AN ENSEMBLE FRAMEWORK FOR NETWORK INTRUSIONS

SUBMITTED BY

V. KANIMOZHI (20PCA004)

Project Report Submitted

In partial fulfillment of the requirements for the Award of

MASTER OF COMPUTER APPLICATIONS

Department of Computer Science

Avinashilingam Institute for Home Science and Higher Education for Women

Coimbatore – 641043

May – 2022

ABSTRACT

The amount of data that moves through a network at any particular time is referred to as network traffic. Data traffic or just plain traffic are other terms for network traffic. The global network traffic analysis market is predicted to increase at a compound yearly growth rate of 9.7% from 2021 to 2028, reaching USD 5.69 billion by 2028, according to a report by Grand View Research. The COVID-19 pandemic outbreak and the accompanying lockdowns and limitations enforced in many parts of the world have had a minor influence on network traffic analysis.

Aims: To propose an ensemble learning framework to detect the different attack types. This project deals with the development of supervised machine learning algorithms to detect anomalies in network traffic from the CIC-IDS2018 dataset. **Method:** The detection of anomalies in network traffic using a supervised machine learning approach comprises five phases. Phase 1 is Data Acquisition. In Phase 2 is the Data Preprocessing method, which transforms the dataset and resamples the majority and minority of attacks on the dataset (CIC-IDS2018). In Phase 3, embedded-based featureselection methods are used to select the important features. In Phase 4, we discuss the supervised machine learning models developed with ensemble methods such as bagging (Random Forest, Decision Tree, Bagging Classifier) and boosting (Adaptive Boosting, Extreme Gradient, Light Gradient Boosting, Histogram-based gradient boosting) and then they are evaluated. The output of different algorithms is evaluated in phase 5 with performance measures such as precision, recall, F1 Score, and accuracy score. It is observed that some models give better accuracy than others, and the entire project is developed on the Python platform. **Results:** From this proposed system, the best accuracy was obtained using the first method of Random Forest Feature Selection with the Bagging method, the decision tree model was obtained with a 96% accuracy score. In the Boosting method, the Light Gradient Boosting model and the Histogram-based gradient boosting model both have a 96% accuracy score. By using the second method of Gradient Boosting Feature Selection with the Boosting method, the highest accuracy was obtained in the Light Gradient Boosting model and Histogram-based gradient boosting model with 96% accuracy score. These are ensemble methods and models that have better detection rates for multi-class attack classifications.

Keywords: **Ensemble Learning, Intrusion Detection, Multi-class Classification, Machine Learning, Network Traffic.**

1. INTRODUCTION

1.1 About the Project

In anomaly-based Network Intrusion Detection Systems (NIDS), machine learning techniques have been extensively used. Diverse machine learning algorithms, including bagging (Random Forest Classifier, Decision Tree Classifier, Bagging Classifier) and boosting (AdaBoost Classifier, Extreme Gradient Boosting Classifier, Light Gradient Boosting Classifier, Histogram-based gradient boosting classifier), are used to detect intrusions.

However, single machine learning algorithms are no longer sufficient to meet the extensive requirements of modern intrusion detection systems. Due to high traffic volume, IDS attacks have diversified and are way more sophisticated. Some models can perform well on one type of attack but perform poorly on other types of attacks. Hence, a machine learning model with multi-class classification for identifying different attacks is the need of the hour.

An ensemble learning-assisted intrusion detection system is developed in this project. Ensemble framework-based IDS mostly aims at one-class or a subset of class classification. The proposed IDS has high performance at multi-class classification, for identifying different attack classes in a dataset. The Canadian Institute for Cybersecurity Intrusion Detection System (CICIDS2018) dataset is used in this project. It is a famous dataset with a larger number of features.

2. METHODOLOGY

A single machine learning classifier cannot be used to detect different attacks effectively. An Ensemble approach is used to combine the results of different classifiers. In ensemble-based methods, predictions are the results of the major of among the contributing models. In ensemble-based methods, all classifiers in the ensemble contribute to the final output regardless of whether the algorithm is capable of detecting the attack or not. Thus, to overcome the above drawback and increase the attack detection rate for multi-attack classification, It is proposed to develop an ensemble framework for attack detection. In this approach, several machine learning classifiers are ranked based on their efficiency in detectingvarious attacks.

The following section elaborates on the workflow of building the machine learning model that is used as a testbed. The machine learning model contained three major stages: preprocessing data, sampling the majority and minority labels, and classifying target data. Thedetailed activities in each stage are depicted in Figure 4.1

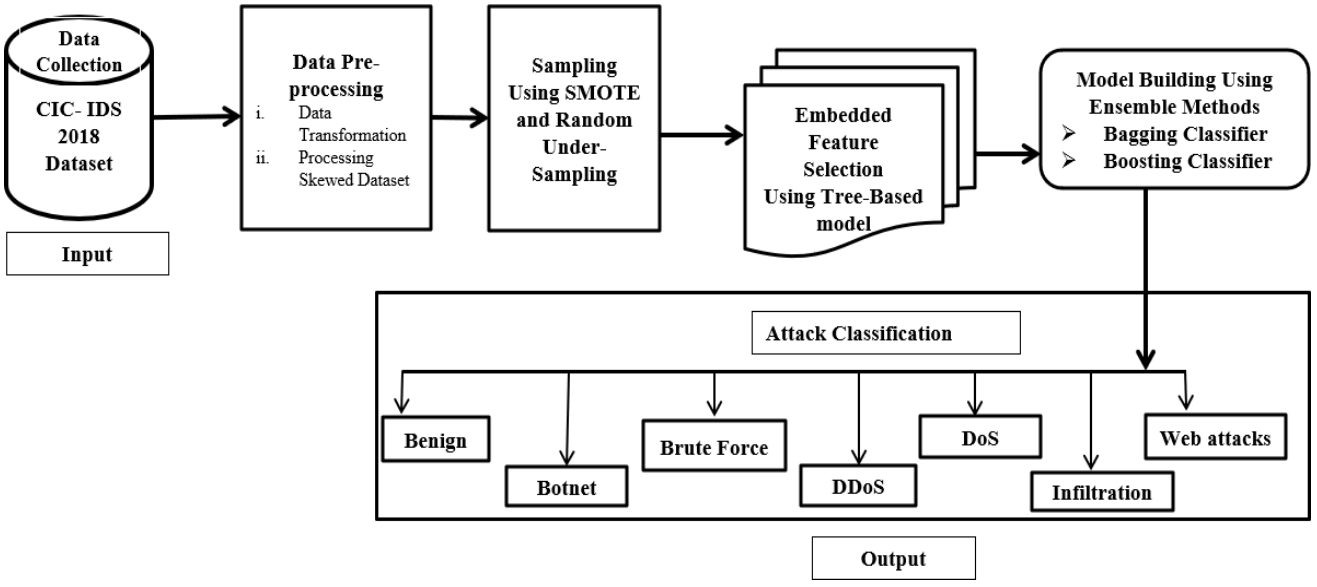

Figure 2.1 : Proposed Methodology

2.1 Data Collection

The process of Data collection is collecting datasets from the relevant sources it is used to test and evaluate the proposed system, In this project CIC IDS2018 Dataset is used. It is provided by Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). This dataset is generated for intrusion detection systems, mainly focusing on network-basedanomaly detection. CSE-CIC-IDS2018 dataset is originally separated into 10 CSV files based on different types of network attacks, including FTP-Brute Force, SSH-Brute Force, Brute Force -Web,Brute Force -XSS, SQL Injection, Infiltration, Bot, DoS-Golden Eye, DoS-Slowloris, DoS-Slow HTTP Test, DoS-Hulk, DDoS attacks-LOIC-HTTP, DDoS-LOIC-UDP, and DDOS attack-HOIC.

The attacking infrastructure includes 50 machines and the victim organization has 5 departments and includes 420 machines, 30 servers with 80 features and approximately sixteen million rows are present in this dataset.

Pandas library provides functions to concatenate and import multiple .csv files into one data frame.

This chapter first introduces common network attacks including brute-force attacks, botnet attacks, web attacks, and infiltration attacks. All listed attacks are explained in terms of characteristics and detection techniques. This chapter also discusses three feature selection methods: filter method, wrapper method, and embedded method. The differences between the three kinds of feature selection methods are explained.

Network Attacks

Network attacks are further classified into types presented as follows,

- Brute-Force
- Botnet
- Web Attacks
- Infiltration
- Denial-of-Service
- Distributed Denial-Of-Service
- Heartbleed Attack

a) Brute-Force

The brute-force attack has been used frequently within the network. However, it is not difficult to detect this type of attack. The intrusion detection system could effectively detect the exhaustive key search-based brute-force attack by setting the utmost number of login attempts. The exhaustive key search attack requires the hackers to undertake all the potential combinations of passwords within the key space until the correct key is found.

When the maximum number of login attempts is reached, the intrusion detection system will detect the abnormal activity and raise the alarm. Therefore, the account would be locked to prevent further attempts by hackers. However, the password-spray attack overcomes the disadvantage of the exhaustive key search. It allows the hackers to repeatedly try the identical password for various accounts to avoid being alerted by the utmost attempts policy that would be activated by multiple unsuccessful login attempts on the identical account. However, the advance of password-spray attacks makes it difficult to detect. There is another approach that was introduced to detect brute force attacks.

b) Botnet

A common approach for detecting botnet attack is to investigate the network traffic. The typical attributes of network traffic include the source and destination address, the protocols, the related port numbers, the duration of the session, the amount of incoming and outgoing packets, andthe cumulative size of the transmitted packets. The detector usually checks the IP addresses of network traffic to determine if the IP addresses are identified as suspicious addresses with abnormalactivities before.

The bots, the compromised computers, often use the particular protocol and associated port toascertain the communication with one another or infect other target computers. For instance, port 25 for SMTP, port 80 for HTTP, port 6667 for IRC, and port 53 for DNS. Counting the packets transmitted within the network and their size shows the significant meaning for detecting the communications between bots or between bot and target. It could indicate the weird behavior of the botnet attack by showing the huge amount of information transmission within a comparatively in need of time, multiple packets of identical size, and many more.

c) Web Attacks

The development of online applications makes the web more functional and productive. People could access many software applications through the web and store the info online. The common web applications that individuals may use daily are online office suites, online video streaming platforms, webmail, e-commerce platform, etc.

