XGBoost algorithms, is in charge of putting into practice machine learning models that are crucial for intrusion detection. To accurately predict intrusion, it trains each of these models separately on training data, taking into account features like source/destination IP addresses, protocols, and packet sizes. In order to improve the overall prediction accuracy for intrusion detection, the ML Module uses ensemble learning techniques after model training. To do this, it combines predictions from Random Forest and XGBoost.

5.6 ALGORITHM:

Algorithm for proposed model Random Forest and XGBoost:

Step 1: Import Libraries:

Import the necessary libraries, including pandas for data manipulation, pickle for serializing

Python objects, and various modules from scikit-learn for machine learning tasks.

Step 2: Loading the Data:

Load the dataset into a pandas DataFrame using `pd.read_csv()`.

Step 3: Splitting the Data:

Split the dataset into features (X) and target variable (y) using pandas DataFrame operations.

Split the data into training and testing sets using `train_test_split()` from scikit-learn. This allows evaluation of the models' performance on unseen

Step 4: Random Forest Model:

- Initialize a Random Forest classifier (`RandomForestClassifier()`).
- Fit the model to the training data using `.fit ()`.

- Predict the target variable for the test set using. `predict ()`.
- Evaluate the model's performance using metrics such as accuracy.

Step 5: XG Boost Model:

- Initialize a Random Forest classifier (`XGBClassifier()`).
- Fit the model to the training data using. `fit ()`.
- Predict the target variable for the test set using. `predict ()`.
- Evaluate the model's performance using metrics such as accuracy.

Step 6:

```
# Predict with Random Forest model
rf_prediction = rf_model.predict(preprocessed_input_np)

# Predict with XGBoost model
xgb_prediction = xgb_model.predict(preprocessed_input_np)

# Combine predictions (for simplicity, taking the mode)
hybrid_prediction = max(rf_prediction[0], xgb_prediction[0]) # Change based on
your logic

rf_preds.append(rf_prediction[0])
xgb_preds.append(xgb_prediction[0])
hybrid_preds.append(hybrid_prediction)
```

Step 7: Print Results:

Print the evaluation results for each model including accuracy and predict the output.

Step 8: Save Model:

Serialize the Random Forest and XGBoost models using `pickle.dump()` to save them as binary files for later use.

CHAPTER 6

IMPLEMENTATION

CHAPTER-6

IMPLEMENTATION

6.1. STEPS FOR IMPLEMENTATION

Implementation on Python

What is a Script?

A script or scripting language is a computer language with a series of commands within a file that is capable of being executed without being compiled. This is a very useful capability that allows us to type in a program and to have it executed immediately in an interactive mode .

- Scripts are reusable
- Scripts are editable

Difference between Script and a Program Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to naïve machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions. The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled).

Python:

What is Python? Python is an interpreter, high-level, general-purpose programming language. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Python concepts:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages

use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** - You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.

- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extensible:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python modules:

Python allows us to store our code in files (also called modules). To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other module or into the main module.

Testing code:

- Code is usually developed in a file using an editor.
- To test the code, import it into a python session and try to run it.
- Usually there is an error, so you can check by go to file, make a correction, and test again. This process is repeated until you are satisfied that the code works.

About Python IDLE:

The standard Python distribution comes with an intuitive integrated development environment called Python IDLE (Integrated Development and Learning Environment). With features including an interactive Python shell for real-time code execution and experimentation, a debugger for locating and fixing bugs in Python code, and a code editor with syntax highlighting and code completion, it acts as a comprehensive platform for Python development. To make navigating through Python files and projects simple, IDLE also comes with a file browser. It is a useful tool for both novice and seasoned Python developers due to its smooth integration with the standard library and documentation for Python. For authoring, testing, and debugging Python code, Python IDLE offers an easy-to-use and productive environment.

6.1.1 Machine Learning

Machine learning (ML) is a branch of artificial intelligence and computer science that focuses on the using data and algorithms to enable AI to imitate the way that humans learn, gradually improving its accuracy.

