A Report on Major Project

Intrusion Detection System Using Machine Learning

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER ENGINEERING

OF

VISHWAKARMA INSTITUTE OF TECHNOLOGY

Savitribai Phule Pune University

BY

Suhas Kadu (GR No. 11910400)
Kushal Kela (GR No. 11911231)
Saket Kolpe (GR No. 11910490)
Chetan Lohkare (GR No. 11910486)

UNDER THE GUIDANCE OF

Prof. Dr. S. R. Shinde

DEPARTMENT OF COMPUTER ENGINEERING

BANSILAL RAMNATH AGARWAL CHARITABLE TRUST'S
VISHWAKARMA INSTITUTE OF TECHNOLOGY
(An Autonomous Institute affiliated to Savitribai Phule Pune University)
PUNE - 411037
2022 - 2023

BANSILAL RAMNATH AGARWAL CHARITABLE TRUST'S

VISHWAKARMA INSTITUTE OF TECHNOLOGY

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

PUNE – 411037

CERTIFICATE

This is to certify that the Major Project titled Intrusion Detection System using Machine Learning submitted by **Suhas Kadu (GR No. 11910400), Kushal Kela (GR No. 11911231), Saket Kolpe (GR No. 11910490), Chetan Lohkare (GR No. 11910486)** is in partial fulfillment for the award of Degree of Bachelor of Technology in Computer Engineering of Vishwakarma Institute of Technology, Savitribai Phule Pune University. This project report is a record of bonafide work carried out by him under my guidance during the academic year 2022-2023.

Prof. Dr. S. R. Shinde
Dept. of Computer Engg.
Guide

Prof. Dr. S. R. Shinde
Vishwakarma Institute of Technology, Pune.
Head of Computer Department

Sign of External Examiner

Date

Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute Of Technology, Pune-37

Department Of Computer Engineering

PROJECT SYNOPSIS

Group No : 8

Group Members:

Roll No	Name	Class	Contact No	Email-ID
35	Suhas Kadu	CS - B	9822890892	suhas.kadu19@vit.edu
49	Kushal Kela	CS - B	7888264184	kushal.kela19@vit.edu
61	Saket Kolpe	CS - B	9766034306	saket.kolpe19@vit.edu
68	Chetan Lohkare	CS - B	7218250997	chetan.lohkare19@vit.edu

Academic Year : 2022-23
Project Title : Intrusion Detection System using Machine Learning
Project Area : Computer Networks, Machine Learning
Sponsor Company : -
Company Address : -

Internal Guide : Prof. Dr. Sandip R. Shinde

Name of the External Guide:
Contact No:

Signature of Internal Guide

ACKNOWLEDGEMENT

It gives us great satisfaction to be able to present this project on Intrusion Detection System using Machine Learning. We would like to express our deep gratitude towards our project guide **Prof. Dr. Sandeep Shinde**, for all the guidance and cooperation, without whom this project would have been an uphill task.

Suhas Kadu
Kushal Kela
Saket Kolpe
Chetan Lohkare

INDEX

Software Project Synopsis

Approvals Signature Block

Project Responsibility	Signature	Date
Project Guide (Internal)		
Project Guide (External)		
Documentation Leader		

1. CONTEXT

In the existing digital era, Internet and computer systems have increased innumerable security issues due to the volatile use of networks. As the digital world is emerging into society, new stuff like viruses and worms are being imported. The malignant users use different techniques like cracking of password, detecting unencrypted text are used to cause vulnerabilities to the system. Hence security is required for the users to secure their system from the intruders. Firewall technique is one of the well known protection techniques and it is used to protect the private network from the public network. Firewalls filter the incoming traffic from the Internet to evade the firewall. For example, external users can connect to the Intranet by dialing through a modem installed in the private network of the organization, this kind of access cannot be detected by the firewall. Hence to tackle this problem in this paper we would like to create a novel system that would allow us to detect these network attacks, this system is also known as Network Intrusion detection system (IDS). An intrusion detection system (IDS) is a device or software application that monitors a network or systems for malicious activity or policy violations. Any malicious activity or violation is typically reported to an administrator.It performs an analysis of passing traffic on the entire subnet, and matches the traffic that is passed on the subnets to the library of known attacks. Once an attack is identified, or abnormal behavior is sensed, the alert can be sent to the administrator.

2. PROBLEM

In recent years there has been a dramatic increase in the use of internet services thus leading to a large amount of networks.This has also led to huge network traffic and vulnerabilities like network intrusions.To distinguish the activities of a network traffic that whether its an intrusion or normal is very difficult and time consuming.Hence there is a requirement of an Intrusion Detection System(IDS) to avoid such attacks.

3. SOLUTION

This paper proposes the use of machine learning algorithms to create a novel Intrusion Detection System (IDS) to detect and classify network attacks. The emphasis is how machine learning algorithms can facilitate IDS with the capability to detect recognized network behavioral features, consequently ejecting the systems intruder and reducing the risk of compromise. To demonstrate the model's effectiveness, we used the UNSW-NB15 dataset, reflecting real modern network communication behavior.

Feasibility Study Report

Approvals Signature Block

Project Responsibility	Signature	Date
Project Guide (Internal)		
Project Guide (External)		
Documentation Leader		

1. CURRENT SYSTEMS AND PROCESSES

There has been a lot of work in the intrusion detection systems we found most systems NSL-KDD dataset, which is quite old. Some system use real time traffic for developing the the model. Some models are deployed and integrated with hardware so that they capture the real time data, pre-process it, pass it to the model and the model gives the result. The table below shows the current systems, their method and accuracy.

Table No.1: Current Intrusion Detection Systems

Sr. No.	Reference	Dataset	Method	Accuracy
1	[9]	NSL-KDD	SVM-Radial Basis Function (RBF)	98.1 %
2	[10]	KDDCup9	9 Cluster center and nearest neighbor (CANN)	99.9 %
3	[3]	UNSW Dataset	Convolutional Nueral Netwrosk	94.4 %
4	[11]	KDDCup9	Hybrid K Means and RBF Kernel Function	KMSVM: 88.71 % KM: 76.32% SVM: 100%
5	[12]	Kyoto 2006+	Bayesian Learning	87.41 %
6	[13]	Real time traffic	Principal Component Analysis and Ant Colony Optimization	96.53 %
	[14]	NSL-KDD	Stacking Ensemble using Naive Bayes, SVM, Random Forest, KNN, Multilayer Perception, CART	83.24 %

2. SYSTEM OBJECTIVES

a. To create an Intrusion Detection System(IDS) that will monitor network traffic to and from all the devices on the network.
b. To create a novel system to find intrusion characteristics for IDS using Machine learning algorithms.
c. The IDS will trigger alerts when suspicious activity or known threats are detected, so the administrator can examine and take appropriate actions to stop/block the attack.

3. ISSUES

There can be several issues that can arise while developing an Intrusion Detection System, some of them are listed below:

1. Data availability: The datasets out are a bit old and not updated frequently. But there are new types of attacks that are rising day by day. Most of the applications out there use the KDD dataset which is almost 20 years old. The dataset which is used in our project although huge has a lot of missing values and this amount of data cant be directly used to build an ML model.
2. Pre-processing: Pre-processing raw data has been always a time-consuming task. As the data is so big so a good technique should be used to pre-process it.
3. Number of features: In the case of network projects, a lot of parameters are involved. Like IP addresses, not of bits, type of protocol, although these are the important ones, some unnecessary parameters are involved which doesn't contribute much to the end result, hence an efficient method is required to select appropriate features or parameters.

4. ASSUMPTIONS AND CONSTRAINTS

1. Assumptions
    a. The system must predict whether there is an intrusion or not.
    b. To built a efficient model which not only good in accuracy but also in others metrics like precision, recall, and f1-score
2. Constraints
    a. This system doesn't specify the type of attack.
    b. This model is not yet deployed and cannot be used in real-time.

5. ALTERNATIVES

There are plenty of alternatives to Network Intrusion Detection Systems in the market. Every solution has its pros and cons with respect to the cost, available features, and security.

1. Snort
2. Suricata
3. Zeek
4. OpenWIGS-ng
5. Sguil
6. Security Onion

Use Case Analysis Document

Approvals Signature Block

Project Responsibility	Signature	Date
Project Guide (Internal)		
Project Guide (External)		
Documentation Leader		

USE CASE TEMPLATE

USE CASE	Detecting network intrusion attack using various parameters of network	
Goal	To detect maximum types of attack with greater accuracy using the most optimal features that help in detecting intrusion effectively.	
Purpose	To develop a algorithm that detects network intrusion more accurately.	
Preconditions	The user should have a network parameters to test.	
Success Condition	User is able to detect intrusion correctly or the condition is normal.	
Failed Condition	Intrusion is detected incorrectly when condition is normal or intrusion is not detected.	
Primary Actors	Network ports. Application to fetch network parameters.	
Secondary Actors	Active network connection	
Trigger	passing the dataset containing network parameters to the system.	
DESCRIPTION	**Step**	**Basic Course of Action**
	1	Preprocess the dataset.
	2	Train the dataset.
	3	Retrive the network parameters.
	4	Pass the input to the trained model.
	5	Check the result if there is any attack on the system.