But the sensitive data of users handled by these web applications could also be threatened by the net attacks. Web attacks aim to interrupt the safety of web services by using the weakness of the programs and hijacking the information traffic between the client and server.

There are two kinds of common web attacks,

- XSS attack
- SQL injection

XSS attack

An XSS attack is targeting to take advantage of the communication between the top users and web applications through the malicious script. The XSS attack may be launched to get users' cookie information by injecting malicious scripts into the HTML content of an online page. When the user accesses this website, the malicious script would be activated and steal the user's cookie. The cookie information may reveal users' sensitive information, like username and password.

SQL injection

SQL injection allows the hackers to manage the SQL database that is connected to the net application by injecting a SQL query through user input. The hackers discover the vulnerable user input of an online page that directly uses user input in SQL queries. After changing the SQL code, the hackers may retrieve, modify, or even delete the data within the database.

d) Infiltration

By observing the software released on the net, none of them is perfectly secure without any vulnerability. However, the hackers may use the weaknesses of the software to interrupt the victims' computers and launch attacks from within the interior network. The hackers may target various software, including document viewers, application programs, program development tools, and many more. The malware will infect users' computers through the vulnerable software. Then the hackers launch the port scanning attack to find other potential victims within the internal network.

The port scanning attacks identify the potential victims by sending packets to the hosts within the network through every possible port. The status of the ports is going to be acknowledged by checking the responses. They are either open or closed. The response packets also contain the target information, for example, the IP address and protocol. Leaked information is also employed by hackers to launch more network attacks.

Even though there are many types of port scanning attacks, they all attack similar targets. The common features that are accustomed to detecting port scanning attacks contain protocols, port numbers, IP addresses, and TCP flags. The port scanning attacks will be employed on transport layer protocols including TCP, UDP, ICMP, and IP. They will be performed against a single port from multiple IPs, and multiple ports will be scanned by one IP further. Since TCP connections are established during the attack and different TCP flags are used for various types of port scanning attacks, TCP flags become one of the special characteristics of the port scanning attack.

e) Denial-of-Service

Detection of Denial of Service This is often addressed in flow-based intrusion detection. Because of their nature, these attacks can cause variations in traffic volume that are usually visible at flow scale.

There are several attack categories under denial-of-service attacks, which are,

- DoS-Slowloris
- DoS-Golden Eye
- DoS-Slow HTTP Test
- DoS-Hulk

DoS-Slowloris

Slowloris attacks are also referred to as Slow GET or Slow Header. This attack allows a single machine to take down another machine's web server while using minimal bandwidth and causing unassociated services and ports to fail. In this scenario, it uses a Slowloris Perl-based instrument to request the offline server.

The HTTP get request is routed to the specified server. This request cannot be validly terminated as it lacks the terminating character r\n\r\n (double line break). The server anticipates the next request, which will include a destroying character. The server configuration limits this waiting. The server terminates the TCP connection when the timer expires.

DoS-Golden Eye

GoldenEye is a DoS testing tool for HTTP/S Layer. Keep Alive (as well as Connection: keep-alive) options, in conjunction with Cache-Control options, are used to keep the socket connection smashing via caching until it consumes all obtainable sockets on the HTTP/S server. GoldenEye is a capable and highly recommended technique to investigate malware-related issues. It can detect malware's condition and sensitive quirks in advanced running and select the malware's likely targeted settings. It can also switch its framework conditions online adaptively to promote testing.

GoldenEye can effectively determine the malware's intended environment by using a specific conjectural implementation engine to observe malware practices in elective situations.

DoS-Slow HTTP Test

- Slow HTTP Test is really a highly customizable tool for simulating application-level-level Denial of Service attacks (DoS attacks).
- Slowloris is one of the most common low-bandwidth application level Denial of Service attacks.
- HTTP POST is slow.
- Slow Read attack (based on TCP persist timer exploit) by draining concurrent connections pool. Apache Range Header attack by causing extremely high memory and CPU usage on the server.

DoS-Hulk

- HULK is a Denial of Service (DoS) program that targets web servers by creating disguised and unique traffic volumes.
- HULK traffic skips caching engines and goes straight to the server's direct pool of resources.

f) Distributed Denial-Of-Service

Distributed Denial of Service (DDoS), that targets at server within the volume of useless traffic from distributed and coordinated attack sources, are a major threat to the stability of the Internet.

Several types of Distributed Denial-Of-Service are,

- DDOS attack-HOIC
- DDoS attacks-LOIC-HTTP
- DDoS-LOIC-UDP

DDoS attack-HOIC

- The High Orbit Ion Cannon (HOIC) is a BASIC-based network training and denial-of-service attack application that can simultaneously attack up to 256 URLs.
- Praetox Technologies' Low Orbit Ion Cannon will be replaced by it. In this example, the HOIC framework is utilised to launch a DDoS attack from four different workstations.

g) Heartbleed Attack

One of the foremost famous tools to need the advantage of Heartbleed is Heart leech. It can scan forsystems in danger of the bug, and might then be accustomed to exploiting them and exfiltrating data.

Some important features:

- Conclusive/inconclusive verdicts on whether the target is vulnerable
- Automatic retrieval of personal keys with no additional steps
- Some limited IDS evasion
- IPv6 support
- Bulk/fast download of Heartbleed data into an oversized file for offline processing usingmany threads
- Tor/Socks5n proxy support
- Extensive connection diagnostic information
- STARTTLS support

2.1.1 Dataset Description

Dataset description describes the features present in it, in 2.1 tabular column gives elaborate explanation of features.

Table 4.1 Feature and Feature description of the CIC-IDS2018 dataset

S.No	Feature Name	Description
1.	FL_DUR	Duration of the flow
2.	TOT_FW_PK	Total number of packets sent in the forward direction
3.	TOT_BW_PK	Total packets in the reverse direction
4.	TOT_L_FW_PKT	Total packet size in the forward direction
5.	FW_PKT_L_MAX	Maximum packet size in the forward direction
6.	FW_PKT_L_MIN	Minimum packet size in the forward direction
7.	FW_PKT_L_AVG	Average packet size in the forward direction
8.	FW_PKT_L_STD	Packet size standard deviation in the forward direction
9.	BW_PKT_L_MAX	Maximum packet size in the reverse direction
10.	BW_PKT_L_MIN	Minimum packet size in the reverse direction
11.	BW_PKT_L_AVG	Mean packet size in the reverse direction
12.	BW_PKT_L_STD	Packet size standard deviation in the reverse direction
13.	FL_BYT_S	flow packets rate, which is the number of packets transferred per second
14.	FL_PKT_S	flow byte rate, which is the number of packets transferred per second
15.	FL_IAT_AVG	Time difference between two flows
16.	FL_IAT_STD	Time two standard deviation flows
17.	FL_IAT_MAX	The longest possible time between two flows

18.	FL_IAT_MIN	The shortest possible time between two flows
19.	FW_IAT_TOT	Total time elapsed between two packets sent forward
20.	FW_IAT_AVG	The average time between two packets sent forward
21.	FW_IAT_STD	Time difference between two packets sent in the forward direction
22.	FW_IAT_MAX	The longest possible time between two packets sent in the forward direction.
23.	FW_IAT_MIN	The shortest possible time between two packets sent in the forward direction.
24.	BW_IAT_TOT	Total time elapsed between two packets sent in the reverse direction
25.	BW_IAT_AVG	The average time between two packets sent in the opposite direction.
26.	BW_IAT_STD	Time difference between two packets sent in the reverse direction
27.	BW_IAT_MAX	Maximum time interval between two packets sent in the reverse direction
28.	BW_IAT_MIN	The shortest possible time between two packets sent in the reverse direction.
29.	FW_PSH_FLAG	The number of times the PSH flag was set in packets travelling forward (0 for UDP)
30.	BW_PSH_FLAG	The number of times the PSH flag was set in packets travelling backward (0 for UDP)
31.	FW_URG_FLAG	The number of times the URG flag has been set in forward-moving packets (0 for UDP)
32.	BW_URG_FLAG	The number of times the URG flag was set in packets that were travelling backward (0 for UDP)
33.	FW_HDR_LEN	In the forward direction, total bytes used for headers
34.	BW_HDR_LEN	In the forward direction, total bytes used for headers

35.	FW_PKT_S	Number of packets forwarded per second
36.	BW_PKT_S	Number of packets sent backwards per second
37.	PKT_LEN_MIN	A flow is minimum length
38.	PKT_LEN_MAX	A flow is maximum length
39.	PKT_LEN_AVG	The average length of a flow
40.	PKT_LEN_STD	The length of a flow's standard deviation
41.	PKT_LEN_VA	Packet inter-arrival time must be kept to a minimum.
42.	FIN_CNT	The total number of FIN packets
43.	SYN_CNT	Packets with SYN (number of packets)
44.	RST_CNT	The number of RST packets
45.	PST_CNT	Number of packets with PUSH
46.	ACK_CNT	The number of packets with an ACK
47.	URG_CNT	The total number of URG packages
48.	CWE_CNT	The number of packets containing CWE
49.	ECE_CNT	The number of ECE packets
50.	DOWN_UP_RATIO	The ratio of downloads to uploads
51.	PKT_SIZE_AVG	Average size of packet
52.	FW_SEG_AVG	Average packet size Average packet size in the forward direction
53.	BW_SEG_AVG	Average packet size in the backward direction
54.	FW_BYT_BLK_AVG	In the forward direction, the average number of bytes bulk rate