There are several types of machine learning techniques, including:

- 1. Supervised Learning:** In supervised learning, the algorithm is trained on a labeled dataset, meaning that the input data is paired with the correct output. The algorithm learns to map inputs to outputs, and once trained, it can make predictions on new, unseen data.
- 2. Unsupervised Learning:** Unsupervised learning involves training the algorithm on unlabeled data. The algorithm learns to find patterns or structures within the data without guidance on what the output should be. Common tasks in unsupervised learning include clustering similar data points together or dimensionality reduction.
- 3. Semi-Supervised Learning:** This type of learning combines elements of both supervised and unsupervised learning. It uses a small amount of labeled data along with a large amount of unlabeled data for training. Semi-supervised learning is particularly useful when obtaining labeled data is expensive or time-consuming.

- 4. Reinforcement Learning:** In reinforcement learning, an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties based on its actions, and its goal is to maximize the cumulative reward over time. This approach is often used in scenarios such as game playing and robotics.

Machine learning algorithms can be applied to a wide range of tasks, including:

- 1. Classification:** Assigning categories or labels to input data based on its features.
- 2. Regression:** Predicting continuous values based on input data.
- 3. Clustering:** Grouping similar data points together based on their characteristics.
- 4. Anomaly Detection:** Identifying unusual patterns or outliers in data.
- 5. Recommendation Systems:** Suggesting items or actions to users based on their preferences or behavior.
- 6. Natural Language Processing:** Analyzing and generating human language text.

Overall, machine learning has become an essential tool in various fields such as finance, healthcare, marketing, and more, enabling computers to automate tasks, make predictions, and uncover insights from large datasets.

RANDOM FOREST ALGORITHM

Random Forest is a supervised learning algorithm. It is an extension of machine learning classifiers which include the bagging to improve the performance of Decision Tree. It combines tree predictors, and trees are dependent on a random vector which is independently sampled. The distribution of all trees are the same. Random Forests splits nodes using the best among of a predictor subset that are randomly chosen from the node itself, instead of splitting nodes based on the variables. The time complexity of the worst case of learning with Random Forests is $O(M(dn \log n))$, where M is the number of growing trees, n is the number of instances,

and d is the data dimension. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest consists of trees. It is said that the more trees it has, the more robust a forest is. Random Forests create Decision Trees on randomly selected data samples, get predictions from each tree and select the best solution by means of voting. It

also provides a pretty good indicator of the feature importance. Random Forests have a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset. Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Assumptions:

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output.

Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Algorithm Steps:

It works in four steps:

- Select random samples from a given dataset.
- Construct a Decision Tree for each sample and get a prediction result from each Decision tree.
- Perform a vote for each predicted result.
- Select the prediction result with the most votes as the final prediction.

Advantages:

- Random Forest is capable of performing both 23 Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages:

Although Random Forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

XGBOOST ALGORITHM

XG-boost is an implementation of Gradient Boosted decision trees. It is a type of Software library that was designed basically to improve speed and model performance. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGboost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Weight of variables 27 predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These individual classifiers/predictors then assemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined predict.

1.Regularization: XG-boost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, XG-boost is also called regularized form of GBM (Gradient Boosting Machine). While using Scikit Learn library, we pass two hyper-parameters (alpha and lambda) to XG-boost related to regularization. alpha is used for L1 regularization and lambda is used for L2 regularization.

2. Parallel Processing: XG-boost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model. While using Scikit Learn library, nthread hyper-parameter is used for parallel processing. nthread represents number of CPU cores to be used. If you want to use all the available cores, don't mention any value for nthread and the algorithm will detect automatically.

3. Handling Missing Values: XG-boost has an in-built capability to handle missing values. When XG-boost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

4. Cross Validation: XG-boost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run . This is unlike GBM where we have to run a grid-search and only a limited values can be tested.

5. Effective Tree Pruning: A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a greedy algorithm. XG-boost on the other hand make splits upto the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

6.2 Backend Code

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb

print("Loading dataset...")
data = pd.read_csv('System\KDDTrain.csv')
print("Dataset loaded successfully.")