Software Requirements Specification

Approvals Signature Block

Project Responsibility	Signature	Date
Project Guide (Internal)		
Project Guide (External)		
Documentation Leader		

1. SCOPE

The intrusion detection system is a system that would help us detect malicious network activity with the aim to prevent any further damage that can be caused by the attacker.

There are many potential areas for further development in this Intrusion Detection System. Some possible directions for future work include:

a. Real time Network Intrusion detection can be implemented using Deep Learning by learning features from unknown attacks.
b. We can add/use some more features while detecting for intrusions in the network.

2. REQUIREMENTS

a. Sklearn: A well-liked open-source Python package for machine learning is called scikit-learn (sklearn). It offers a collection of tools and APIs for putting a variety of machine-learning algorithms and methodologies into practice. Sklearn is used in our project's Intrusion detection system context for activities like dividing a dataset into training and testing sets and assessing the model's performance on a specific dataset. This module offers a selection of classes and functions for choosing and assessing the effectiveness of machine learning models. Using the prebuilt methods we have built multiple models by changing the default parameters. Also we have used ensemble learning to build models of very high accuracy. For instance, it has functions for dividing datasets into training and testing sets so that metrics like accuracy, precision, and recall may be used to assess a model's performance.
b. Pandas: Pandas is an open-source library designed primarily for working quickly and logically with relational or labeled data. It offers a range of data structures and procedures for working with time series and numerical data. The NumPy library serves as the foundation for this library. Pandas is quick and offers its users exceptional performance & productivity. In our project, we have used pandas to apply pre-processing methods, to find correlations between features and end results, to handle missing values, and to create a new dataset from raw dataset
c. Matplotlib: Matplotlib is a popular open-source library for data visualization in Python. It can be used to create a wide range of plots, charts, and visualizations to help understand and interpret data. In the context of neural machine translation in our project, Matplotlib is used to visualize the performance of our ML model and to compare the performance of different models on a particular dataset. For example, Matplotlib is used to create line plots showing the change in training and testing accuracy or loss during training, of different models. By using Matplotlib, we can more easily visualize and interpret the results of the models and can gain insights into the behavior of their models that may not be apparent from raw numerical data alone.
d. The UNSW-NB15 dataset:

The IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) creates the raw network packets for the UNSW-NB 15 data set in order to produce a hybrid of real current normal activities and synthetic recent attack behaviors. 100 GB of the raw traffic is captured using the Tcpdump tool. Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms represent the nine attack families in this data collection. Twelve algorithms are built and the Argus and Bro-IDS tools are used to generate a total of 49 characteristics with the class label. The UNSW-NB15 freatures.csv file contains a description of these features. The four CSV files UNSW-NB15 1.csv, UNSWNB15 1.csv, UNSW-NB15 1.csv, and UNSW-NB15 1.csv have a combined total of two million, 540,044 records. The event file with the name UNSW-NB15 LIST EVENTS is the ground truth table with the name UNSW-NB15 GT.CSV.

UnSW NB15 training-set.csv and UNSW NB15 testing-set, two partitions from this data set, are set up as a training set and testing set, respectively. The testing set has 82,332 records from various attack and normal categories, whereas the training set contains 175,341 records. As shown in Table No. 1, the table has 49 attributes along with their description. Most of attributes are of integer and float type, but it also has attributes of string and binary type. The end result is of binary type. We have converted the string type variables into float using OridinalEncoder to train the model. We have reduced the attributes by correlation, observation and by using feature reduction techniques hence keeping only important attributes. This is important as it will reduce the training time and better features will lead to a better model.

Table No.2: Attributes in UNSW NB-15 Dataset

No.	Name	Type	Description
1	srcip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol
6	state	nominal	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	integer	Source to destination packet count
18	Dpkts	integer	Destination to source packet count
19	swin	integer	Source TCP window advertisement value
20	dwin	integer	Destination TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number

22	dtcpb	integer	Destination TCP base sequence number
23	smeansz	integer	Mean of the ?ow packet size transmitted by the src
24	dmeansz	integer	Mean of the ?ow packet size transmitted by the dst
25	trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len	integer	Actual uncompressed content size of the data transferred from the server's http service.
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	record start time
30	Ltime	Timestamp	record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
34	synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
35	ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
36	is_sm_ips_ports	Binary	If source (1) and destination (3)IP addresses equal and port numbers (2)(4) equal then, this variable takes value 1 else 0
37	ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
39	is_ftp_login	Binary	If the ftp session is accessed by user and password then 1 else 0.
40	ct_ftp_cmd	integer	No of flows that has a command in ftp session.
41	ct_srv_src	integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dst	integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm	integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).

44	ct_src_ ltm	integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct_src_dport_ltm	integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46	ct_dst_sport_ltm	integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src_ltm	integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).
48	attack_cat	nominal	The name of each attack category. In this data set , nine categories e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
49	Label	binary	0 for normal and 1 for attack records

Software Project Plan

Approvals Signature Block

Project Responsibility	Signature	Date
Project Guide (Internal)		
Project Guide (External)		
Documentation Leader		

1. OVERVIEW

We adopted the Agile methodology of Software development to complete the project. We planned a sprint of 1 week. At the end of each week, we reviewed and merged all the code written during the sprint and also panned the tasks for the next week.

2. PROJECT GOALS

The main goal of the project was to build an Intrusion Detection System(IDS) that will monitor network traffic to and from all the devices on the network. To develop a novel system that uses machine learning techniques to identify intrusion characteristics for IDS. When suspicious behavior or recognized threats are found, the IDS will send alerts so the administrator may investigate and take the necessary precautions to stop or prevent the attack. Furthermore, by deploying the model, the user can have access to a web application that will be used to detect the attack on the system.

Project Goal	Priority	Comment/Description/Reference
Business Goals:		
Deploying the model	3	Deploying the model to a web application which can be directly used by anyone on the internet.
Technological Goals:		
Data Preprocessing	1	The rows having missing values are dropped. Filling the missing values with the most occurring value
Encoding Categorical Variables and Removing less important columns	2	The categorical variables are encoded using OrdinalEncoder. Columns that have a correlation between -0.3 to 0.3 are dropped Some basic pandas methods are applied to check the datatypes, no. of samples, null values, etc.
Building ML Models	3	Different ML models are built using the pre-processed dataset.

3. LITERATURE REVIEW

[1] In this paper the UNSW-NB15 dataset is used to develop a CNN model to find network intrusion with an accuracy of 94.4%. A good hyperparameter optimization technique is used. The sequence of Convo2D, MaxPooling, and Dropout is used which reduces overfitting. Adam optimizer is also used along with EarlyStopping and ModelCheckPoint so that the model reserves the best weights. 9 Types of attacks are classified using the model. Besides the predefined dataset, a new dataset is also created by merging data from other sources, and 95.6% accuracy is achieved on that dataset.

[2] The main idea of this paper is to use an advanced intrusion detection system with high network performance to detect the unknown attack package, by using a deep neural network algorithm, also in this model, the attack detection is done in two ways (binary classification and multiclass classification). It uses anomaly detection techniques without accessing information in the packet payload to avoid a breach of data privacy. The dataset used in this paper was KDD CUP 1999 which is an extremely large dataset that is used with intrusion detection experiments. This dataset has 41 features that can be classified into three main Categories: TCP connections features, Content features, and Traffic features.

[3] In this paper, to determine the root of issues with various machine learning algorithms in identifying invasive behaviors, a thorough examination and analysis of these techniques have been conducted. According to each attack, an attack classification and feature mapping are given. Additionally, problems with utilizing network attack datasets to detect low-frequency assaults are examined, and workable solutions are offered. Machine learning algorithms have been examined and contrasted in terms of how well they can identify the different types of attacks. Additionally, each category's restrictions are mentioned. The article also includes a number of machine learning data mining tools. Future approaches for attack detection using machine learning techniques are presented at the conclusion. [4] In this paper, a survey of the Intrusion Detection Systems (IDS) using the most recent ideas and methods proposed for the IoT is presented. To understand and illustrate IDS platform differences and the current research trend towards a universal, cross-platform distributed approach, the survey starts with a historical examination of intrusion detection systems. This examination of the foundations of IDS research based on the components that make up the IoT is followed by a look at the current holistic trend and analysis of these schemes. Finally, guidelines to potential IDS in the IoT are proposed before identifying the open research problems.

[5] In this paper, a Support Vector Machine with a nonlinear scaling method is used for intrusion detection. The UNSWNB15 dataset containing more contemporary behaviors is used for showing the effectiveness of the proposed SVM-based model. The experiments are carried out both on binary classification and multi-classification, and the results perform well. For binary-classification of the proposed method, the accuracy reaches 85.99% and Multi-classification results also perform well with 75.77%.