55.	FW_PKT_BLK_AVG	In the forward direction, the average number of packets bulk rate
56.	FW_BLK_RATE_AVG	In the forward direction, average number of bytes bulk rate
57.	BW_BYT_BLK_AVG	Average number of bytes bulk rate in the backward direction
58.	BW_PKT_BLK_AVG	In a backward direction, the average number of packets bulk rate
59.	BW_BLK_RATE_AVG	In a backwards approach, the average number of bulk rate
60.	SUBFL_FW_PK	In a forward direction sub flow, the average number of packets.
61.	SUBFL_FW_BYT	In the forward direction, the average number of bytes in a sub flow
62.	SUBFL_BW_PKT	In the backward direction, the average number of packets in a sub flow
63.	SUBFL_BW_BYT	In the backward direction, the average number of bytes in a sub flow
64.	FW_WIN_BYT	In the forward direction, the number of bytes sent in the initial window
65.	BW_WIN_BYT	In the backward direction, the number of bytes sent in the initial window
66.	FW_ACT_PKT	of packets in the forward direction with at least 1 byte of TCP data payload
67.	FW_SEG_MIN	In the forward direction, the smallest segment size was recorded
68.	ATV_AVG	Before being idle, a flow spent the majority of its time busy.
69.	ATV_STD	The standard deviation of how long a flow was active before it became idle.
70.	ATV_MAX	The longest a flow was operating before it became idle.
71.	ATV_MIN	Before a flow became idle, it had to be active for at least a certain amount of time.
72.	IDL_AVG	Before being active, a flow spent the majority of its time idle.
73.	IDL_STD	The standard deviation of how long a flow was idle before it became active
74.	IDL_MAX	The longest a flow could be idle before becoming active
75.	IDL_MIN	Before a flow became active, it had to be idle for a certain amount of time

2.1.2 Data Transformation

The CIC-IDS 2018 dataset is a comprehensive dataset comprising various modern-day attacks. The 14 attack classes are present in the CIC IDS 2018 dataset. The 14 attack classes are reclassified into six classes based on the nature of the attack– Bot, Brute Force, DDoS, DoS, Infiltration, and Web attacks. The final dataset comprises benign samples and samples from the above six attack categories.

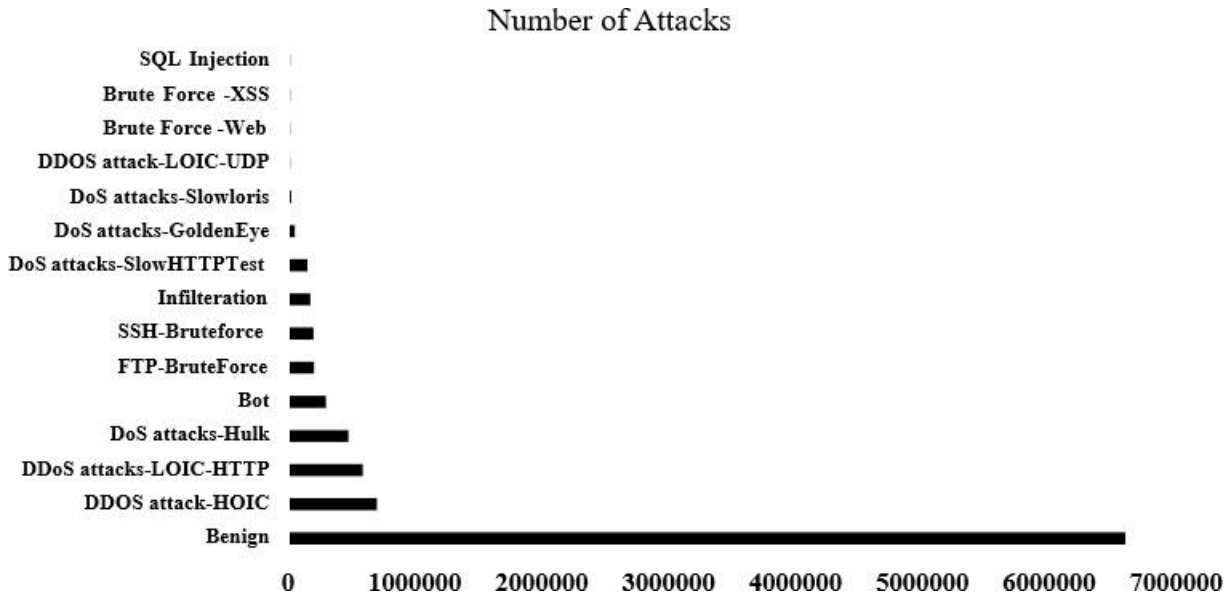

Figure 2.2 Attacks with distribution

The figure 4.2 illustrate the types of attacks with number of distributions that presented in the dataset.

2.1.3 Processing Skewed Dataset

In an intrusion detection dataset, only a small fraction of the dataset reflects attacks. This makes it challenging to create efficient models with high attack detection rates. A very biased prediction model is not practically useful because the prediction is extremely ill with the bulk class samples.

CIC-IDS 2018 is a highly imbalanced ratio for various attack categories. The mitigation of the sophistication imbalance of giant Data poses an even greater problem because of the comparatively diverse and nuanced nature of an oversized dataset. Many approaches are proposed within the research literature to handle imbalanced data.

2.1.3.1 Resampling Techniques

The problem of an imbalanced dataset is difficult to handle. So resampling techniques have been created. Resampling techniques include oversampling, under-sampling, combining oversampling and under-sampling techniques, and ensemble sampling. These resample techniques are aimed at changing the ratios between the majority classes and minority classes.

Under sampling

Under-sampling is a technique for dealing with uneven datasets in which all of the data is kept in the minority class and the size of the majority class is reduced. This method can be used to retrieve more accurate data from datasets that were previously skewed.

Oversampling

The oversampling method includes introducing duplicate records to the dataset at random in minority classes.

The following are some examples of oversampling techniques:

- Borderline-SMOTE
- SVM-based borderline oversampling
- Synthetic Minority Oversampling Technique
- Adaptive Synthetic Sampling (ADASYN)
- Random Oversampling (SMOTE)
- Synthetic Minority Oversampling Technique (SMOTE)

In this project, The Synthetic Minority Oversampling Technique (SMOTE) from Oversampling is used to increase majority records in Web Attacks.

a. Synthetic Minority Oversampling Technique (SMOTE)

The most often used approach for synthesizing new records is the Synthetic Minority Oversampling Technique, or SMOTE. This technique was devised by Nitesh Chawla et al. in their 2002 paper "SMOTE: Synthetic Minority Over-sampling Technique." SMOTE is one of the most commonly utilized oversampling algorithms to deal with an unbalanced dataset (synthetic minority oversampling technique). Its purpose is to re-create minority classes at random in order to balance the class distribution.

A hybrid strategy combining SMOTE and Under Sampling is offered to address the issue of category imbalance.

To overcome the skewness in the dataset, both SMOTE and under-sampling techniques have been explored. That some of the attack categories, like web attacks, are highly skewed. Web attack samples are oversampled using SMOTE. Web attack samples are increased to 14453 using SMOTE. As there are huge numbers of samples in normal traffic, and also some huge number of attacks can be under-sampled (i.e., 10 % of the original count). After applying under-sampling and SMOTE, the imbalance ratio is obtained.

2.2 Feature Selection

The performance of a machine learning model heavily depends upon the features selected for training the model. Excessive features may end up in the poor performance of the model. So, choosing the correct set of features is of paramount importance while building machine learning models. Selecting the proper features not only increases the efficiency of the model but also reduces the detection time. Intrusion detection datasets are generally high-dimensional datasets. For better efficiency, selecting the proper features in an IDS dataset is incredibly essential.

These techniques are classified as under :

- Filter methods
- Wrapper methods
- Embedded methods
- Hybrid methods

a) Embedded Methods

These methods combine the benefits of both the wrapper and filter methods by incorporating feature interactions while retaining a low computational cost. Embedded approaches are iterative in that they look after each iteration of the model training process and thoroughly extract the characteristics that are most important to the training for that iteration.

There are two widely used embedded approaches that are discussed. They are,

- Regularization Method
- Tree-based method

In this project, a tree-based method has been implemented, which gives better results for datasets having many more features.

Tree-Based Method

One of the most appealing aspects of employing tree-based algorithms is that they are simple to understand. This also makes determining the importance of each variable in the tree-based approach's decision-making process simple. In other words, it is simple to calculate how much each variable contributes to that decision using this method.

There must be two techniques to feature selection utilising tree-based models in this project

- Random Forest
- Gradient Boosting

a) Random Forest

The random forest instance defining the number of trees is selected first in the SelectFromModel technique. The SelectFromModel method from Scikit-learn is then used to select the features automatically. SelectFromModel picks characteristics whose value is greater than the sum of all features' importance. SelectFromModel alters the threshold value as an input.

Calculate Feature Importance Random Forest

The value of a characteristic can be calculated by each tree in the random forest based on its potential to boost the purity of the leaves. It's about Classification And Regression Trees (CART) and how they work. The greater the increase in leaf purity, the more important the trait becomes. This is done for each tree separately, then averaged across all trees, and then normalized to 1. So, the sum of the importance scores calculated by a Random Forest is 1.

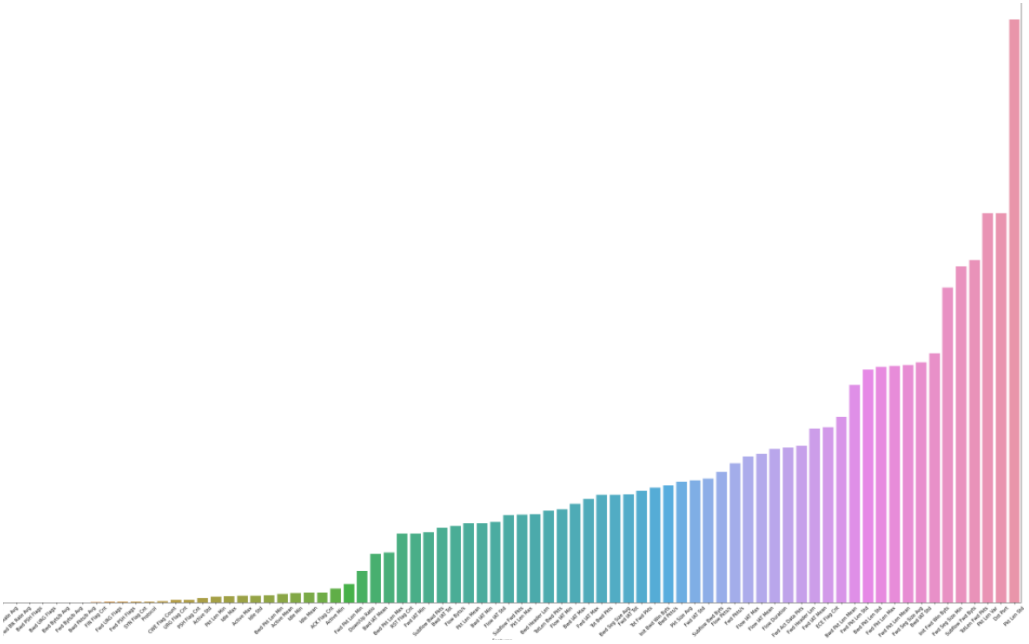

Figure 2.3 Feature Importance Representation

The figure 2.3, represents the importance of features by using random forest feature selection.