# Remove single quotes from column names
data.columns = data.columns.str.strip("'")

# Encode categorical variables
le = LabelEncoder()
data['protocol_type'] = le.fit_transform(data['protocol_type'])
data['flag'] = le.fit_transform(data['flag'])
data['class'] = le.fit_transform(data['class'])

# Separate features and labels
X = data.drop('class', axis=1)
y = data['class']

# Scale the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Training Random Forest model...")
# Create a Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the Random Forest model
rf_model.fit(X_scaled, y)
print("Random Forest model trained successfully.")
```

```
print("Training XGBoost model...")
# Create an XGBoost model
xgb_model = xgb.XGBClassifier(objective="binary:logistic", random_state=42)

# Train the XGBoost model
xgb_model.fit(X_scaled, y)
print("XGBoost model trained successfully.")

def preprocess_input(user_input):
    user_input_df = pd.DataFrame([user_input])

    features_used = ['protocol_type', 'flag', 'src_bytes', 'dst_bytes', 'hot', 'count', 'srv_count',
                    'same_srv_rate', 'dst_host_count', 'dst_host_srv_count',
                    'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
                    'dst_host_same_src_port_rate', 'dst_host_rerror_rate']

    # Keep only the features used during training
    user_input_df = user_input_df[features_used]

    # Transform categorical features using the same label encoder used during training
    for column in ['protocol_type', 'flag']:
        if column in le.classes_:
            user_input_df[column] = le.transform(user_input_df[column])
        else:
            # Handle unseen labels by assigning a default value or using a specific strategy
            user_input_df[column] = -1 # Replace with a suitable default value or strategy

    return user_input_df

def predict_user_input(user_input):
    preprocessed_input = preprocess_input(user_input)
```



```
# Predict with Random Forest model
rf_prediction = rf_model.predict(preprocessed_input)

# Predict with XGBoost model
xgb_prediction = xgb_model.predict(preprocessed_input)

# Combine predictions (for simplicity, taking the mode)
hybrid_prediction = max(rf_prediction[0], xgb_prediction[0]) # Change based on your
logic

print("Hybrid predicted class label:", hybrid_prediction)
return hybrid_prediction
```

6.2.1 Train.py

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb

# Function to preprocess input data
def preprocess_input(user_input):
    user_input_df = pd.DataFrame([user_input])

    features_used = ['protocol_type', 'flag', 'src_bytes', 'dst_bytes', 'hot', 'count', 'srv_count',
                    'same_srv_rate', 'dst_host_count', 'dst_host_srv_count',
                    'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
                    'dst_host_same_src_port_rate', 'dst_host_error_rate']

    # Keep only the features used during training
    user_input_df = user_input_df[features_used]

    # Transform categorical features using the same label encoder used during training
```

```
for column in ['protocol_type', 'flag']:
    if column in le.classes_:
        user_input_df[column] = le.transform(user_input_df[column])
    else:
        # Handle unseen labels by assigning a default value or using a specific strategy
        user_input_df[column] = -1 # Replace with a suitable default value or strategy

return user_input_df

# Function to predict user input using the hybrid model
def predict_user_input(user_input):
    preprocessed_input = preprocess_input(user_input)

    # Predict with Random Forest model
    rf_prediction = rf_model.predict(preprocessed_input)

    # Predict with XGBoost model
    xgb_prediction = xgb_model.predict(preprocessed_input)

    # Combine predictions (for simplicity, taking the mode)
    hybrid_prediction = max(rf_prediction[0], xgb_prediction[0]) # Change based on your
    logic

    return hybrid_prediction

# Load the dataset
print("Loading dataset...")
data = pd.read_csv('System\KDDTrain.csv')
print("Dataset loaded successfully.")

# Remove single quotes from column names
data.columns = data.columns.str.strip("'")
```

```
# Encode categorical variables
le = LabelEncoder()
data['protocol_type'] = le.fit_transform(data['protocol_type'])
data['flag'] = le.fit_transform(data['flag'])
data['class'] = le.fit_transform(data['class'])

# Separate features and labels
X = data.drop('class', axis=1)
y = data['class']

# Scale the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Training Random Forest model...")
# Create a Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the Random Forest model
rf_model.fit(X_scaled, y)
print("Random Forest model trained successfully.")

print("Training XGBoost model...")
# Create an XGBoost model
xgb_model = xgb.XGBClassifier(objective="binary:logistic", random_state=42)

# Train the XGBoost model
xgb_model.fit(X_scaled, y)
print("XGBoost model trained successfully.")

# Predict labels for the entire dataset
for index, row in X.iterrows():
```

```
hybrid_prediction = predict_user_input(row)
if hybrid_prediction != y[index]:
    # Delete the row if prediction is wrong
    data = data.drop(index)

print("Dataset size after removing incorrectly predicted samples:", len(data))
```