[6] This paper tackles the issues by presenting a revolutionary deep-learning intrusion detection technique. Authors describe their nonsymmetric deep autoencoder (NDAE) proposal for unsupervised learning features. They also offer their original deep learning categorization model, which was built using stacked NDAEs. Utilizing the benchmark KDD Cup '99 and NSL-KDD datasets, their suggested classifier has been implemented in TensorFlow which is GPU-enabled. Their model has so far shown promising findings that show advantages over current methods and a high likelihood of being applied to contemporary NIDSs. [7] This paper talks about the need and necessity of IDS and how they help to avoid attacks on the system even when we have firewalls present to defend the system. It also talks about how the IDS evolutionized and what were some of the important factors that motivated its evolution.The IDS can be classified on the basis of their architectural structure, the type of system it protects, and the processing time of the data. According to their location, there are two types of intrusion detection systems, Host-Based and Network-Based. Also, IDSs can be classified according to their techniques such as Signature-Based and Anomaly-Based.The paper provides brief info on various Intrusion Detection Approaches used such as ANN, SVM, Rule-Based Systems, Fuzzy Logic, etc. The most challenging phase to determine the performance of Intrusion Detection Systems is to find the appropriate dataset and hence the paper provides the most commonly used datasets for attack detection systems.

[8] This paper uses an unsupervised ML algorithm to build a model for Intrusion Detection System. Main focus if this paper is to reduce false positives and false negative rates, using a signature-based approach. K-means clustering is used with different cluster values. NSL-KDD dataset is used, best results were generated when a number of clusters is equal to the data types in the dataset. Firstly, the data is pre-processed

then the K-means algorithm is used with different values of k and metrics are calculated for each value of k. It is found that increasing or decreasing the number of clusters beyond the number of data types decreases the efficiency of the model. Other unsupervised techniques like Bayesian and Hierarchical can be used to compare the results with K-means. [13] In this paper, the Deep Learning approach is used to build an intrusion detection system. Multiple datasets like KDDCup 99 NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS 2017 are used. Both Binary and Multiclass classification is done. The experiments are done with 1000 epochs and a learning rate between 0.1 to 0.5. [9] This paper presents an intrusion detection model that combines chi-square feature selection and multi-class support vector machine to achieve high accuracy and low false positive rates. The model optimizes the kernel parameter using a technique called variance tuning, which involves identifying the highest attribute variance and using that information to determine the best kernel parameter. This variance tuning technique results in a more accurate SVM classifier with minimal time complexity. The proposed model has an average accuracy of more than 95% for all attacks and normal traffic, though the accuracy for U2R attacks is lower due to a smaller number of training samples.

[10] This paper proposes a new method for representing data in a dataset for effective and efficient intrusion detection. This method, called CANN, combines cluster centers and nearest neighbors. First, k-means clustering is used to identify the cluster centers of each category in the dataset. Then, the nearest neighbor of each data sample in the same cluster is found. The sum of the distance between the data sample and the cluster centers and the distance between the data sample and its nearest neighbor is calculated to create a distance-based feature representation. This new dataset, containing only one dimension, is then used for k-Nearest Neighbor classification, which allows for effective and efficient intrusion detection. The idea behind CANN is that the cluster centers provide discrimination capabilities for recognizing both similar and dissimilar classes, and the distances between a data sample and these cluster centers can provide additional information for recognition. Similarly, the distance between a data sample and its nearest neighbor in the same class also has discriminatory power. [12] This research focuses on the issue of detecting anomalies in order to identify intrusions, and presents a new approach that utilizes a Bayesian-based infinite bounded Generalized Gaussian mixture model. This model includes a feature selection component to ensure that only relevant features are used in the modeling process. The Bayesian methodology was chosen because it can prevent overfitting and underfitting, incorporate prior knowledge, and express uncertainty through probability distributions. The use of an infinite assumption, rather than a finite one, allows the model to learn both the parameters and number of components simultaneously. The inclusion of a feature selection mechanism aims to improve accuracy by only considering the most relevant features.

System Implementation Document

Approvals Signature Block

Project Responsibility	Signature	Date
Project Guide (Internal)		
Project Guide (External)		
Documentation Leader		

1. SYSTEM ARCHITECTURE

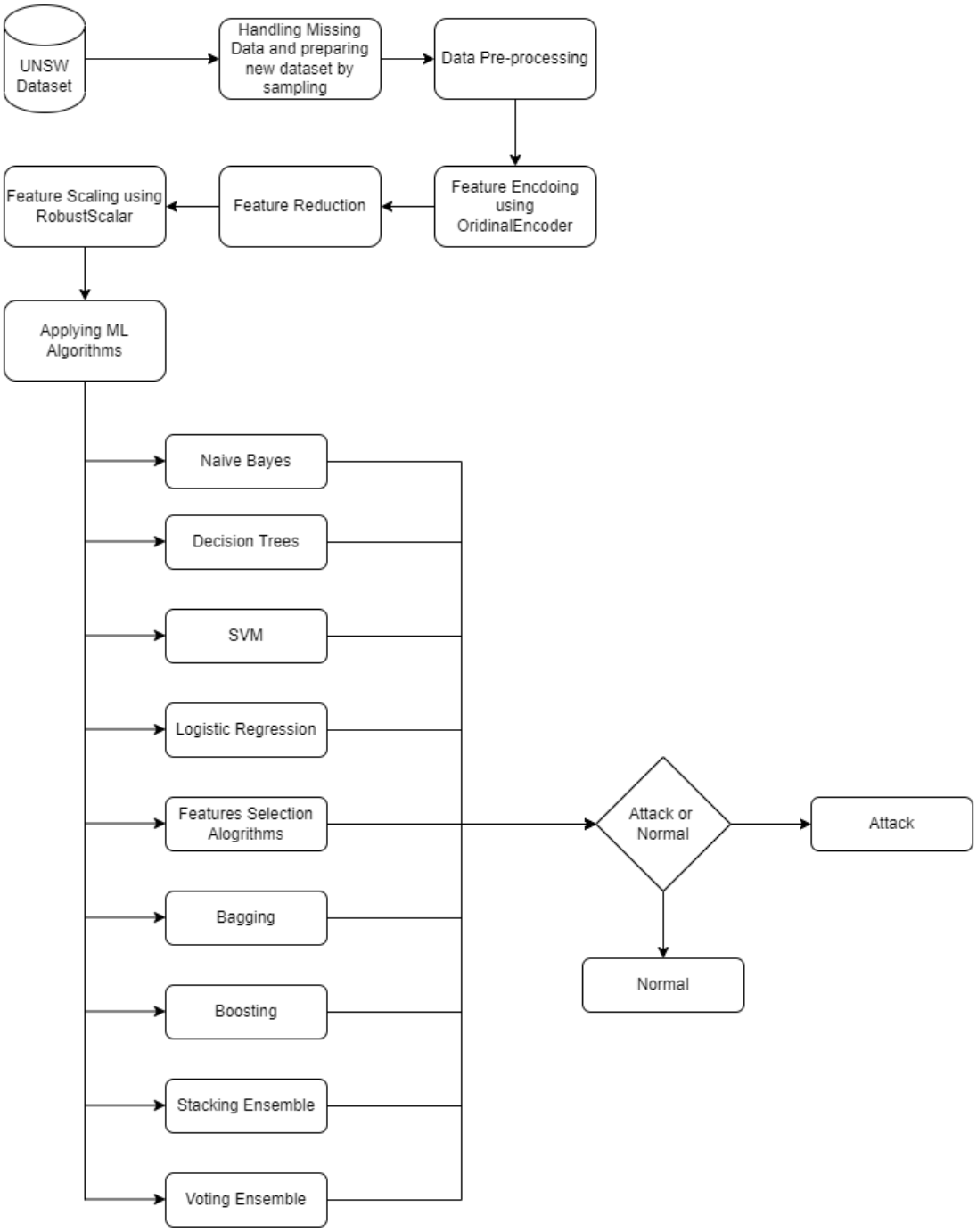

Fig No. 1: System Architecture

Above figure shows the flow of the project. At first, the UNSW dataset is taken as there was a huge amount of data so the missing values are handled by deleting the rows and columns. For the Normal class i.e. where there is no intrusion we have randomly selected some rows and marked them with the label Normal, thus a new dataset is prepared. This new dataset is then pre-processed using pandas. After that, the features are encoded using OrdinalEncoder, then features having less correlation with the end result are removed. All the features are then scaled using RobustScalar so that their values will be in the same range. Then using ML algorithms multiple models are built. Finally, the results are represented in table and graph format.

2. IMPLEMENTATION

The project is divided into three parts.

- Although the dataset already contains pre-processed train and test files, we have used the 4 raw dataset files which are not pre-processed. These files didn't even have column names, first, we named all the columns from the UNSW-NB15 features.csv file.
- In the first part the missing values from the huge dataset are handled. The dataset has around 2 million rows and is divided into 4 datasets. To handle the missing values three techniques are used, at first, the first dataset is taken and each technique is applied to it, after that the best technique is selected and applied to the remaining datasets. Only three columns have missing values viz. ct_flw_http_mthd, is_ftp_login columns, and attack_cat. The ct_flw_http_mthd and is_ftp_login columns have very very less correlation with the final result so they are directly dropped, the column attack_cat is very important hence it cannot be dropped. The techniques are as follows:
    1. Removing the rows and columns: At first, the first file is taken. The rows having missing values are dropped.
    2. Filling the missing values with the most occurring value:
    3. In this technique, the missing values are replaced by the most occurring values in that column, which results in bias on a particular value.
    4. Using sklearn Imputers: In this technique, SimpleImputer, IterativeImputer, and KnnImputer are used to impute the missing values. For IteravtiveImputer LabelEncoder is used and for KnnImputer MinMaxScalar is used, but these imputers also result in bias of particular value.