Table: 2.2.1 List of 37 selected features

By implementing tree based random forest it selected top 37 features are selected among 80 features which is listed below in Table: 2.2.1

S.No.	Selected Features Using Random Forest Feature Selection
1.	DST PORT
2.	FLOW DURATION
3.	TOT FWD PKTS
4.	TOT BWD PKTS
5.	TOTLEN FWD PKTS
6.	TOTLEN BWD PKTS
7.	FWD PKT LEN MAX
8.	FWD PKT LEN MEAN
9.	FWD PKT LEN STD
10.	BWD PKT LEN MEAN
11.	BWD PKT LEN STD
12.	FLOW PKTS/S
13.	FLOW IAT MEAN
14.	FLOW IAT MAX
15.	FLOW IAT MIN
16.	FWD IAT TOT
17.	FWD IAT MEAN
18.	FWD IAT MAX
19.	BWD IAT STD

20.	FWD HEADER LEN
21.	BWD HEADER LEN
22.	FWD PKTS/S
23.	BWD PKTS/S
24.	PKT LEN MAX
25.	PKT LEN STD
26.	PKT LEN VAR
27.	ECE FLAG CNT
28.	PKT SIZE AVG
29.	FWD SEG SIZE AVG
30.	BWD SEG SIZE AVG
31.	SUBFLOW FWD BYTS
32.	SUBFLOW BWD PKTS
33.	SUBFLOW BWD BYTS
34.	INIT FWD WIN BYTS
35.	INIT BWD WIN BYTS
36.	FWD ACT DATA PKTS
37.	FWD SEG SIZE MIN

b) Gradient Boosting Feature Selection

Gradient boosting uses a robust metric, called feature/importance, to retrieve the uncountable attribute in line with importance after the boosted tree is formed. This scoring model provides the importance of every feature in terms of constructing key decisions while constructing decision trees. Generally, feature importance provides a score that defines the assorted role of every attribute. This importance is computed by comparing and ranking all the features amongst each other within the dataset.

The amount of every feature split point, weighted by the number of observations from that node, is used to calculate the relevance of a single decision tree. This split point is used to improve the algorithm's performance and efficiency. Purity (Gini Index) is used to determine the split points and to identify a more particular error function. Every tree's feature importance is averaged across all of the model's decision trees. The most important aspect of this method is to use Weighted Feature Importance to save training time by deleting unimportant features from the dataset. Once the most promising ones have been identified using the Gradient Boosting Feature Selection technique, they may be used to train and test the model efficiently.

Table: 2.2.2 List of 13 selected features

List of 13 selected features using Gradient Boosting Feature Selection technique which is displayed in tabular form in table 2.2.2

S.No	selected features using gradient boosting feature selection method
1.	DST PORT
2.	FWD PKT LEN MEAN
3.	FLOW BYTS/S
4.	BWD IAT MIN
5.	FWD HEADER LEN
6.	PKT LEN MAX
7.	PKT LEN STD
8.	RST FLAG CNT

9.	FWD SEG SIZE AVG
10.	BWD SEG SIZE AVG
11.	INIT FWD WIN BYTS
12.	INIT BWD WIN BYTS
13.	FWD SEG SIZE MIN

CIC IDS2018 may be a high-dimensional dataset with 80 features. Several features are correlated and redundant. Training on the redundant data will increase the complexity and time and may lead to a flawed model. The proposed model used an embedded feature selection using Random Forest and Gradient Boosting.

First, the features are selected using Random Forest feature selection. Feature importance using a Random Forest classifier is computed, which supports the average impurity of every feature in a very large tree within the forest. Using this method, the 37 most significant features are selected. After that, the remaining 13 topmost features from the dataset are selected using the gradient boosting method.

2.3 Model Building

Machine learning techniques could be defined as a science and art that enables programmed computers to learn from data provided to them. Computers are frequently trained on the data (training set) given to them during the machine learning model, and they can demonstrate their performance on a specific data set (test set). In this manner, the problem is resolved while minimizing human intervention. Machine learning is widely used in many situations where traditional methods are inefficient. The following areas of application will be listed:

- It can interpret large amounts of complex data and solve complex problems that traditional methods cannot solve.
- Machine Learning methods can provide better solutions in situations where existing solutions require too much external intervention/update without external intervention.
- It can be done in a variety of settings. By examining the data, machine learning techniques are typically applied in the proposed situation.

Supervised learning

The machine learning model had been classified and labeled using this method. For instance, all flows in the dataset have data (labels/tags) concerning their nature (such as harmful or normal). In the following stage, the test/prediction process, these tags are especially in comparison to the algorithm's results, and the algorithm's progress is calculated. The strategy's performance is excellent. However, supervised learning is valuable because it labels data using external services (e.g., manual tagging), and this process is repeated until the algorithm obtains a high level of accuracy/performance.

Ensemble Methods

Ensemble learning is an essence meta-machine learning approach that seeks to improve predictive performance by implementing predictions from various models.

These techniques are classified into three types:

- Bagging
- Boosting
- Stacking

An effective intrusion detection system might also detect all types of attacks with a high attack detection rate. Various supervised machine learning techniques are proposed in the literature to achieve this. However, one algorithm is incapable of detecting all types of attacks effectively. During this study, the model is trained using multiple machine learning approaches, which are then combined using the proposed framework, which is based on the performance of each algorithm in classifying different attacks. Many algorithms are available in the literature, but seven of the most popular machine learning techniques are chosen for testing on the CIC- IDS2018 dataset for attack classification.

Most points of the algorithms used here are given below:

i. Bagging
   - Random Forest
   - Bagging Classifier
   - Decision Tree

ii. Boosting
   - Adaptive Boosting Classifier
   - Extreme Gradient Boosting Classifier
   - Light Gradient Boosting Classifier
   - Histogram-based gradient boosting classifier

2.3.1 Bagging

Bagging algorithms are parallel ensemble methods that combine bootstrap and aggregation. Bagging is used to acquire a subset of features of data for training that may have a high density and normal points in the border that may have a low density using bootstrap sampling. An ensemble random tree may work well for anomaly detection with this recommendation, and a random tree is efficient for calculating the adversity of isolating data points.

a) Random Forest

For classification, Random Forest employs a slew of Decision Tree classifiers. An extremely RF tree is trained on a bootstrapped sample from the dataset. The split characteristic is chosen at random to divide the sample-supported impurity based on the Gini index/entropy. The results of all trees are then combined by voting on the last word prediction. Almost all of the time, even when no hyperparameters are used, the RF results outperform the DT algorithm. It is a popular algorithm that produces quick and accurate results.

From the training set data, a random forest generates n decision trees. It randomly resamples the training data set for each tree during this process. As a result, n decision tree algorithms are obtained, and each differs from the other. Finally, it is done by selecting new estimates from n trees' estimates. The final value is determined by the worth with the highest rating.

b) Decision Tree

Decision trees are a common classifier used in machine learning techniques. The principles used in this approach are fairly understandable and straightforward.

Each decision tree has two nodes, a root-node and a sub-node, as well as branches and leaves. There is a call statement within each node. In accordance with the outcome of this decision, the algorithm selects one of the two branches in the following step (the number of branches could even be over two in some sub-algorithms). This chosen branch directs the algorithm to the next node. This procedure concludes with the final element, the leaf.

c) Bagging Classifier

A bagging classifier is an ensemble learning method that integrates the outputs of many learners to improve performance. It fits base classifiers on random subsets of the original dataset and then aggregates their predictions (via voting or averaging) to form a final prediction. An ensemble meta estimator of this type can be used to reduce the variance of black-box test results (e.g., a decision tree).

2.3.2 Boosting

Boosting creates a strong classification model from a collection of base learners. Boosting works sequential manner by training a base learner's set and combining it for prediction. The boosting algorithm iteratively applies the base learning algorithms, with different distributions or weightings of training data on the base machine learning.

a) Adaptive Boosting Classifier

The adaptive advantage The algorithm is primarily interested in maintaining the weights and over training data. At each round, the weights of incorrectly classified examples are increased so that the base learner can focus on the most difficult examples in the training data. The revised classifier is determined by a majority vote of the bottom classified.

Adaptive Boost is a variant of the Adaptive Boost algorithm that employs multiclass learners rather than binary classifiers. To reduce a pseudo-loss, AdaBoost employs the one-versus-one strategy. The one-versus-one strategy reduces a multitasking class to a binary class, where the purpose of each task is to label the occasions as belonging to the jth or kth class. Adaptive Boost mitigates the ranking loss. The superlative class includes the appropriate class.

b) Extreme Gradient Boosting Classifier

Extreme Gradient Boost is a tree-boosting method that makes the best use of available hardware and memory. It is nearly ten times faster than the most recent techniques. Extreme Gradient Boost can perform the three primary gradient boosting methods, namely Regularized Boosting, Gradient Boosting, and Stochastic Boosting. The Extreme Gradient Boost algorithm's main advantages are faster execution through parallel processing, portability, regularization, and tree pruning.