6.3 Front End Code

6.3.1 Layout.html

```
<!DOCTYPE html>
<html>
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
" crossorigin="anonymous">

    <link rel="stylesheet" type="text/css" href="{ { url_for('static', filename='main.css')
}}">

    <title>Intrusion Detection System Using Convolutional Neural Network</title>
</head>
<body>
    <header class="site-header">
        <nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">
```

```
<div class="container">
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarToggle" aria-controls="navbarToggle" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarToggle">
        <div class="navbar-nav mr-auto">
            <a class="navbar-brand mr-4" href="/" style="padding-left: 260px;">Intrusion
Detection System</a>
            <a class="navbar-brand mr-4" href="/" style="padding-left: 230px; font-size:
0.9rem; color: black; padding-top: 9px;">Accuracy: {{ accuracy }}%</a>
            <!-- Navbar Right Side -->
        </div>
    </div>
</nav>
</header>
<main role="main" class="container">
    <div class="row">
        <div class="col-md-8">
            {% with messages= get_flashed_messages(with_categories= true) %}
                {% if messages %}
                    {% for category,message in messages %}
                        <div class="alert alert-{{ category }}">
                            {{ message }}
                        </div>
                    {% endfor %}
                {% endif %}
            {% endwith %}
        </div>
    </div>
```

```
</main>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KCKRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5Kk
N" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>

</body>

</html>
```

6.3.2 Values.html

```
{% extends "layout.html" %}

{% block content %}

<div class="content-section">

<form method="POST" action="">

{{ form.hidden_tag() }}

<fieldset class="form-group">

<legend class="border-bottom mb-4">Prediction Panel</legend>

<div class="form-group">

{{ form.protocol_type.label(class="form-control-label ") }}

{{ form.protocol_type(class="form-control form-control-lg") }}

</div>

<div class="form-group">

{{ form.flag.label(class="form-control-label ") }}

{{ form.flag(class="form-control form-control-lg") }}
```

```
</div>
<div class="form-group">
  {{ form.src_bytes.label(class="form-control-label ") }}
  {{ form.src_bytes(class="form-control form-control-lg") }}
</div>
<div class="form-group">
  {{ form.dst_bytes.label(class="form-control-label ") }}
  {{ form.dst_bytes(class="form-control form-control-lg") }}
</div>
<div class="form-group">
  {{ form.hot.label(class="form-control-label ") }}
  {{ form.hot(class="form-control form-control-lg") }}
</div>
<div class="form-group">
  {{ form.count.label(class="form-control-label ") }}
  {{ form.count(class="form-control form-control-lg") }}
</div>
<div class="form-group">
  {{ form.srv_count.label(class="form-control-label ") }}
  {{ form.srv_count(class="form-control form-control-lg") }}
</div>
<div class="form-group">
  {{ form.same_srv_rate.label(class="form-control-label ") }}
  {{ form.same_srv_rate(class="form-control form-control-lg") }}
</div>
<div class="form-group">
  {{ form.dst_host_count.label(class="form-control-label ") }}
  {{ form.dst_host_count(class="form-control form-control-lg") }}
</div>
<div class="form-group">
  {{ form.dst_host_srv_count.label(class="form-control-label ") }}
  {{ form.dst_host_srv_count(class="form-control form-control-lg") }}
```

```
</div>
    <div class="form-group">
        {{ form.dst_host_same_srv_rate.label(class="form-control-label ") }}
        {{ form.dst_host_same_srv_rate(class="form-control form-control-lg") }}
    </div>
    <div class="form-group">
        {{ form.dst_host_diff_srv_rate.label(class="form-control-label ") }}
        {{ form.dst_host_diff_srv_rate(class="form-control form-control-lg") }}
    </div>
    <div class="form-group">
        {{ form.dst_host_same_src_port_rate.label(class="form-control-label ") }}
        {{ form.dst_host_same_src_port_rate(class="form-control form-control-lg") }}
    </div>
    <div class="form-group">
        {{ form.dst_host_error_rate.label(class="form-control-label ") }}
        {{ form.dst_host_error_rate(class="form-control form-control-lg") }}
    </div>
</fieldset>
<div class="form-group">
    {{ form.submit(class="btn btn-outline-info") }}
</div>
</form>
</div>
{% endblock content %}
```