Out of all the above approaches, the Dropping of rows and columns is one good method for handling these missing values. The other two methods, replacing with most_frequent value and Imputers create a bias and as the dataset is huge, hence some of the rows can be dropped to avoid this bias.
After this same technique is applied to the remaining datasets. Then combination of two datasets i.e datasets 1 and 2 are made from the four because it has a better distribution of attack categories and is very less biased, the total samples in this dataset are 52,000. For the label 0 i.e when there is no intrusion, we have randomly selected 40,000 rows and they are randomly merged with 52,000 samples.

- In the second part, at first the categorical variables are encoded using OrdinalEncoder. Ordinal Encoder converts categorical values into float datatype. Then correlation is calculated between the variables and the final result. Those columns which have a correlation between -0.3 to 0.3 are dropped. Also, source and destination ip address columns are dropped as they are in string format and even if they are encoded different IPs will have the same value. Then some basic pandas methods are applied to check the datatypes, no. of samples, null values, standard deviation, mean, maximum and minimum values, etc.

- In the third part, ML models are built using the pre-processed dataset. At first feature, scaling is done so that all the values will be in the same range. RobustScalar is used for scaling.
    - RobustScalar: This Scaler scales the data according to the quantile range without using the median (defaults to IQR: Interquartile Range). The 1st quartile (25th quantile) to 3rd quartile interquartile range (IQR) is the range of values (75th quantile). By calculating the relevant statistics on the samples in the training set, centering and scaling are applied independently to each feature. Then, using the transform approach, the median and interquartile range are stored for use in later data. RobustScalar is very less likely to be susceptible to outliers.

After scaling various ML models are built using the following algorithms:

1. Naive Bayes: A group of supervised learning algorithms known as "naive" Bayes methods utilize Bayes' theorem with the "naive" assumption that every pair of features is conditionally independent given the value of the class variable. Gaussian Naive Bayes Algorithm is used for which the likelihood is as follows.

$$p(x_i|y) = \frac{1}{\sqrt{2\Pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \quad (1)$$

where, μ = mean,
σ^2 = variance.

2. Decision Trees: A non-parametric supervised learning technique for classification and regression is called a decision tree (DT). The objective is to learn simple decision rules derived from the data features in order to build a model that predicts the value of a target variable. We have built multiple models using Decision by changing the parameters like criterion and max_features. Different functions like, gini, and entropy are used for splitting, and sqrt and log2 are used for feature selection during a split.

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \quad (2)$$

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2 \quad (3)$$

3. Support Vector Machine: Finding a hyperplane in an N-dimensional space (N is the number of features) that categorizes the data points clearly is the goal of the support vector machine algorithm. We have built multiple models using SVM changing the kernel function. For kernel, we have used the Radial Basis function, Polynomial, Linear, and Sigmoid.
    a. Radial Basis Function Kernel:

$$K(x, x') = e^{-\gamma\|x-x'\|^2} \quad (4)$$

where, $\gamma = \frac{1}{n\,features * \sigma^2}$

    b. Polynomial Kernel

$$K(x, x')(\gamma\langle x, x'\rangle + r)^d \quad (5)$$

where, Υ = coefficient,
d = degree

    c. Linear Kernel:

$$K(x, x') = \langle x, x'\rangle \quad (6)$$

    d. Sigmoid Kernel:

$$K(x, x') = \tanh(\gamma\langle x, x'\rangle + r) \quad (7)$$

where, Υ = hyperparameter,
r = coefficient

4. Logistic Regression: When the dependent variable (target) is categorical, logistic regression is used. The algorithm used to solve the optimization problem is liblinear because the dataset is small. Two models are built using the l1 and l2 penalty. L1 regularisation penalizes the total of the weights' absolute values, whereas L2 regularisation penalizes the total of the weights' squares.
5. Logistic regression with Recursive Feature Elimination: Recursively considering fewer and smaller sets of features is the goal of recursive feature elimination (RFE), which aims to choose features. The importance of each feature is first determined by training the estimator on the initial set of characteristics and then by any individual attribute or callable. The least crucial features are then removed from the present list of features. Once the appropriate number of features to pick has been reached, the technique is recursively repeated on the pruned set. This algorithm is used Logistic Regression and 11 most important features are selected out 21 features.
6. Logistic regression with Forward Feature Elimination: Forward Feature selection uses SequentialFeatureSelector to find no. of specified features. It works on the greedy method, initially, it starts with zero features, then adds the most important features in the set till no. of required features is reached. This algorithm is used Logistic Regression and the 11 most important features are selected out of 21 features. After that, a new model is built using these 11 selected features.
7. Logistic regression with Backward Feature Elimination: Backward Feature selection uses SequentialFeatureSelector to find no. of specified features. It works on the greedy method, initially, it starts all features, then eliminates the least important features in the set till no. of required features are reached. This algorithm is used Logistic Regression and the 11 most important features are selected out of 21 features. After that, a new model is built using these 11 selected features.
8. Random Forest Importance: Using RandomForestClassifier the importance of each feature is calculated and features having very low importance are dropped. Out of 21, 3 features are dropped and a new model is built.
9. Bagging Classifier: An ensemble meta-estimator called a bagging classifier fits base classifiers one at a time to random subsets of the original dataset, and then it aggregates the individual predictions (either by voting or by averaging) to provide a final prediction. The BaggingClassifier is built using two base estimators or models, d_tree_entropy_sqrt, and svm_linear, the no. of estimators is set to 50 and the random state is equal to 42.
10. RandomForest: A random forest is a meta-estimator that uses averaging to increase predicted accuracy and reduce overfitting after fitting numerous decision tree classifiers to different dataset subsamples. The max samples parameter controls the sub-sample size. For the RandomForest classifier also multiple models are built by changing its parameters. Different split functions like gini and entropy are used and to sub-sample features, sqrt and log2 methods are used.
11. Extra Tree: The ExtraTreeClassifer class provides a meta estimator that uses averaging to increase prediction accuracy and prevent overfitting while fitting a variety of randomized decision trees (also known as extra-trees) on different sub-samples of the dataset. We have built two models using ExtraTreeClassifier out of which one uses gini and the other uses entropy for splitting.
12. AdaBoost: A meta-estimator called an AdaBoost classifier starts by fitting a classifier to the initial dataset. It then fits additional copies of the classifier to the same dataset, but with the weights of instances that were incorrectly classified being changed so that later classifiers would concentrate more on challenging cases. We have two models using AdaBoostClassifier, for one model the base estimator is d_tree_entropy_sqrt, and the other model uses nb_gaussian as the base estimator.

13. GradientBoosting: This approach allows for the optimization of any differentiable loss function and constructs an additive model in a forward stage-wise manner. Each stage involves fitting n classes_ regression trees on the loss function's negative gradient, such as a binary or multiclass log loss. In the particular scenario of binary classification, just one regression tree is generated. In this case, we have kept the loss function as log_loss, no. of estimators equal to 30, and to compute the initial predictions decision tree with entropy function and max features equal to sqrt, and other Gaussian Naive Bayes are used.
14. Stacking: In stacked generalization, the output of each individual estimator is stacked, and the final prediction is computed using a classifier. By using each estimator's output as the input for a final estimator, stacking enables the utilization of each estimator's strongest attributes. We have stacked 5 estimators viz. d_tree_entropy_sqrt, svm_linear, rf_entropy_log2 and nb_gaussian, ext_classifier_entropy_log2.For the final estimator, we have used logistic regression with an l1 penalty.
15. Voting: Using a majority vote or the average projected probability (soft vote), the VotingClassifier combines conceptually distinct machine learning classifiers to predict class labels. In order to counteract the flaws of a group of equally effective models, such a classifier can be helpful. We have used 5 estimators viz. d_tree_entropy_sqrt, svm_linear, log_reg_l1, rf_entropy_log2, and nb_gaussian. Both hard and soft is applied on the above estimators, for hard we have tried different weights and preserved the best one which give maximum accuracy. For soft voting, prediction scores are used as weights.

3. DEVELOPMENT AND TESTING

We have developed multiple models from one ML algorithm, by changing the parameters and comparing them. As accuracy cannot be the only metric to judge an ML model. We have also used other metrics such as precision, recall, and fl-score. We mainly focus on accuracy and f1-score as the fl-score is the harmonic mean of precision and recall. All the results are represented below in tables and graphs.

Table No.3: Naive Bayes

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
nb_gaussian	0.975525	0.958699	1.00000	0.978914	0.975993	0.959594	1.00000	0.979380

Although the recall is 1, means it detects the all the positive samples, but accuracy and f1-score is not that good.