Gradient boosting techniques are used in this algorithm. This algorithm makes predictions based on the integration of weighted input data. Depending on the data, there may be a number of parameters. Finding suitable variables from a set of data is critical to ensuring the algorithm's performance. The predictive performance is used to forecast the final result.

c) Light Gradient Boosting Classifier

In 2017, Microsoft created LightBGM as a boosting framework. The Gradient Boosted Tree (GBDT) algorithm has been improved in LightGBM. In terms of speed, performance, and power, this framework outperforms Xgboost. Unlike the other GBDT techniques, LightGBM remains effective when the data is too large and has several dimensions.

This is because of two distinct strategies: a Special Feature Bundle (EFB) and Gradient-Based One-Side Sampling (GOSS). It is a tree-based method that supports categorical attributes, which eliminates the need for feature numerical transformation and normalization during the data preparation step. The tree-growth method based on leaves speeds up the matching process during decision-making.

Machine Learning hyperparameter values are used for experimentation to evaluate the proposed approach.

d) Histogram-based gradient boosting classifier

Gradient Boosting ensembles are usually ineffective in terms of time. Binning the continuous variables whereas training the model can improve tree training. Histogram-Based Gradient Boosting is a Gradient Boosting ensemble that bins continuous values to speed up the model (HBGB). HBGB was inspired by Microsoft's Light Gradient Boosting machine.

2.4 Evaluating Model Performance

A machine learning model's evaluation is critical for validation or evaluation. A machine learning algorithm is evaluated using a variety of metrics. The most appropriate metrics must be chosen in order to fine-tune a model which is based on its performance.

For evaluation, the following criteria are used:

- Confusion Matrix
- Accuracy
- Precision
- Recall
- F1-score

2.4.1 Confusion Matrix

The counts of test records accurately and inaccurately predicted by the model are used to evaluate the performance of a classifier. The confusion matrix gives a comprehensive picture of a predictive model's performance, including which classes are correctly predicted and incorrectly, as well as the types of errors made.

True Positive (TP): The cases in which one predicted yes and the actual output was also yes.

True Negative (TN): The cases in which one predicted no and the actual output was no.

False Positive (FP): The cases in which one predicted yes and the actual output was no.

False Negative (FN): The cases in which one predicted no and the actual output was yes.

Table 2.4 Prediction of Confusion Matrix

		ACTUAL	
		NEGATIVE	**POSITIVE**
PREDICTED	**NEGATIVE**	True Negative	False Negative
	POSITIVE	False Positive	True Positive

A confusion matrix is a table that shows how many true and false predictions a classifier made. It can be used to assess the performance of a classifier by computing performance metrics such as precision, recall, accuracy, and F1-score.

2.4.2 Accuracy

Accuracy is also defined as the ratio of correct positive cases to the number of cases under evaluation. The best accuracy value is 1 and the worst value is 0.

Accuracy = (True positives +True negatives) / (Total number of data items)

2.4.3 Precision

Precision can all be defined about relation to either class. The precision of the negative class is innately the classifier's ability to not label a negative sample as positive. The precision of true positive is intuitively the classifier's ability to not label a positive sample as negative. Precision has the best value of 1 and the worst value of 0.

Precision = (True positive) / (True Positive + False Positive)

2.4.4 Recall(Sensitivity)

Recall can be defined in terms of either of the classifications. The ratio of the True Positive(TP) to the number of actual positive cases is defined as the recall of the positive class. It can be expressed intuitively as the classifier's ability to capture all positive cases. It is also known as the True Positive Rate (TPR).

Sensitivity = True Positive / (True Positive + False Negative)

2.4.5 Recall(Specificity)

Specificity is defined as the ratio of the True Negative(TN) to the number of actual false negatives when recalling a negative class. It can be expressed intuitively as the classifier's ability to capture all negative cases. It is also known as the True Negative Rate (TNR).

Specificity = True Negative / (False Positive + True Negative)

2.4.6 F1-score

Regardless of class imbalance, the F1 score is regarded as one of the best performance measures for classification models. The F1 score is the weighted score of the class's recall and precision. It has the best value of 1 and the worst value of 0.

F1-score = (2 * Precision * Recall) / (Precision + Recall)

Accuracy is the best metric for evaluating the performance of the proposed ensemble methods. If the classification model is evaluated based on the attack detection accuracy, the performance of the model may suffer. A classifier with a high Recall rate but precision may classify one attack classification while misclassifying others, resulting in a decrease in an IDS's attack detection rate.

As a result, the accuracy performance measure is used to select the best technique for each attack classification in order to improve overall accuracy and attack detection rate. Despite the fact that accuracy is the ideal measure for selecting the best machine learning technique for each class in the proposed technique, accuracy is used in the comparative results.

The recall rate indicates the model's ability to reliably identify True Positives, whereas Accuracy is the ratio of classified instances predictions to total predictions. An IDS's primary goal is to accurately identify attacks and protect the network from intrusions. As a result, the accuracy rate is used to make comparisons of the findings.

3. RESULTS AND DISCUSSION

In this study, the dataset is trained on seven machine learning algorithms to find the best method to detect different attack categories.

3.1 Data Collection

CIC-IDS2018 (Canadian Institute for Cybersecurity (CIC) Intrusion Detection System 2018(IDS) - includes seven different attack scenarios: Brute-force, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside.

Fwd Blk Ra	Bwd Byts/l	Bwd Pkts/l	Bwd Blk Ra	Subflow Fv	Subflow Fv	Subflow Bv	Subflow Bv	Init Fwd W	Init Bwd W	Fwd Act Da	Fwd Seg Si	Active Mea	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	0	0	0	3	0	0	0	-1	-1	0	0	0	0	0	0	56320860	139.3	56320958	56320761	Benign
0	0	0	0	3	0	0	0	-1	-1	0	0	0	0	0	0	56320733	114.5513	56320814	56320652	Benign
0	0	0	0	3	0	0	0	-1	-1	0	0	0	0	0	0	56319312	301.9346	56319525	56319098	Benign
0	0	0	0	15	1239	10	2273	65535	233	6	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	14	1143	11	2209	5808	233	6	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	16	1239	12	2273	5808	234	7	20	0	0	0	0	0	0	0	0	Benign
0	0	0	0	3	0	0	0	-1	-1	0	0	0	0	0	0	56320240	203.6468	56320384	56320096	Benign
0	0	0	0	3	0	0	0	-1	-1	0	0	0	0	0	0	56320622	62.2254	56320666	56320578	Benign
0	0	0	0	5	211	3	463	14600	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	220	3	472	14600	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	220	3	472	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	209	3	461	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	211	3	463	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	206	3	458	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	211	3	463	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	211	3	463	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	214	3	466	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	209	3	461	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	215	3	467	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	215	3	467	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	215	3	467	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	215	3	467	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	215	3	467	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	215	3	467	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	216	3	468	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	215	3	467	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	215	3	467	14480	219	1	32	0	0	0	0	0	0	0	0	Benign
0	0	0	0	5	214	3	466	14480	219	1	32	0	0	0	0	0	0	0	0	Benign