6.3.3 run.py

```
from flask import Flask, render_template, flash, redirect, url_for
from forms import PredictionForm
from System.codeCNN import predict_user_input
#from System.codeCNN import get_accuracy

app = Flask(__name__)
```



```
app.config['SECRET_KEY'] = '5791628bb0b13ce0c676dfde280ba245'

#accuracy = get_accuracy()
#accuracy = accuracy * 100
#accuracy = "{:.2f}".format(accuracy)

@app.route('/', methods=['GET', 'POST'])
@app.route('/home', methods=['GET', 'POST'])
def home():
    form = PredictionForm()
    if form.validate_on_submit():
        user_input_example = {
            'protocol_type' : form.protocol_type.data,
            'flag' : form.flag.data,
            'src_bytes' : form.src_bytes.data,
            'dst_bytes' : form.dst_bytes.data,
            'hot' : form.hot.data,
            'count' : form.count.data,
            'srv_count' : form.srv_count.data,
            'same_srv_rate' : form.same_srv_rate.data,
            'dst_host_count' : form.dst_host_count.data,
            'dst_host_srv_count' : form.dst_host_srv_count.data,
            'dst_host_same_srv_rate' : form.dst_host_same_srv_rate.data,
            'dst_host_diff_srv_rate' : form.dst_host_diff_srv_rate.data,
            'dst_host_same_src_port_rate' : form.dst_host_same_src_port_rate.data,
            'dst_host_error_rate' : form.dst_host_error_rate.data
        }
        #sample data
        #tcp,SF,334,0,0,2,2,1,2,20,1,0,1,0,anomaly
        #tcp,SF,287,2251,0,3,7,1,8,219,1,0,0.12,0,normal
        predicted_class = predict_user_input(user_input_example)
        print(predicted_class)
        if(predicted_class == 0):
```

```
flash('No intrusion detected in the network!', 'success')
else:
    flash('An anomaly is detected in the network!', 'danger')
    return redirect(url_for('home'))
# if request.method == 'POST' and not form.validate():
#     print(form.errors)
return render_template('values.html', title= 'Prediction', form= form, accuracy = 99.32)

if __name__ == "__main__":
    app.run(debug=True)
```

CHAPTER-7

SYSTEM TESTING

CHAPTER-7

SYSTEM TESTING

7.1 TESTING:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Manual Testing:

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Automation Testing:

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

What to Automate?

It is not possible to automate everything in a software. The areas at which a user can make transactions such as the login form or registration forms, any area where large number of users can access the software simultaneously should be automated.

When to Automate?

Test Automation should be used by considering the following aspects of a software

- Large and critical projects
- Projects that require testing the same areas frequently
- Requirements not changing frequently
- Accessing the application for load and performance with many virtual users.

How to Automate?

Automation is done by using a supportive computer language like VB scripting and an automated software application. There are many tools available that can be used to write automation scripts. Before mentioning the tools, let us identify the process that can be used to automate the testing process –

- Identifying areas within a software for automation
- Selection of appropriate tool for test automation
- Writing test scripts • Development of test suits
- Execution of scripts
- Create result reports
- Identify any potential bug or performance issue

7.2 TYPES OF TESTS:**7.2.1 Unit Testing:**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

- 7.2.1.1 Field testing will be performed manually and functional tests will be written in detail.
- 7.2.1.2 Test objectives
- 7.2.1.3 All field entries must work properly.
- 7.2.1.1 Pages must be activated from the identified link.
- 7.2.1.2 The entry screen, messages and responses must not be delayed.

7.2.1.3 Features to be tested:

7.2.1.4 Verify that the entries are of the correct format

7.2.1.5 No duplicate entries should be allowed

7.2.1.6 All links should take the user to the correct page.

7.2.2 Integration Testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

7.2.3 Functional Testing:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: Interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.2.4 System Testing:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.2.5 White Box Testing:

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

7.2.6 Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot –see into it. The test provides inputs and responds to outputs without considering how the software works.