Table No.4: Decision Trees

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
d_tree_sqrt_entropy	0.999168	0.998619	0.99991	0.999268	0.999066	0.998552	0.999	0.999181
d_tree_sqrt_gini	0.825119	0.764630	1.00000	0.866618	0.826343	0.766525	1.000	0.867834
d_tree_log2_entropy	0.825119	0.764630	1.00000	0.866618	0.826343	0.766525	1.000	0.867834
d_tree_log2_gini	0.926776	0.923791	0.94943	0.936439	0.924169	0.924663	0.943	0.934182

The model d_tree_sqrt_entropy has the best accuracy and fl-score out of all the algorithms. The models d_tree_sqrt_gini and d_tree_log2_entropy have exact same metrics despite different parameters, but the training time for d_tree_sqrt_gini(0.0189s) is less as compared to d_tree_log2_entropy(0.0219s) and the testing time is very very close. D_tree_log2_gini models has poor performance.

Table No.5: Support Vector Machine

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
svm_rbf	0.999538	0.999864	0.99932	0.999593	0.999425	0.999811	0.999181	0.999496
svm_poly	0.870495	0.814378	0.99997	0.897683	0.868967	0.813218	0.999811	0.896912
svm_linear	0.9996	1.000000	0.99929	0.999647	0.999497	1.000000	0.999117	0.999559
svm_sigmoid	0.921847	0.931546	0.93083	0.931193	0.917987	0.930519	0.925240	0.927872

svm_linear and svm_rbf have very similar metrics but svm_linear is best of all, with high accuracy, high f1-score and 100% precise. svm_sigmoid has poor performance and svm_poly performs worst.

Table No.6: Logistic Regression

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
log_reg_l1	0.9996	1.000000	0.99929	0.999647	0.999461	0.999937	0.9991	0.999527
log_reg_l2	0.9996	1.000000	0.99929	0.999647	0.999461	0.999937	0.9991	0.999527

Both the models are 100% precise have exact metrics but during coding, we found the training and testing time for log_reg_l1(0.1934s and 0.0019s) is very low as compared to log_reg_l2(0.4039s and 0.0009s).

Table No.7: Feature Selection Algorithms

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
rfc_imp	0.975063	0.958906	0.99891	0.978502	0.975526	0.959784	0.9989	0.978965
log_reg_rfe	0.9996	1.000000	0.99929	0.999647	0.999497	1.000000	0.9991	0.999559
log_reg_ffe	0.9995	-	-	-	0.9994	-	-	-
log_reg_bfe	0.87948	-	-	-	0.8776	-	-	-

In above feature selection algorithms we have reduced the number of features from 21 to 11. For each model best 11 features will be selected and the model is built using these features. log_reg_rfe performs best and is 100% precised. rfc_imp gives poor performance. log_reg_ffe and log_reg_bfe use SequentialFeatureSelector which does not has to predict method, hence only accuracy is mentioned, log_reg_ffe gives very good accuracy as compared to log_reg_bfe.

Table No.8: BaggingClassifier

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
bagging_clf_d_tree	0.975171	0.958127	1.000000	0.978616	0.975418	0.958666	1.000000	0.978897
bagging_clf_svm_linear	0.965113	0.957042	0.982703	0.969703	0.963414	0.957701	0.979072	0.968269

Both the models perform very poor w.r.t to accuracy and f1-score as compared to other model covered so far. It was found that bagging_clf_svm_linear takes lot of time for training(1026.4731s) and testing(232.0518s).

Table No.9: Random Forest Classfier

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
rf_gini_sqrt	0.994856	0.991105	0.99991	0.995492	0.995148	0.991746	0.9998	0.995762
rf_gini_log2	0.992607	0.987780	0.99934	0.993531	0.992812	0.988645	0.9988	0.993729
rf_entropy_log2	0.994224	0.990568	0.99934	0.994939	0.994501	0.991552	0.9988	0.995195
rf_entropy_sqrt	0.993731	0.989165	0.99991	0.994513	0.994322	0.990322	0.9998	0.995044

In case of Random Forest Classifier all the algorithms give very good accuracy and fl-score, but rf_entropy_log2 is the best of all.

Table No.10: Extra Tree Classifier

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
ext_classifier	0.924589	0.952628	0.91264	0.932209	0.921689	0.952277	0.9081	0.929694
ext_classifier_entropy_log2	0.975571	0.958774	1.00000	0.978953	0.975957	0.959591	0.9999	0.979349

Extra Trees Classifier didn't perform well as compared to other algorithms, discussed so far.

Table No.11: AdaBoost Classfier

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
ada_boost_clf_d_tree	0.99578	0.992706	0.99991	0.996299	0.995903	0.992989	0.9998	0.996419
ada_boost_clf_nb	0.931412	0.892278	1.00000	0.943073	0.930386	0.891186	1.0000	0.942462

AdaBoost Classifier performs very good when the base estimator is d_tree_sqrt_entropy, with nb_gaussian it gave a poor performance.SVM and Logistic Regression take infinite time or give probability error when used as base estimators.

Table No.12: GradientBoosting

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
grad_boost_clf	0.999969	1.000000	0.99994	0.999973	0.999856	1.000000	0.9997	0.999874
grad_boost_clf_nb	0.684693	0.848404	0.54181	0.661306	0.688338	0.849330	0.5511	0.668476

Gradient Boosting Classifier performed very well when initial estimator was d_tree_sqrt_entropy. The accuracy and f1-score with nb_gaussian as the initial estimator are very very low.

Table No.13: Stacking Ensemble

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
stacking_clf	0.997258	0.995893	0.99929	0.997591	0.99773	0.996918	0.9991	0.998017

Best models from each type of algorithm i.e. from Decision Trees, SVM, etc. are selected In total five best models are stacked and passed into the Stacking Ensemble, the final estimator for Stacking Ensemble is log_reg_l1 and then accuracy and other metrics are calculated, which are very good than other models.

Table No.14: Voting Ensemble

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
voting_clf	0.9996	1.000000	0.99929	0.999647	0.999497	1.000000	0.9991	0.999559
voting_clf_soft	0.994624	0.990627	1.00000	0.995291	0.99522	0.991686	1.0000	0.995826

In case of Voting Ensemble best models from each type of algorithm i.e. from Decision Trees, SVM, etc. are selected In total five best models are combined and passed into the Voting Ensemble and then accuracy and other metrics are calculated. In the case of voting_clf, hard voting is used, in hard voting, the weights were calculated manually and the best weights were preserved. For voting_clf_soft, the test accuracies are used as weights.

Table No.15: All Models

All the tables above are combined and merged into one table. Then multiple graphs are plotted using these values.

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
nb_gaussian	0.975525	0.958699	1.00000	0.978914	0.975993	0.959594	1.0000	0.979380
d_tree_sqrt_entropy	0.999168	0.998619	0.99991	0.999268	0.999066	0.998552	0.9998	0.999181
d_tree_sqrt_gini	0.825119	0.764630	1.00000	0.866618	0.826343	0.766525	1.0000	0.867834
d_tree_log2_entropy	0.825119	0.764630	1.00000	0.866618	0.826343	0.766525	1.0000	0.867834
d_tree_log2_gini	0.926776	0.923791	0.94943	0.936439	0.924169	0.924663	0.9438	0.934182
svm_rbf	0.999538	0.999864	0.99932	0.999593	0.999425	0.999811	0.9991	0.999496
svm_poly	0.870495	0.814378	0.99997	0.897683	0.868967	0.813218	0.9998	0.896912
svm_linear	0.9996	1.000000	0.99929	0.999647	0.999497	1.000000	0.9991	0.999559
svm_sigmoid	0.921847	0.931546	0.93083	0.931193	0.917987	0.930519	0.9252	0.927872
log_reg_l1	0.9996	1.000000	0.99929	0.999647	0.999461	0.999937	0.9991	0.999527
log_reg_l2	0.9996	1.000000	0.99929	0.999647	0.999461	0.999937	0.9991	0.999527
rfc_imp	0.975063	0.958906	0.99891	0.978502	0.975526	0.959784	0.9989	0.978965
log_reg_rfe	0.9996	1.000000	0.99929	0.999647	0.999497	1.000000	0.9991	0.999559
log_reg_ffe	0.9995	-	-	-	0.9994	-	-	-

log_reg_bfe	0.87948	-	-	-	0.8776	-	-	-
bagging_clf_d_tree	0.975171	0.958127	1.00000	0.978616	0.975418	0.958666	1.0000	0.978897
bagging_clf_svm_linear	0.965113	0.957042	0.98270	0.969703	0.963414	0.957701	0.9790	0.968269
rf_gini_sqrt	0.994856	0.991105	0.99991	0.995492	0.995148	0.991746	0.9998	0.995762
rf_gini_log2	0.992607	0.987780	0.99934	0.993531	0.992812	0.988645	0.9988	0.993729
rf_entropy_log2	0.994224	0.990568	0.99934	0.994939	0.994501	0.991552	0.9988	0.995195
rf_entropy_sqrt	0.993731	0.989165	0.99991	0.994513	0.994322	0.990322	0.9998	0.995044
ext_classifier	0.924589	0.952628	0.91264	0.932209	0.921689	0.952277	0.9081	0.929694
ext_classifier_entropy_log2	0.975571	0.958774	1.00000	0.978953	0.975957	0.959591	0.9999	0.979349
ada_boost_clf_d_tree	0.99578	0.992706	0.99991	0.996299	0.995903	0.992989	0.9998	0.996419
ada_boost_clf_nb	0.931412	0.892278	1.00000	0.943073	0.930386	0.891186	1.0000	0.942462
grad_boost_clf	0.999969	1.000000	0.99994	0.999973	0.999856	1.000000	0.9997	0.999874
grad_boost_clf_nb	0.684693	0.848404	0.54181	0.661306	0.688338	0.849330	0.5511	0.668476
stacking_clf	0.997258	0.995893	0.99929	0.997591	0.997736	0.996918	0.9991	0.998017

voting _clf	0.9996	1.000000	0.99929	0.999647	0.999497	1.000000	0.9991	0.999559
voting _clf_soft	0.994624	0.990627	1.00000	0.995291	0.99522	0.991686	1.0000	0.995826