Figure 3.1 Dataset

3.2 Data Transformation

3.2.1 Exploratory Data Analysis

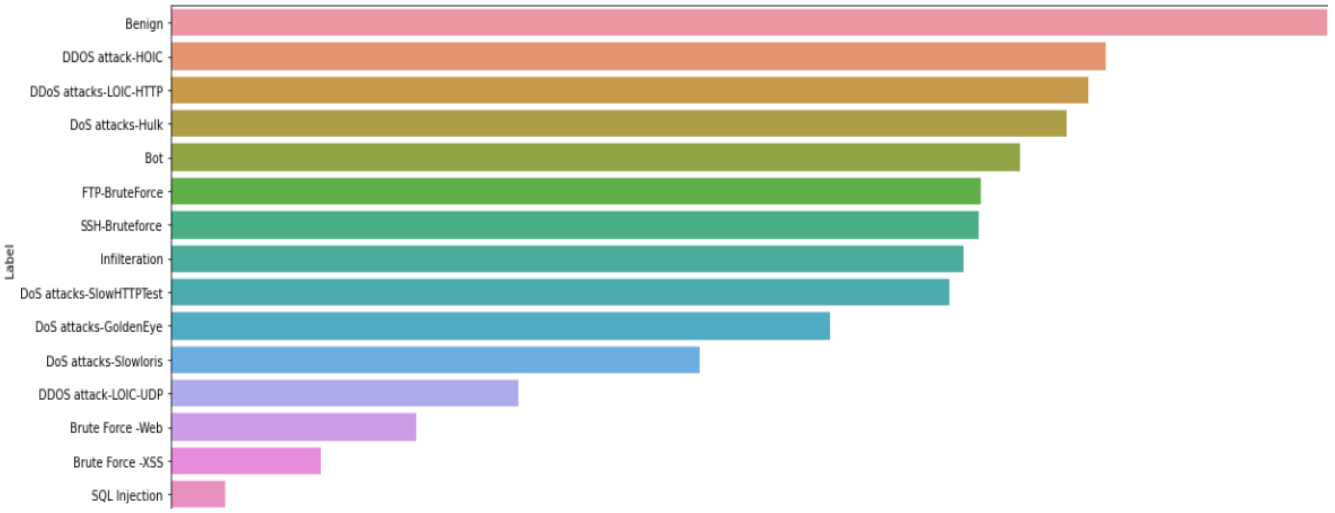

Figure 3.2 Attack distribution in the CIC IDS 2018 dataset

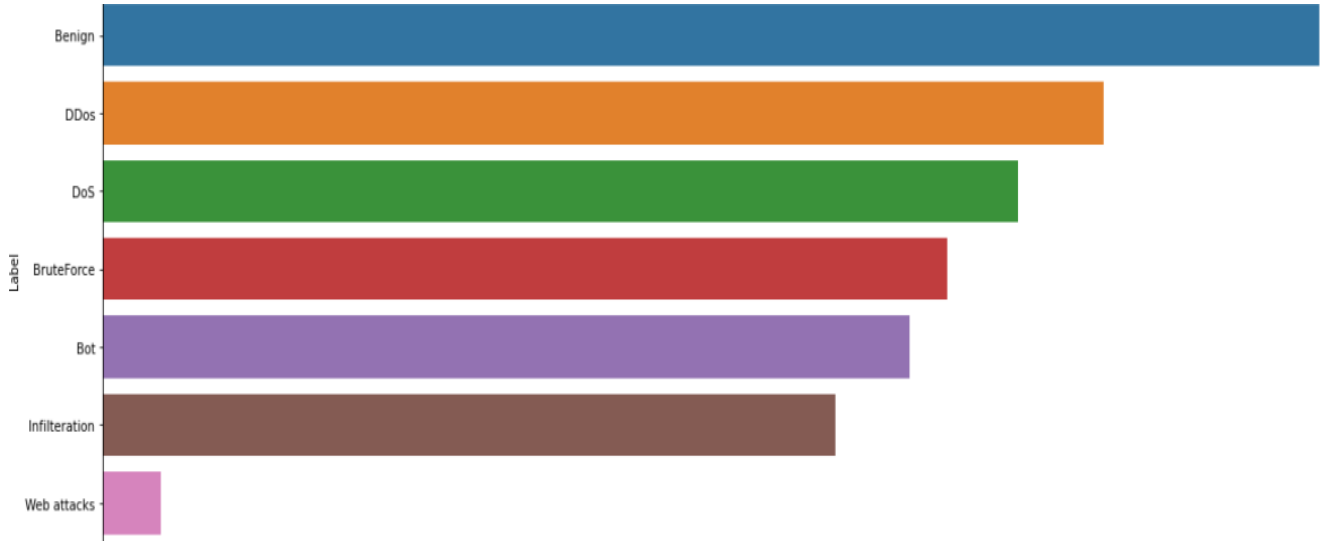

Figure 3.3 Attack distribution in the CIC IDS 2018 dataset after re-classified

3.3 Sampling Techniques

Table .3.3.1 Imbalance ratio of the samples in the CIC IDS 2018 dataset

Label	No Of Samples	% (Volume)	Imbalanced Ratio
Benign	5515108	82.98%	Majority Class
DDoS	775955	7.83%	10.5941019
DoS	196568	4.05%	20.46497784
Brute Force	94095	2.36%	35.15023245
Bot	144535	1.77%	46.78775713
Infiltration	140610	1.00%	83.35606546
Web attacks	861	0.01%	14429.13254
Total	6867732		

CIC-IDS2018 Before Sampling

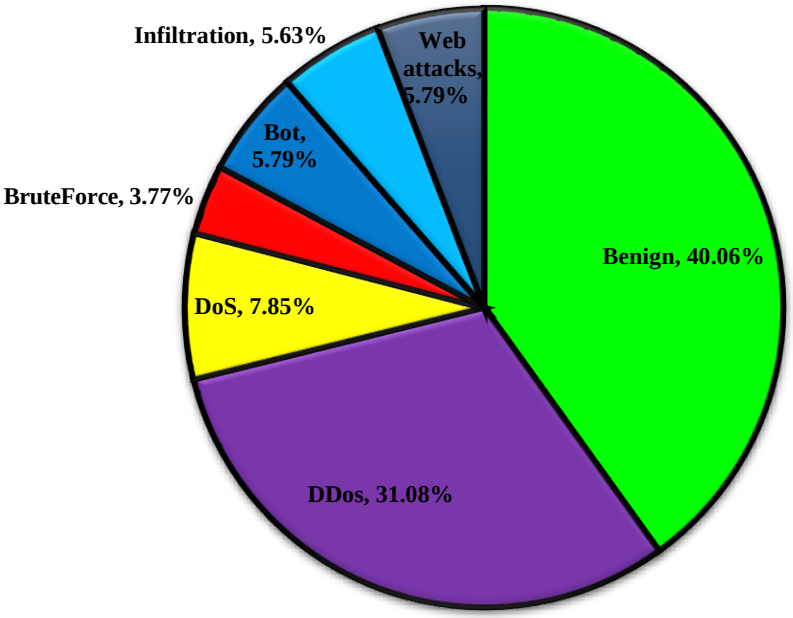

Figure 3.4 Attack distribution of CIC IDS 2018 dataset

Table : 3.3.2 Imbalance ratio of attack samples in the CIC IDS 2018 after sampling

Label	No Of Samples	% (Volume)	IMBALANCED RATIO
Benign	100000	40.06%	Majority Class
DDoS	77595	31.08%	1.099925392
DoS	19656	7.85%	2.124762341
Brute Force	9409	3.77%	3.649448868
Bot	14453	5.79%	4.857706916
Infiltration	14061	5.63%	8.654386544
Web attacks	14453	5.79%	4.857709619
Total	249627		

CIC-IDS2018 After Sampling

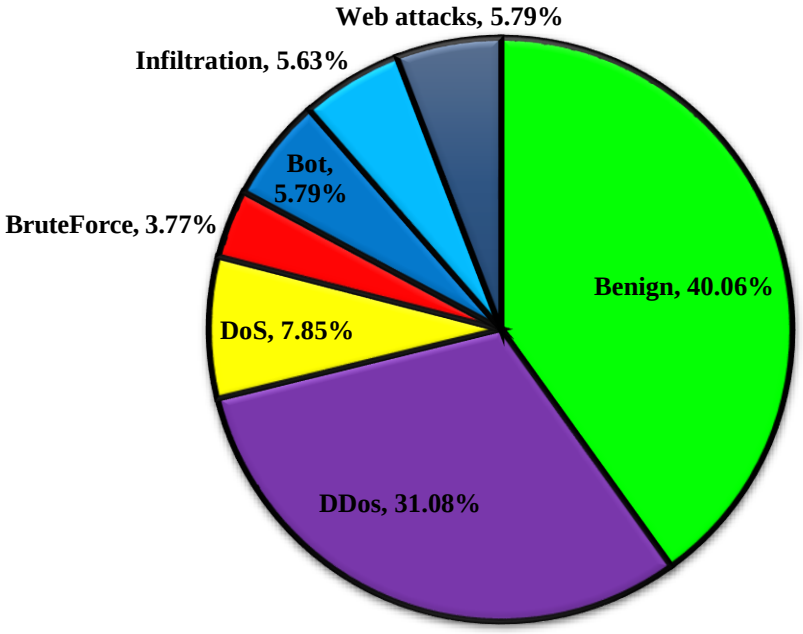

Figure 3.5 Attack distribution of CIC IDS 2018 dataset after sampling.

3.4 Feature Selection

```
'Dst Port', 'Protocol', 'Flow Duration', 'Tot Fwd Pkts', 'Tot Bwd Pkts
'TotLen Fwd Pkts', 'TotLen Bwd Pkts', 'Fwd Pkt Len Max',
'Fwd Pkt Len Min', 'Fwd Pkt Len Mean', 'Fwd Pkt Len Std',
'Bwd Pkt Len Max', 'Bwd Pkt Len Min', 'Bwd Pkt Len Mean',
'Bwd Pkt Len Std', 'Flow Byts/s', 'Flow Pkts/s', 'Flow IAT Mean',
'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min', 'Fwd IAT Tot',
'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min',
'Bwd IAT Tot', 'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd IAT Max',
'Bwd IAT Min', 'Fwd PSH Flags', 'Bwd PSH Flags', 'Fwd URG Flags',
'Bwd URG Flags', 'Fwd Header Len', 'Bwd Header Len', 'Fwd Pkts/s',
'Bwd Pkts/s', 'Pkt Len Min', 'Pkt Len Max', 'Pkt Len Mean',
'Pkt Len Std', 'Pkt Len Var', 'FIN Flag Cnt', 'SYN Flag Cnt',
'RST Flag Cnt', 'PSH Flag Cnt', 'ACK Flag Cnt', 'URG Flag Cnt',
'CWE Flag Count', 'ECE Flag Cnt', 'Down/Up Ratio', 'Pkt Size Avg',
'Fwd Seg Size Avg', 'Bwd Seg Size Avg', 'Fwd Byts/b Avg',
'Fwd Pkts/b Avg', 'Fwd Blk Rate Avg', 'Bwd Byts/b Avg',
'Bwd Pkts/b Avg', 'Bwd Blk Rate Avg', 'Subflow Fwd Pkts',
'Subflow Fwd Byts', 'Subflow Bwd Pkts', 'Subflow Bwd Byts',
'Init Fwd Win Byts', 'Init Bwd Win Byts', 'Fwd Act Data Pkts',
'Fwd Seg Size Min', 'Active Mean', 'Active Std', 'Active Max',
'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Idle Min', 'Label'
```

Figure 3.6 Before Feature Selection

The above figure 3.6 illustrates the total features that presents in the dataset , Here there is total 80 features are present which represents before feature selection.

a) Random Forest Feature Selection

```
'Dst Port', 'Flow Duration', 'Tot Fwd Pkts', 'Tot Bwd Pkts',
'TotLen Fwd Pkts', 'TotLen Bwd Pkts', 'Fwd Pkt Len Max',
'Fwd Pkt Len Mean', 'Fwd Pkt Len Std', 'Bwd Pkt Len Max',
'Bwd Pkt Len Mean', 'Bwd Pkt Len Std', 'Flow Pkts/s', 'Flow IAT Mean
'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min', 'Fwd IAT Tot',
'Fwd IAT Max', 'Bwd IAT Std', 'Bwd IAT Min', 'Fwd Header Len',
'Bwd Header Len', 'Fwd Pkts/s', 'Bwd Pkts/s', 'Pkt Len Max',
'Pkt Len Std', 'Pkt Len Var', 'RST Flag Cnt', 'Fwd Seg Size Avg',
'Bwd Seg Size Avg', 'Subflow Fwd Pkts', 'Subflow Fwd Byts',
'Subflow Bwd Byts', 'Init Fwd Win Byts', 'Fwd Act Data Pkts',
'Fwd Seg Size Min', 'Label'],
```

Figure 3.7 After Feature Selection using random forest Feature Selection

The above figure 3.7 refer to the importance of feature selection using the random forest. The figure which represents the CIC-IDS2018 dataset with 80 total features, In a figure which represents the 37 topmost selected features by using random forest model .

b) Gradient Boosting Feature Selection

```
'Dst Port', 'Fwd Pkt Len Mean', 'Flow Byts/s', 'Bwd IAT Min',
'Fwd Header Len', 'Pkt Len Max', 'Pkt Len Std', 'RST Flag Cnt'
'Fwd Seg Size Avg', 'Bwd Seg Size Avg', 'Init Fwd Win Byts',
'Init Bwd Win Byts', 'Fwd Seg Size Min'],
```

Figure : 3.8 After Feature Selection using Gradient Boosting Feature Selection

The above figure 3.8 refer to the importance of feature selection using the random forest. The figure which represents the CIC-IDS2018 dataset with 80 total features, In a figure which represents the 13 topmost selected features by using Gradient Boosting Feature Selection

3.5 Model Building

In this project, the dataset is trained on seven machine learning models to find the best method to detect different attack classifications. Based on the performance result of each machine learning model, a rank matrix was calculated. Based on the rank matrix, the results of the best-performing algorithm are considered for the final attack classification prediction.