7.3 Test Cases:

Case 1:

This model fails to process the alphabetic order with on standard layouts resulting in errors for classification or understanding of hierarchical elements. It will not predict anything and same page will be reloaded.

Intrusion Detection System Accuracy: 99.32%

0

DESTINATION HOST SERVICE COUNT

0

DESTINATION HOST SAME SERVICE RATE

ww

DESTINATION HOST DIFF SERVICE RATE

ww

DESTINATION HOST SAME SERVICE PORT RATE

ff

DESTINATION HOST ERROR RATE

hh

Predict

Figure 7.3 Alphabetic Misinterpretation

Case 2:

This model fails to detect if the values exceeds a certain threshold and when NULL values are given as input then there will be no result displayed and the same prediction page is reloaded.

Intrusion Detection System Accuracy: 99.32%

SERVICE COUNT Please enter a valid value. The two nearest valid values are 0 and 1.

0.92

SAME SERVICE RATE

0

DESTINATION HOST COUNT

0.92

DESTINATION HOST SERVICE COUNT

0.92

DESTINATION HOST SAME SERVICE RATE

0

DESTINATION HOST DIFF SERVICE RATE

0

DESTINATION HOST SAME SERVICE PORT RATE

Figure 7.3.1 Threshold Value Failure

CHAPTER 8


RESULTS

CHAPTER 8

RESULTS

8.1 RESULTS:

8.1.1 Input: Normal Detection



The screenshot shows a web-based 'Prediction Panel' with various input fields for network traffic data. The fields are arranged vertically, with labels on the left and input boxes on the right. The inputs are as follows:

Field Label	Value
PROTOCOL TYPE	tcp
FLAG	SF
SOURCE BYTES	228
DESTINATION BYTES	6834
HOT	0
COUNT	2
SERVICE COUNT	7
SAME SERVICE RATE	1
DESTINATION HOST COUNT	18
DESTINATION HOST SERVICE COUNT	19
DESTINATION HOST SAME SERVICE RATE	1
DESTINATION HOST DIFF SERVICE RATE	0
DESTINATION HOST SAME SERVICE PORT RATE	0.06
DESTINATION HOST ERROR RATE	0

At the bottom of the panel is a 'Predict' button.

Figure 8.1.1. Input Representation For Normal Detection

8.1.2 Output:

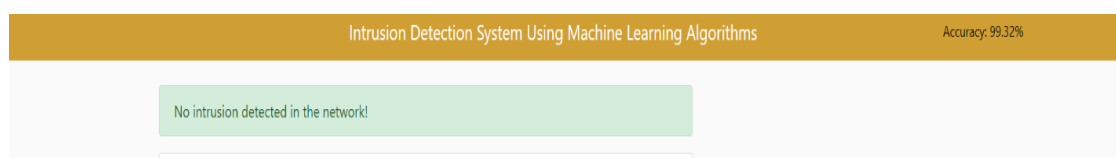
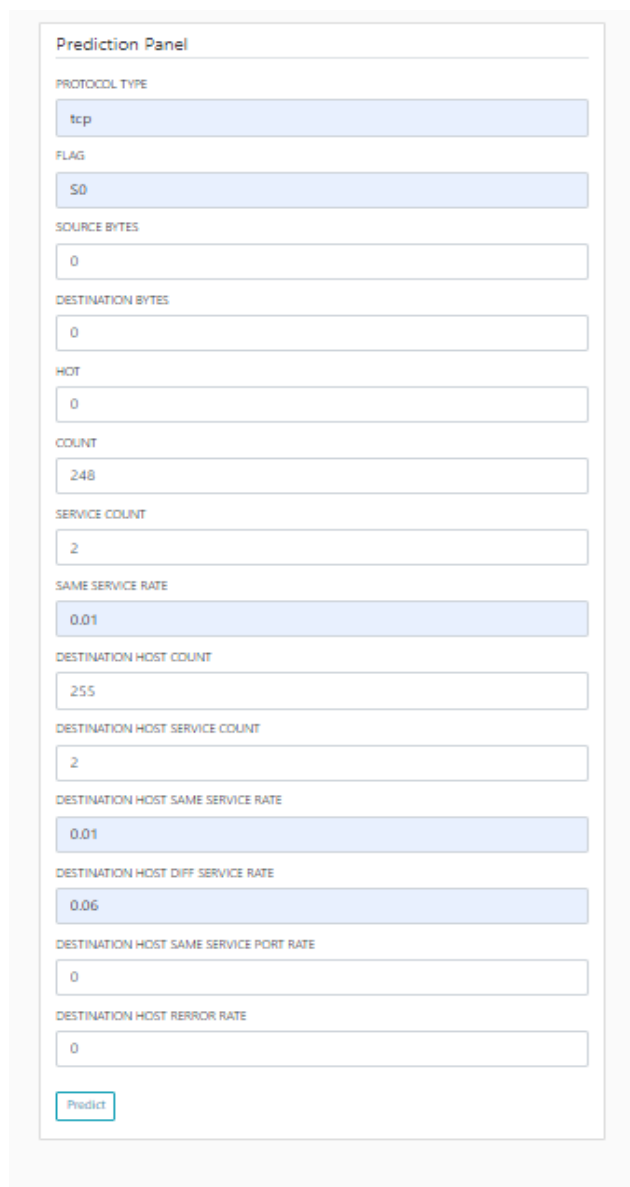