Graphs:

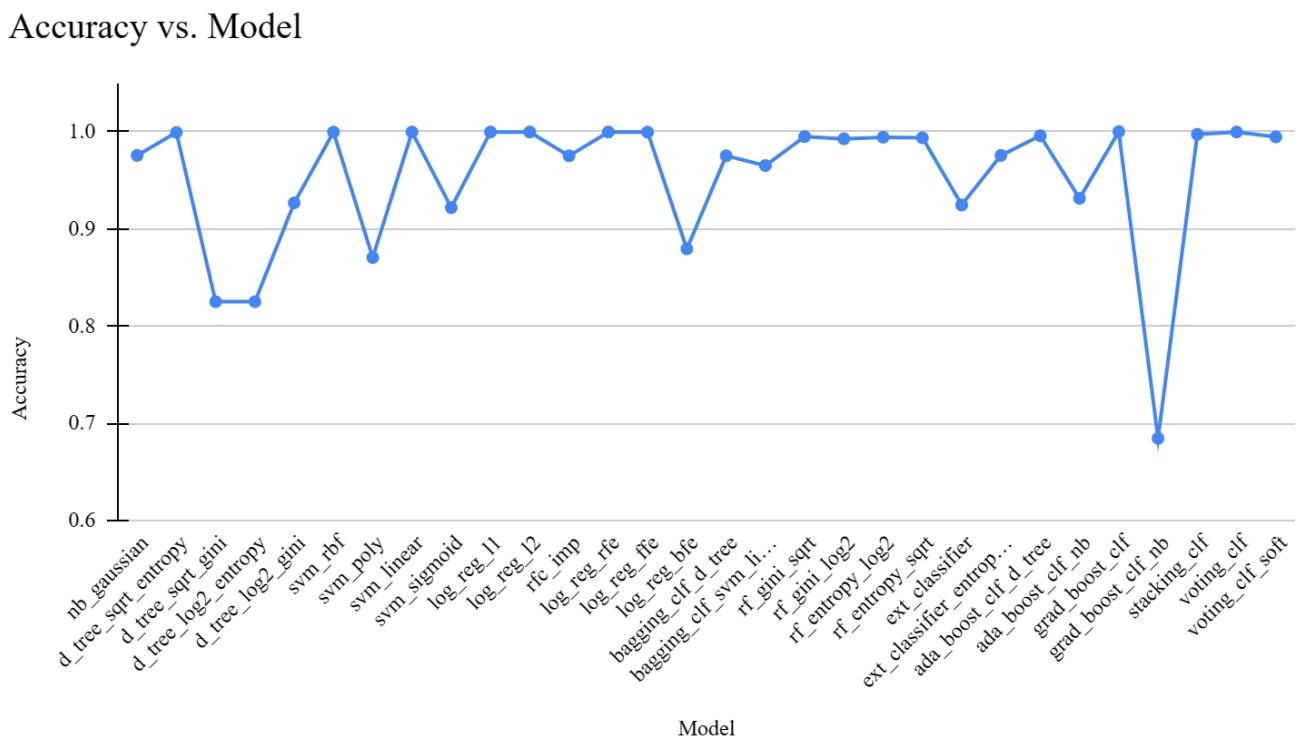

Fig No. 2: Training Accuracy of all Algorithms

In above graph we have plotted all the training accuracies of all the models. It is clear that most of the models have high accuracy, although some have pretty low accuracy.

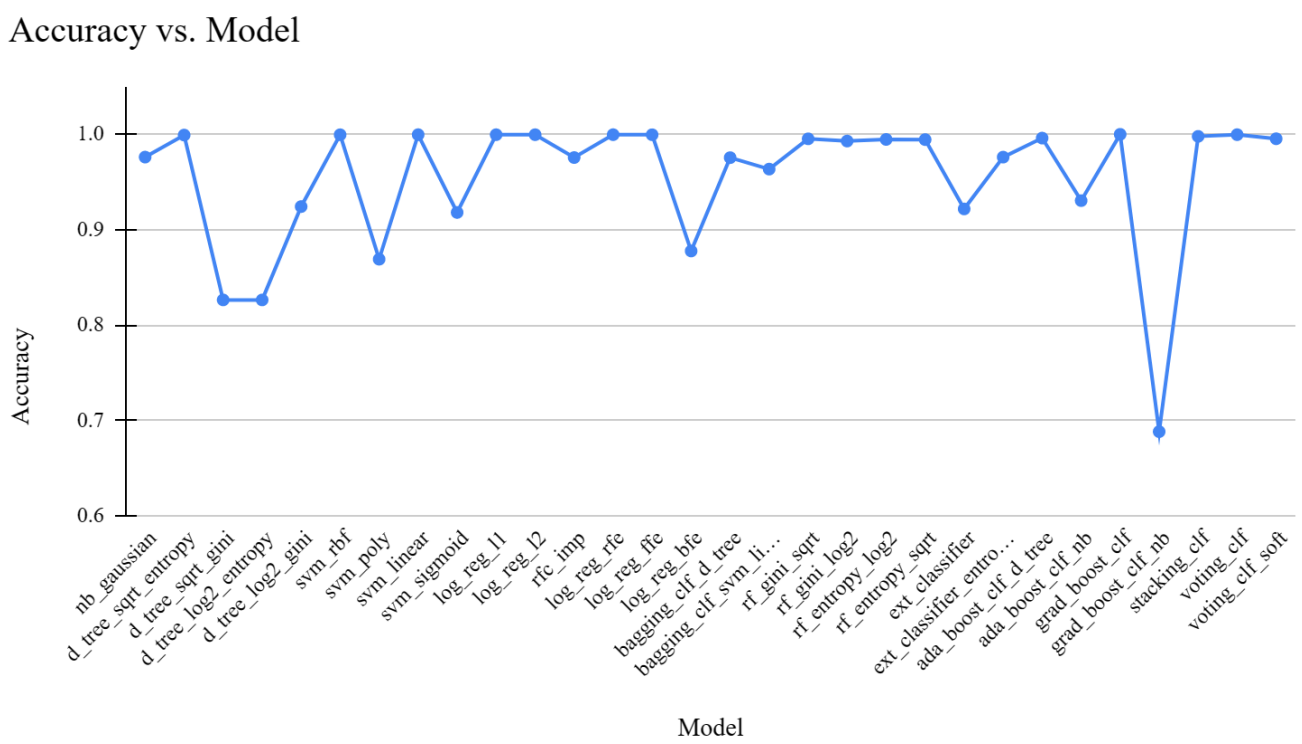

Fig No. 3: Testing Accuracy of all Algorithms

In above graph we have plotted all the testing accuracies of all the models. It is clear that most of the models have high accuracy, although some have pretty low accuracy.

4. RESULTS

	Training				Testing			
Model	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
svm_linear	0.9996	1.000000	0.99929	0.999647	0.999497	1.000000	0.9991	0.999559
d_tree_sqrt_entropy	0.999168	0.998619	0.99991	0.999268	0.999066	0.998552	0.9998	0.999181
log_reg_rfe	0.9996	1.000000	0.99929	0.999647	0.999497	1.000000	0.9991	0.999559
rf_entropy_sqrt	0.993731	0.989165	0.99991	0.994513	0.994322	0.990322	0.9998	0.995044
nb_gaussian	0.975525	0.958699	1.00000	0.978914	0.975993	0.959594	1.0000	0.979380
grad_boost_clf	0.9999	0.999946	0.99989	0.999919	0.9999	0.999874	0.99993	0.999905
log_reg_l1	0.9996	1.000000	0.99929	0.999647	0.999461	0.999937	0.9991	0.999527
voting_clf	0.9996	1.000000	0.99929	0.999647	0.999497	1.000000	0.9991	0.999559

Graphs:

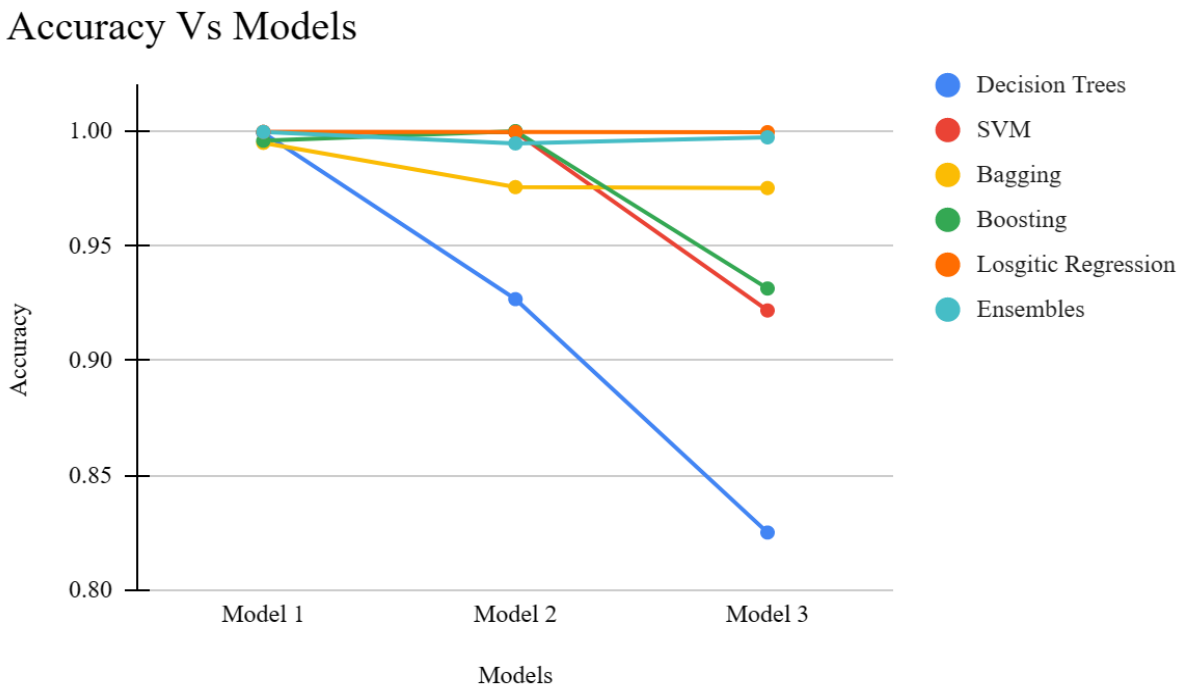

Fig No. 4: Models of highest training accuracy of each type

In above we have selected the top three models having highest training accuracy of each type of algorithm and plotted. It can be seen that models of Logistic Regression has highest accuracy for all three models.

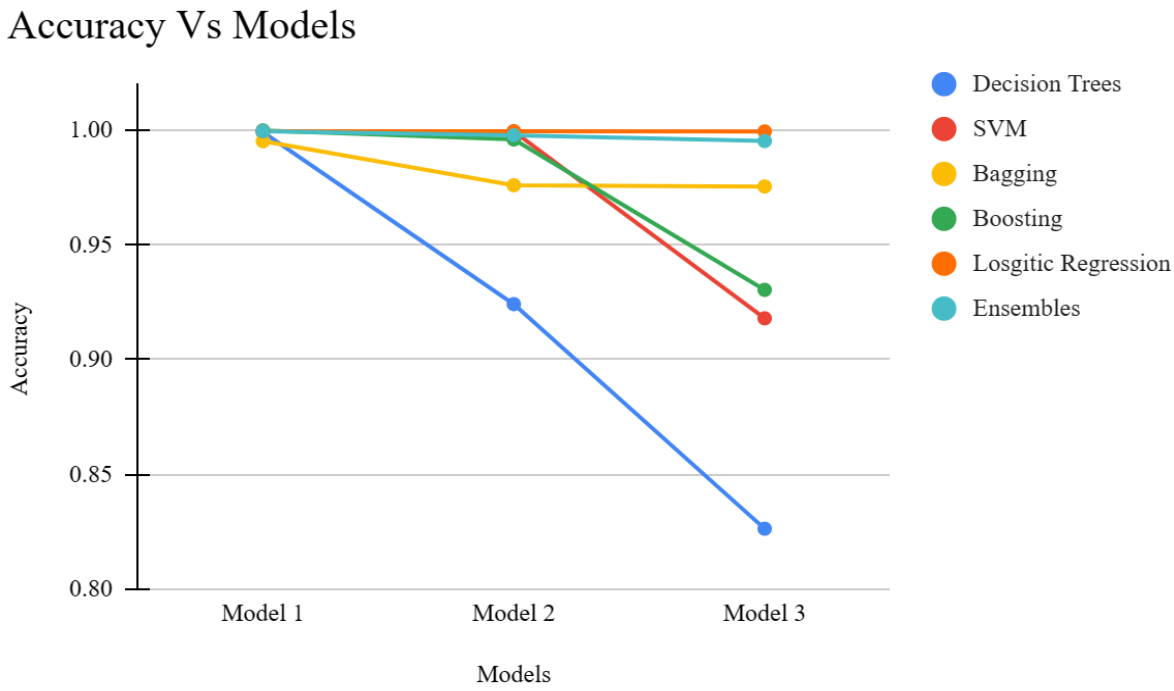

Fig No. 5: Models of highest testing accuracy of each type

In above we have selected the top three models having highest testing accuracy of each type of algorithm and plotted. It can be seen that models of Logistic Regression has highest accuracy for all three models.

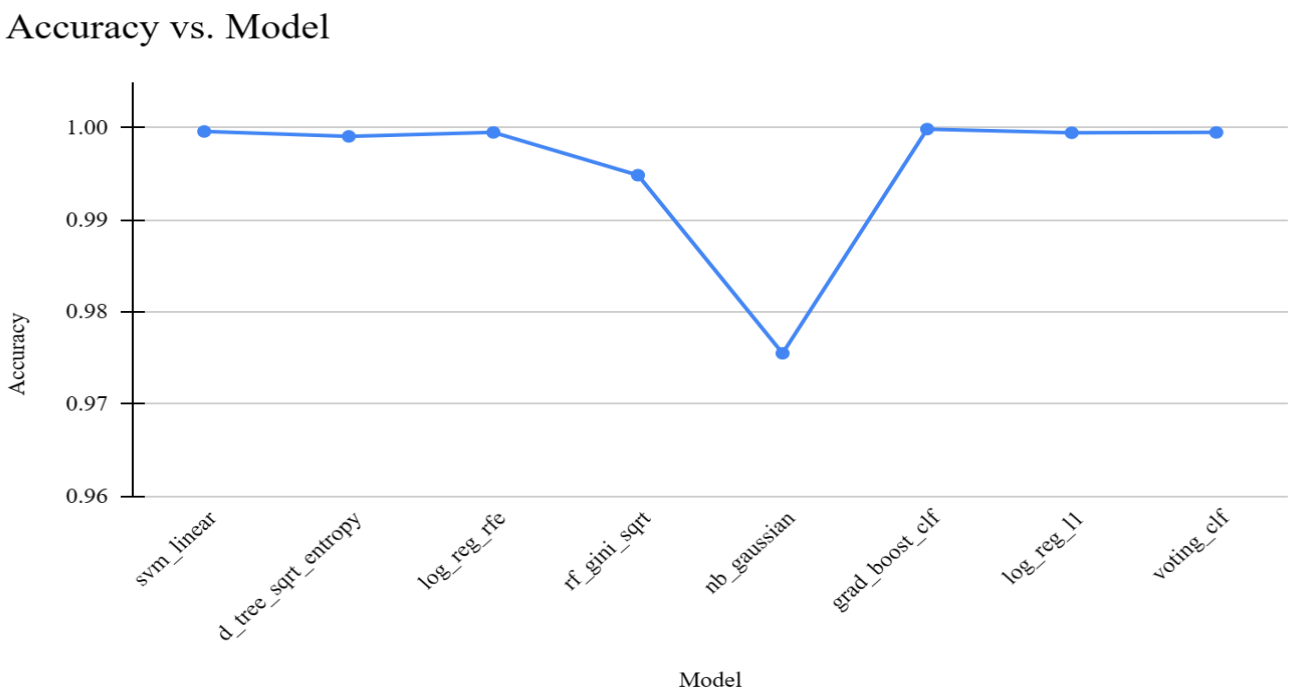

Fig No. 6: Models of highest training accuracy

In the above figure algorithms one model having highest training accuracy is selected from each type of algorithm and graph is plotted. svm_linear, log_reg_rfe, log_reg_l1 and voting_clf have very high and similar accuracy while nb_gaussian has the lowest accuracy.