3.5.1 Comparison of model in attacks classification with random forest feature selection:

	precision	recall	f1-score
Benign	0.96	0.92	0.94
Bot	1.00	1.00	1.00
BruteForce	1.00	1.00	1.00
DDos	1.00	1.00	1.00
DoS	1.00	1.00	1.00
Infilteration	0.43	0.62	0.51
Web attacks	1.00	1.00	1.00

Figure : 3.9 Bagging Classifier using random forest feature selection

	precision	recall	f1-score
Benign	0.84	1.00	0.91
Bot	1.00	0.96	0.98
BruteForce	1.00	0.99	1.00
DDos	1.00	1.00	1.00
DoS	1.00	1.00	1.00
Infilteration	0.00	0.00	0.00
Web attacks	1.00	0.64	0.78

Figure : 3.10 Decision Tree using random forest feature selection

	precision	recall	f1-score
Benign	0.836	1.000	0.911
Bot	1.000	0.958	0.979
BruteForce	1.000	0.993	0.997
DDos	1.000	1.000	1.000
DoS	1.000	1.000	1.000
Infilteration	0.000	0.000	0.000
Web attacks	1.000	0.641	0.781

Figure : 3.11 Random Forest using random forest feature selection

	precision	recall	f1-score
Benign	1.00	0.80	0.89
Bot	0.98	1.00	0.99
BruteForce	1.00	1.00	1.00
DDos	1.00	1.00	1.00
DoS	0.99	1.00	0.99
Infilteration	0.00	0.00	0.00
Web attacks	0.26	1.00	0.41

Figure : 3.12 Adaptive Boosting Classifier using random forest feature selection

	precision	recall	f1-score
Benign	0.83	1.00	0.91
Bot	1.00	0.98	0.99
BruteForce	1.00	0.99	1.00
DDos	1.00	1.00	1.00
DoS	1.00	1.00	1.00
Infilteration	0.00	0.00	0.00
Web attacks	1.00	0.62	0.77

Figure : 3.13 Extreme Gradient Boosting Classifier using random forest feature selection

	precision	recall	f1-score
Benign	0.91	0.99	0.95
Bot	1.00	1.00	1.00
BruteForce	1.00	1.00	1.00
DDos	1.00	1.00	1.00
DoS	1.00	1.00	1.00
Infilteration	0.83	0.31	0.45
Web attacks	1.00	1.00	1.00

Figure : 3.14 Light Gradient Boosting Classifier using random forest feature selection

	precision	recall	f1-score
Benign	0.92	0.99	0.95
Bot	1.00	1.00	1.00
BruteForce	1.00	1.00	1.00
DDos	1.00	1.00	1.00
DoS	1.00	1.00	1.00
Infilteration	0.81	0.35	0.49
Web attacks	1.00	1.00	1.00

Figure : 3.15 Histogram-based gradient boosting classifier using random forest feature selection

The above figures represent Model Building with bagging and boosting methods by using Random Forest feature selection selected features. Based on the rank matrix, the results of the best-performing algorithm are considered for the final attack prediction.

As given in the above figures, that a single model is not enough to detect all kinds of attack classifications. It is observed that some of the classifiers have a high Recall rate but a poor Precision rate. For example, the Bagging Classifier, Decision Tree, Random Forest, Extreme Gradient Boosting Classifier, Light Gradient Boosting Classifier, Histogram-based gradient boosting classifier performance results for detecting Botnet, Brute Force, DDoS, DoS attack show a high Recall rate of 100 % . Choosing these models as the best algorithm for detecting those attacks will increase the detection rate.

For Infiltration attack ,Light Gradient Boosting Classifier model gives high precision rate with 83% but low recall rate as 31% Choosing Light Gradient Boosting as the best algorithm for detecting Infiltration will increase the precision detection rate for Infiltration but decrease the detection rate due to low recall rate.

For Web attack, Bagging Classifier, Decision Tree, Random Forest, Extreme Gradient Boosting Classifier, Light Gradient Boosting Classifier, Histogram-based gradient boosting classifier gives high precision detection rate of 100% as well as same recall detection rate so choosing these models are best detection attack classification for web attack , whereas Adaptive Boosting Classifier model gives low precision detection rate with 26% but high recall rate of 100%.

3.5.2 Comparison of model in attacks classification with Gradient Boosting feature selection:

	precision	recall	f1-score
Benign	0.96	0.92	0.94
Bot	1.00	1.00	1.00
BruteForce	1.00	1.00	1.00
DDos	1.00	1.00	1.00
DoS	1.00	1.00	1.00
Infilteration	0.43	0.61	0.50
Web attacks	0.97	1.00	0.98

Figure : 3.16 Bagging Classifier using Gradient Boosting feature selection

	precision	recall	f1-score
Benign	0.91392	0.98528	0.94826
Bot	0.99965	0.99828	0.99896
BruteForce	1.00000	1.00000	1.00000
DDos	1.00000	1.00000	1.00000
DoS	1.00000	1.00000	1.00000
Infilteration	0.77623	0.36239	0.49411
Web attacks	0.99672	0.97017	0.98326

Figure : 3.17 Decision Tree using Gradient Boosting feature selection

	precision	recall	f1-score
Benign	0.841	1.000	0.913
Bot	1.000	0.983	0.992
BruteForce	1.000	0.993	0.997
DDos	1.000	1.000	1.000
DoS	0.995	1.000	0.998
Infilteration	1.000	0.000	0.001
Web attacks	0.994	0.658	0.792

Figure : 3.18 Random Forest using Gradient Boosting feature selection

	precision	recall	f1-score
Benign	0.99	0.74	0.85
Bot	0.00	0.00	0.00
BruteForce	0.99	1.00	1.00
DDos	1.00	1.00	1.00
DoS	0.83	1.00	0.91
Infilteration	0.00	0.00	0.00
Web attacks	0.81	0.94	0.87

Figure : 3.19 Adaptive Boosting Classifier using Gradient Boosting feature selection

	precision	recall	f1-score
Benign	0.45	1.00	0.62
Bot	0.00	0.00	0.00
BruteForce	0.00	0.00	0.00
DDos	0.00	0.00	0.00
DoS	0.62	0.84	0.72
Infilteration	0.00	0.00	0.00
Web attacks	0.00	0.00	0.00

Figure : 3.20 Extreme Gradient Boosting Classifier using Gradient Boosting feature selection

	precision	recall	f1-score
Benign	0.91	0.99	0.95
Bot	1.00	1.00	1.00
BruteForce	1.00	1.00	1.00
DDos	1.00	1.00	1.00
DoS	1.00	1.00	1.00
Infilteration	0.85	0.32	0.46
Web attacks	1.00	0.97	0.98

Figure : 3.21 Light Gradient Boosting Classifier using Gradient Boosting feature selection

	precision	recall	f1-score
Benign	0.91	0.99	0.95
Bot	1.00	1.00	1.00
BruteForce	1.00	1.00	1.00
DDos	1.00	1.00	1.00
DoS	1.00	1.00	1.00
Infilteration	0.81	0.36	0.49
Web attacks	1.00	0.97	0.98

Figure : 3.22 Histogram-based gradient boosting classifier using Gradient Boosting feature selection

The above figures represent Model Building with bagging and boosting methods byusing Gradient Boosting feature selection selected features. Based on the rank matrix, the results of the best-performing algorithm are considered for the final attack prediction.

For Bot attack Bagging Classifier, Random Forest, Light Gradient Boosting Classifier, Histogram-based gradient boosting classifier shows high precision and recall detection rate of 100 % so these models are refer as best for bot attack.

For Brute Force, DDoS, DoS attack , the models which gives high precision and recall detection rate with 100% in Bagging Classifier, Decision Tree Random Forest, Light Gradient Boosting Classifier, Histogram-based gradient boosting classifier.

In Infiltration attack the high precision detection rate is 100% and recall decision rateis 0 in Random Forest model . So random forest model is best for detecting the infiltration attack.

For Web Attacks the high precision detection rate is 100% and high recall rate is 97% in Light Gradient Boosting Classifier, Histogram-based gradient boosting classifier. So these are the best algorithms to detect the web attacks.

3.6 Model Comparison

To compute the results in machine learning model with accuracy score hence accuracy score give better performance among those seven machine learning model.