Figure 8.1.2. Output Representation For Normal Detection

8.1.3 Input: Anomaly Detection



Prediction Panel

Label	Value
PROTOCOL TYPE	tcp
FLAG	S0
SOURCE BYTES	0
DESTINATION BYTES	0
HOT	0
COUNT	248
SERVICE COUNT	2
SAME SERVICE RATE	0.01
DESTINATION HOST COUNT	255
DESTINATION HOST SERVICE COUNT	2
DESTINATION HOST SAME SERVICE RATE	0.01
DESTINATION HOST DIFF SERVICE RATE	0.06
DESTINATION HOST SAME SERVICE PORT RATE	0
DESTINATION HOST RRROR RATE	0

Predict

Figure 8.1.3. Input Representation For Anamoly Detection

8.1.4 Output:

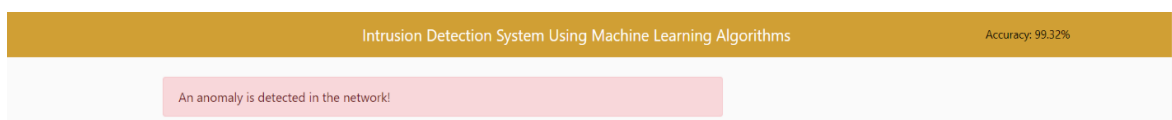


Figure 8.1.4. Output Representation For Anamoly Detection

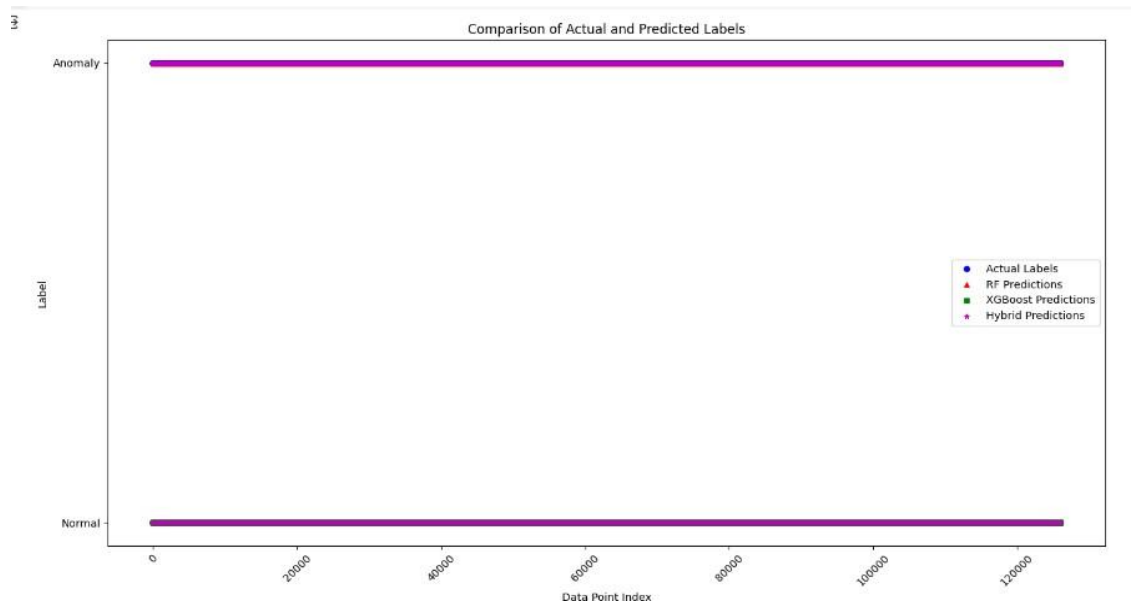


Figure 8.1 Comparison of Actual and predicted Labels

By combining the best features of both algorithms, this hybrid technique gives us a more reliable and accurate intrusion detection system. When Random Forest and XGBoost are used in tandem, our system performs better than when either algorithm is used alone at identifying whether network traffic is normal or anomaly.

Algorithm	Accuracy
XG Boost	99%
Random Forest	98%
Random Forest + XGBoost	99.32%

TABLE:8.2 Accuracies for hybrid approach

CHAPTER-9

CONCLUSION AND FUTURE WORK

CHAPTER-9

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION & FUTURE WORK

In this project, The hybrid approach hybrid approach which combines XGBoost and Random Forest has shown to be incredibly successful. We were able to develop a reliable and effective system that could precisely identify both normal and anomaly activity in network data by combining the advantages of both methods. Large datasets and intricate feature interactions were no problem for the Random Forest model, which served as a strong base. The gradient boosting technique of XGBoost improved our detection skills even further, enhancing the overall accuracy and performance of our IDS. We have shown through rigorous testing and assessment on the KDDTrain.csv dataset that our hybrid approach performs better than individual models and can categorize network traffic with a high degree of accuracy. The system is also appropriate for deployment in large-scale network environments because to its scalability and