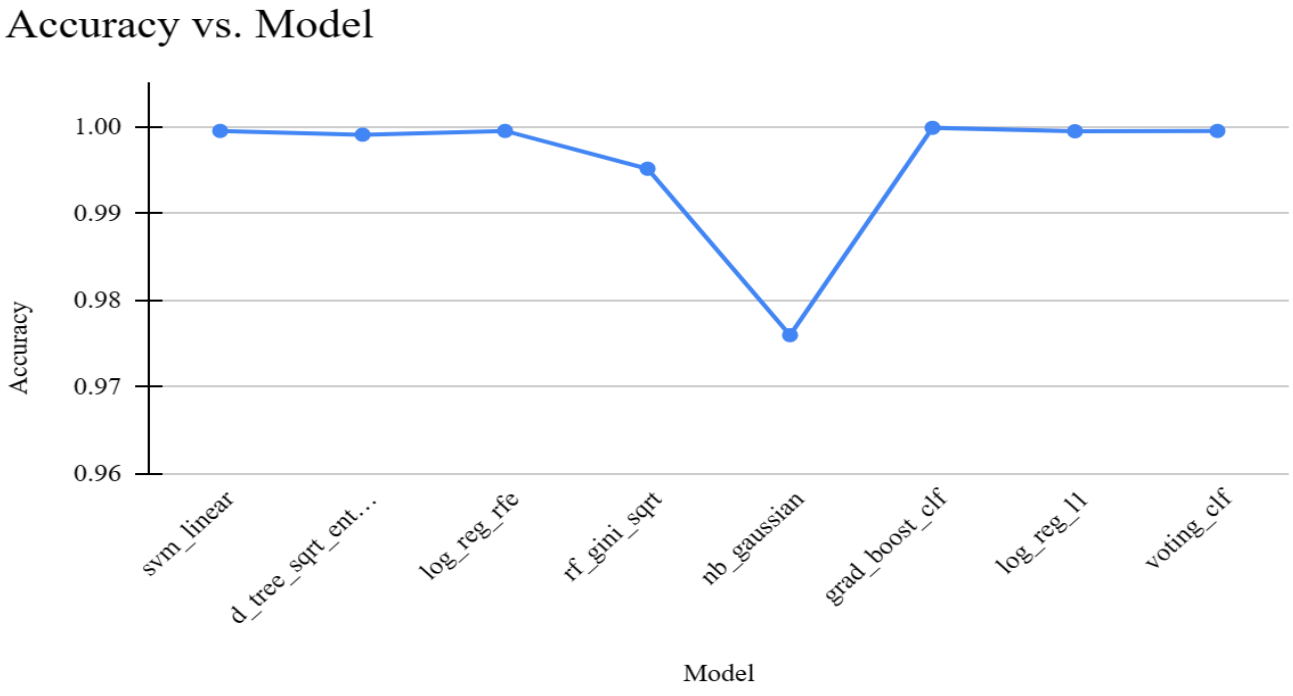

Fig No. 7: Models of highest testing accuracy

In above figure algorithms one model having highest testingaccuracy is selected from each type of algorithm and graph is plotted. svm_linear, log_reg_rfe, log_reg_l1 and voting_clf have very high and similar accuracy while nb_gaussian has the lowest accuracy.

5. CHALLENGES

1. The main challenge was to prepare a dataset from the huge amount of data.
2. To select an appropriate method for handling missing values.
3. To reduce features using the proper methods.

6. FUTURE SCOPE

We are very keen to improve the future work of the IDS. We would like to implement an attack detection model, especially for low frequency attacks by exploring the deep leaning approaches. Later, when IDS approaches are utilised in a dynamic and changing network environment like cloud computing and other similar environments, a variety of difficulties will be prioritised. Currently, this project is only software based, later the model can be deployed and combined with hardware, thus fetching inputs in realtime and giving outputs in real time

7. CONCLUSION

Intrusion Detection System is undergoing lost of enhancement since past decades. It is important for the organization to implement the Intrusion Detection System. IDS Technology doesn't need human interventions. The very important aspect of IDS is designing and implementation phase. In most cases, it is desirable to implement a hybrid for a network based IDS. The decision can vary between organizations depending on the requirements. We have studied and preprocessed the dataset. Performance analysis of various machine learning algorithms has been done. We have found that, normal algorithms give great accuracy but when combined using bagging/boosting/ensemble techniques the accuracy increases even more, with boosting and voting ensemble found to be most accurate and efficient method to combine all algorithms and detect intrusion. Gradient Classifier Boost has given the best test accuracy among all the algorithms 99.985 %.

8. REFERENCES

1. L. Ashiku and C. H. Dagli, "Network Intrusion Detection System using Deep Learning," Procedia Computer Science, vol. 185, pp. 239 - 247, Elsevier B. V., Jun 2021
2. Maithem, Mohammed Ali and Ghadaa A. Al-sultany. "Network intrusion detection system using deep neural networks." Journal of Physics: Conference Series 1804 (2021): n. pag.
3. Mishra, Preeti et al. "A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection." IEEE Communications Surveys & Tutorials 21 (2019): 686-728.
4. Gendreau, A.A., & Moorman, M.E. (2016). Survey of Intrusion Detection Systems towards an End to End Secure Internet of Things. 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), 84-90.
5. D. Jing and H. -B. Chen, "SVM Based Network Intrusion Detection for the UNSW-NB15 Dataset," 2019 IEEE 13th International Conference on ASIC (ASICON), 2019, pp. 1-4, doi: 10.1109/ASICON47005.2019.8983598.
6. Nehra, Divya & Mangat, Veenu & Saluja, Krishan. (2021). A Deep Learning Approach for Network Intrusion Detection Using Non-symmetric Auto-encoder. 10.1007/978-981-16-1295-4_38.
7. G. Karatas, O. Demir and O. Koray Sahingoz, "Deep Learning in Intrusion Detection Systems," 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), 2018, pp. 113-116, doi: 10.1109/IBIGDELFT.2018.8625278.
8. Solane Duque, Mohd. Nizam bin Omar, Using Data Mining Algorithms for Developing a Model for Intrusion Detection System (IDS), Procedia Computer Science, Volume 61, 2015, Pages 46-51, ISSN 1877-0509
9. I. Sumaiya Thaseen and C. Aswani Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," J. King Saud Univ. - Comput. Inf. Sci., vol. 29, no. 4, pp. 462–472, 2017
10. W.C. Lin et al., CANN: An intrusion detection system based on combining cluster centers and nearest neighbors, Knowl. Based Syst. (2015)
11. Ravale, Ujwala & Marathe, Nilesh & Padiya, Puja. (2015). Feature Selection Based Hybrid Anomaly Intrusion Detection System Using K Means and RBF Kernel Function. Procedia Computer Science. 45. 428-435. 10.1016/j.procs.2015.03.174.
12. Alhakami, Wajdi & Alharbi, Abdullah & Bourouis, Sami & Alroobaea, Roobaea & Bouguila, Nizar. (2019). Network Anomaly Intrusion Detection Using a Nonparametric Bayesian Approach and Feature Selection. IEEE Access. 7. 52181-52190. 10.1109/ACCESS.2019.2912115.
13. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," in IEEE Access, vol. 7, pp. 41525-41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

9. ACRONYMS

Sr. No.	Short form	Description
1	nb_gaussian	Gaussian Navie Bayes.
2	d_tree_sqrt_entropy	Decision Tree, where the split function is entropy and no. of features while splitting is sqrt(n_features).
3	d_tree_sqrt_gini	Decision Tree, where the split function is gini and no. of features while splitting is sqrt(n_features).
4	d_tree_log2_entropy	Decision Tree, where the split function is entropy and no. of features while splitting is log2(n_features).
5	d_tree_log2_gini	Decision Tree, where the split function is gini and no. of features while splitting is log2(n_features).
6	svm_rbf	Support Vector Machine, where kernel function is RBF(Radial Basis Function).
7	svm_poly	Support Vector Machine, where kernel function is Polynomial.
8	svm_linear	Support Vector Machine, where kernel function is Linear.
9	svm_sigmoid	Support Vector Machine, where kernel function is Sigmoid.
10	log_reg_l1	Logistic Regression, where the penalty is 'l1'.
11	log_reg_l2	Logistic Regression, where the penalty is 'l2'.
12	rfc_imp	Random Forest Importance.
13	log_reg_rfe	Logistic Regression with Recursive Feature Elimination.
14	log_reg_ffe	Logistic Regression with Forward Feature Elimination.
15	log_reg_bfe	Logistic Regression with Backward Feature Elimination.
16	bagging_clf_d_tree	Bagging Classifier, where base estimator is d_tree_srqt_entropy.
17	bagging_clf_svm_linear	Bagging Classifier, where base estimator is svm_linear.
18	rf_gini_sqrt	Random Forest Classifier, where the split function is gini and no. of features while splitting is sqrt(n_features).
19	rf_gini_log2	Random Forest Classifier, where the split function is gini and no. of features while splitting is log2(n_features).
20	rf_entropy_log2	Random Forest Classifier, where the split function is entropy and no. of features while splitting is log2(n_features).
21	rf_entropy_sqrt	Random Forest Classifier, where the split function is entropy and no. of features while splitting is sqrt(n_features).
22	ext_classifier	Extra Tree Classifier, where the split function is gini and no. of

		features while splitting is sqrt(n_features).
23	ext_classifier_entropy_log2	Extra Tree Classifier, where the split function is entropy and no. of features while splitting is log2(n_features).
24	ada_boost_clf_d_tree	AdaBoost Classifier, where the base estimator is d_tree_sqrt_entropy.
25	ada_boost_clf_nb	AdaBoost Classifier, where the base estimator is nb_gaussian.
26	grad_boost_clf	GradientBooosting Classifier, where initial estimator is d_tree_sqrt_entropy.
27	grad_boost_clf_nb	GradientBooosting Classifier, where initial estimator is nb_gaussian.
28	stacking_clf	Stacking Classifier, where the final estimator is log_reg_l1.
29	voting_clf	Voting Classifier, where the voting is hard.
30	voting_clf_soft	Voting Classifier, where the voting is soft.