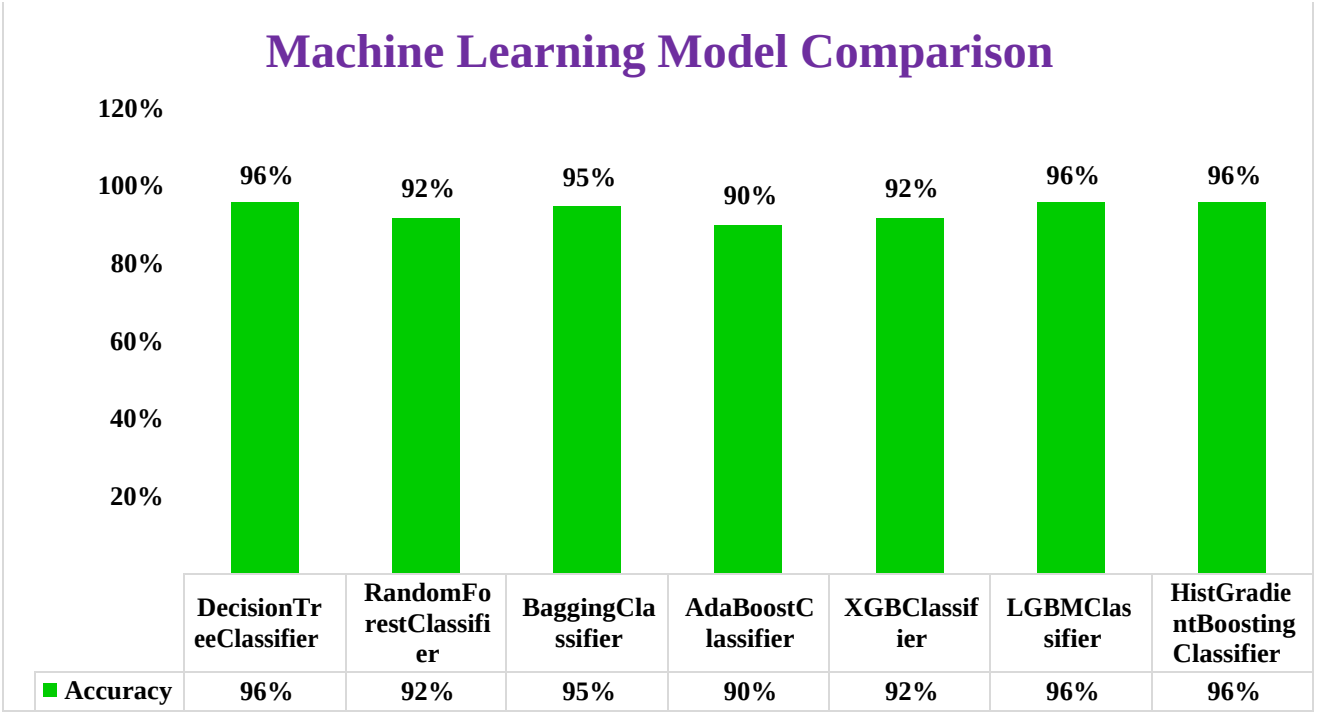

Figure : 3.23 Comparison Model Using Random Forest Feature Selection Accuracy Score

From the above figure 3.23, By using random forest feature selection, the highest accuracy score is obtained in Decision tree, Light gradient Boosting , Histogram Gradient Boosting is 96%.

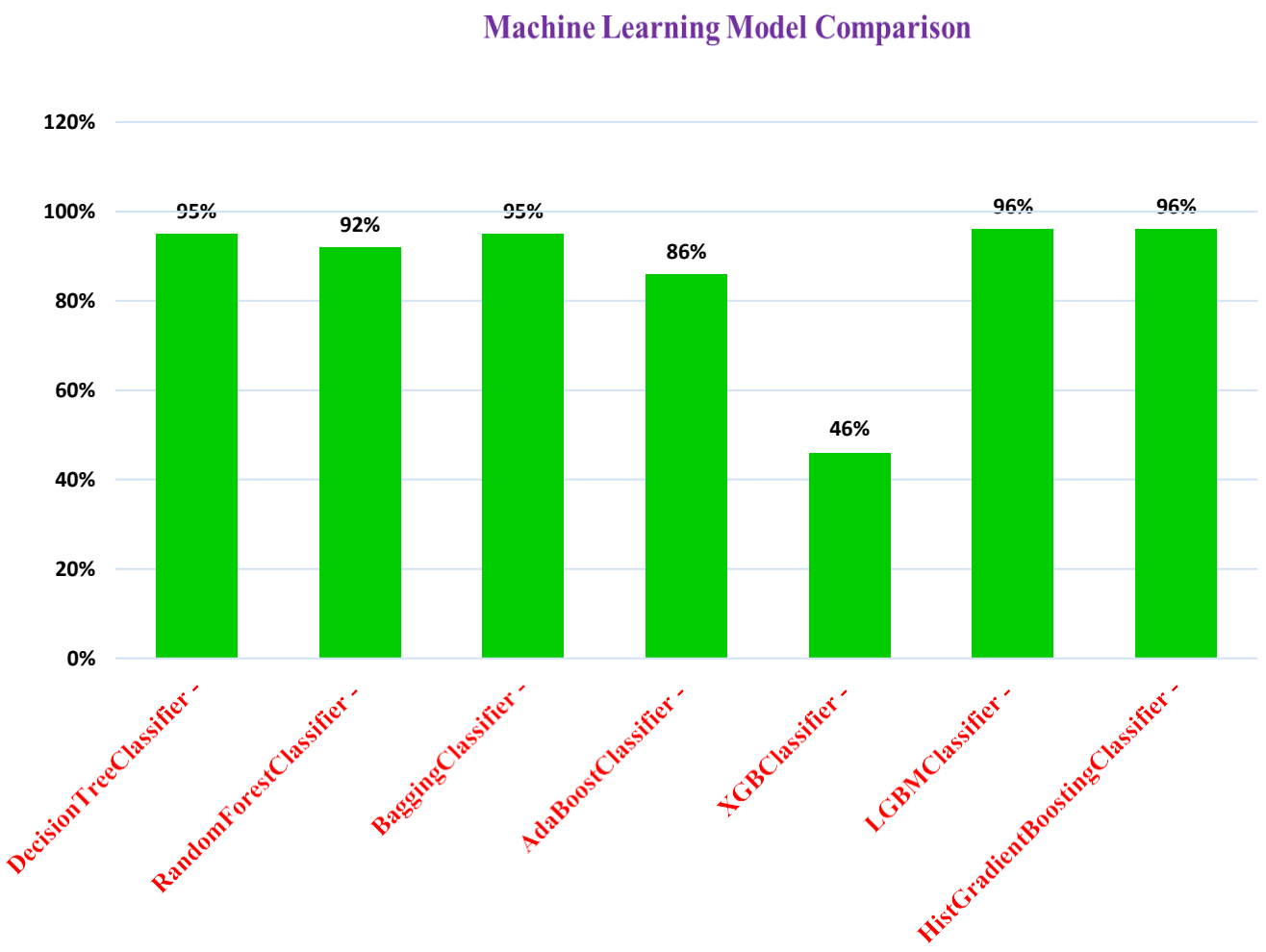

Figure 3.24 Comparison Model Using Gradient Boosting Feature Selection Accuracy Score

From the above figure 3.24, By using Gradient Boosting feature selection, the highest accuracy score is obtained in Light gradient Boosting , Histogram Gradient Boosting is 96%.

4. Conclusion and Future Scope

The study aimed to detect the abilities of multiple classifiers to improve attack detection accuracy. This paper proposes a machine learning model featuring a novel framework that brings together the advantages of several base classifiers for this, and different machine learning algorithms have been trained and tested on the latest CIC IDS 2018 dataset. By Using Random Forest Feature Selection -Decision Tree, LightGBM, and HBGB were finally used to detect multiple attacks with high attack detection rates and low prediction latency using the proposed framework. In Gradient Boosting Feature Selection LightGBM and HBGB were finally used to detect multiple attacks with high accuracy.

The CIC IDS 2018 dataset is highly skewed, so the problem of class imbalance was addressed using a hybrid approach of under-sampling of majority class and oversampling some of the attack classes using the SMOTE technique. This dataset balancing was done for the training process. The proposed approach enhances the detection accuracy of many attack categories in Machine Learning approaches. Future work can explore unsupervised learning to train models on unlabeled datasets in the security domain.

The training and test data in this study came from a series of CSV files comprising features extracted from the network flow. However, in real-world systems, this strategy is not feasible. However, by introducing a module that captures real network data and makes it workable with the deep learning algorithm, this problem can be solved.

5. References

1. S. Seth, K. K. Chahal and G. Singh, "A Novel Ensemble Framework for an Intelligent Intrusion Detection System," in *IEEE Access*, vol. 9, pp. 138451-138467, 2021, doi: 10.1109/ACCESS.2021.3116219.

2. T. N. Rincy and R. Gupta, "Ensemble Learning Techniques and its Efficiency in Machine Learning: A Survey," 2nd International Conference on Data, Engineering and Applications (IDEA), 2020, pp. 1-6, doi: 10.1109/IDEA49133.2020.9170675.

3. D. Upadhyay, J. Manero, M. Zaman and S. Sampalli, "Gradient Boosting Feature Selection With Machine Learning Classifiers for Intrusion Detection on Power Grids," in IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 1104-1116, March 2021, doi: 10.1109/TNSM.2020.3032618.

4. Towards Enhancement of Machine Learning Techniques Using CSE-CIC-IDS2018 Cybersecurity Dataset Ravikumar, Dharshini. Rochester Institute of Technology ProQuest Dissertations Publishing, 2021. 28321989.

5. S. Ho, S. A. Jufout, K. Dajani and M. Mozumdar, "A Novel Intrusion Detection Model for Detecting Known and Innovative Cyberattacks Using Convolutional Neural Network," in IEEE Open Journal of the Computer Society, vol. 2, pp. 14-25, 2021, doi: 10.1109/OJCS.2021.3050917.

6. N. F. Haq, A. R. Onik and F. M. Shah, "An ensemble framework of anomaly detection using hybridized feature selection approach (HFSA)," 2015 SAI Intelligent Systems Conference (IntelliSys), 2015, pp. 989-995, doi: 10.1109/IntelliSys.2015.7361264.

7. HCRNNIDS: Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System by Muhammad Ashfaq Khan IoT and Big-Data Research Center, Department of Electronics Engineering, Incheon National University, Incheon 2012, Korea Academic Editor: Chien-Chih Wang

8. Q. R. S. Fitni and K. Ramli, "Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems," 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), 2020, pp. 118-124, doi: 10.1109/IAICT50021.2020.9172014.

9. W. Seo and W. Pak, "Real-Time Network Intrusion Prevention System Based on Hybrid Machine Learning," in IEEE Access, vol. 9, pp. 46386-46397, 2021, doi: 10.1109/ACCESS.2021.3066620.

10. P. Chuang and S. Li, "Network Intrusion Detection using Hybrid Machine Learning," 2019 International Conference on Fuzzy Theory and Its Applications (iFUZZY), 2019, pp. 1-5, doi: 10.1109/iFUZZY46984.2019.9066223.