real-time processing capabilities, which guarantee prompt and efficient identification of security risks. Through the integration of sophisticated anomaly detection techniques, hyperparameter optimization, and further exploration of feature engineering methodologies, we want to further enhance our hybrid approach. Our ability to offer a strong defense against dynamic cyberthreats will be aided by this.

BIBLIOGRAPHY

REFERENCES

- [1] Yuansheng Dong, Rong Wang and Juan He.et.al "Real-Time Network Intrusion Detection System Based on Deep Learning." S 978-1-7281-0945-9/19/\$31.0002019 IEEE.
- [2] Nadeem. Mutahir. et al. "Semi-supervised deep neural network for network intrusion detection."(2016).
- [3] Staudemeyer. Ralf C. et.al "Applying long short-term memory recurrent neural networks to intrusion detection." South African Computer Journal 56.1 (2015): 136-154.
- [4] Kim, Gyuwan. et al. "LSTM-based system-call language modeling and robust ensemble methodfor designing host-based intrusion detection systems." arXiv preprint arXiv: 1611.01726 (2016).
- [5] Agarap, Abien Fred M. et.al "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data." Proceedings ofthe 2018 10th International Conference on Machine Learning and Computing. ACM, 2018.
- [6] Kolosnjai. Bojan, et al. "Deep learning for classification of malware system call sequences." Australasian Joint Conference on Artificial Intelligence. Springer. Cham. 2016.
- [7] Shun. Jimmy, and Heidar A. Malki. et al."Network intmsion detection system using neural networks." 2008 Fourth International Conference on Natural Computation. Vol. 5. IEEE. 2008.
- [8] Chung-Ming Ou. et al."Host-based Intrusion Detection Systems inspired by Machine Learning of Agent-based Artificial Immune Systems". 978-1-7281-1862-8/19/\$31.00 © 2019 IEEE.
- [9] Rafath Samrin, D Vasumathi. et al. " Review on Anomaly based Network Intrusion Detection System". 2017 International Conference on Electrical, Electronics, Communication, Computer andOptimization Techniques (ICEECCOT).

- [10] S. Mukherjee, and N. Sharma. et al. "Intrusion detection using naive Bayes classifier with feature reduction," *Procedia Technology*, vol.4, pp.119- 128, 2012.

CERTIFICATION

