DDoS (Distributed Denial of Service) ATTACK DETECTION USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

NAVEEN KUMAR N (910619106034)

MUKESH KUMAR R (910619106032)

GNANA SEKAR P (910619106016)

PRAVEEN RAJ T (910619106303)

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

K.L.N. COLLEGE OF ENGINEERING, POTTAPALAYAM

(An Autonomous Institution, Affiliated to Anna University Chennai)

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“DDoS (Distributed Denial of Service) ATTACK DETECTION USING MACHINE LEARNING”** is the bonafide work of

NAVEEN KUMAR N (910619106034)

MUKESH KUMAR R (910619106032)

GNANA SEKAR P (910619106016)

PRAVEEN RAJ T (910619106303)

Who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Dr. V. KEJALAKSHMI, M.E.,Ph.D.,	**Mr. S.R. NARESH, M.E.,**
HEAD OF THE DEPARTMENT	
Professor, Dept. of ECE,	Associate professor, Dept. of ECE,
K.L.N. College of Engineering, Pottapalayam, Sivagangai-630 612.	K.L.N. College of Engineering, Pottapalayam, Sivagangai-630 612.

Submitted for the project viva-voce conducted on__________________

INTERNAL EXAMINAR EXTERNAL EXAMINAR

ACKNOWLEDGEMENT

We extend our gratitude to the Founder, **Late Thiru. K.L.N. KRISHNAN, K.L.N College of Engineering and Management Members** for making us march towards the glory of success. We express our sincere thanks to our respected Principal **Dr A.V.RAM PRASAD**, **M.E, Ph.D., MISTE.,FIE.,** for all the facilities offered.

We would like to express our profound gratitude and heartfelt thanks to **Dr. V. KEJALAKSHMI, M.E., PH.D.,** Head of the Department of Electronics and Communication Engineering, who motivated and encouraged us to do this outrival project for this academic year.

Our delightful and sincere thanks to our Project guide **Mr .S.R. NARESH.,M.E.,** and our respected project mentor **Dr .P. KARPAGAVALLI., M.E., Ph.D.,** and our respected Project Coordinator **Mrs .U.DHIVYA.,M.E.,** whose support was inevitable during the entire period of our work.

We thank our teaching staffs for sharing their knowledge and view to enhance our project. We also thank our non-teaching staff for extending their technical support to us. I thank my parents for giving me such a wonderful life and my friends for their friendly encouragement throughout the project. Finally, we thank the Almighty for giving the full health to finish the project successfully.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Distributed Denial of Service (DDoS) attacks pose an ongoing threat to businesses and organizations, causing extensive downtime and significant financial losses. Due to the intricate nature of DDoS attacks, conventional methods of detection often prove inadequate. Machine learning (ML) algorithms have emerged as a promising solution for detecting DDoS attacks by recognizing patterns and irregularities in network traffic that may indicate an attack. This article offers a comprehensive overview of DDoS attack detection utilizing ML, covering a range of algorithms, including supervised, unsupervised, and deep learning. The article also discusses crucial features required to train ML models for detecting DDoS attacks, such as packet size, packet rate, and network flow features. Furthermore, this paper evaluates various machine learning algorithms such as random forest, cat-boost classifier, and gradient boosting by predicting their accuracy.

CHAPTER - 1

1. INTRODUCTION

1.1. OVERVIEW

DDoS (Distributed Denial of Service) attacks are a serious issue for companies worldwide. DDoS attacks can cause service interruptions, money loss, and reputational harm to an organization. Using machine learning algorithms to spot and stop DDoS attacks is one technique for minimizing their effects.

This project aims to create a machine learning-based DDoS assault detection system. A dataset of network traffic data, comprising both legitimate and malicious traffic, will be used to train the system. The incoming network traffic will subsequently be classified as either normal or an attack using the trained model.

1.2. PURPOSE OF THE PROJECT

The project aims to create a machine learning-based DDoS attack detection system. The technology uses network traffic data analysis to find and stop DDoS attacks on a network. The project aims to offer a conditions-deployable, trustworthy, and practical approach to identifying DDoS attacks. The project will assist organizations in reducing the effects of DDoS assaults, such as service interruptions, financial losses, and reputational damage to a company, by effectively establishing a DDoS attack detection system.

The project's success will be measured based on the accuracy of the trained model in detecting DDoS attacks and the system's ability to handle a high volume of network traffic in real-time.

1.3. Distributed Denial of Service (DDoS) :

Attacks known as distributed denial of service (DDoS) are a frequent cyberthreat that can seriously harm businesses and their online services. In a DDoS attack, a large group of hacked systems or devices, known as a botnet, are used to overload a targeted network, server, or website with a tremendous quantity of traffic and prevent normal users from accessing it. By examining network traffic data, machine learning techniques like Random Forest, Gradient Boosting, and Cat-Boost have been used to identify DDoS attacks. These algorithms can spot trends and irregularities in the traffic data that point to a DDoS assault, including an unusual spike in traffic from a particular IP address or an increase in the number of requests.

1.4. Requirements : H/W Requirements :

- Processor – i3 / i5
- RAM – 4GB / 8GB & System Free Space – Minimum 15GB

S/W Requirements :

- Programming Language – Python
- IDE – Jupyter Notebook (Anaconda)

CHAPTER - 2

2. Literature Survey :

Table 2.1: Literature Survey

S.No	Title	Author Name	Objective	Disadvantage
1.	Hybrid Deep Learning – based Anomaly Detection Scheme for Suspicious Flow Detection in SDN.	Sahil Garg , Kuljeet Kaur , Neeraj Kumar ,Joel J.P.C.Rodrigues	The paper aims to provide insights into which machine learning algorithms are most effective for detecting DDoS attacks	They have used Bagging algorithms like SVM, Naïve Bayes,etc..
2.	DDoS Attack Detection Using Machine Learning Techniques	Shuang Wang, Qinghe Du, and Yanan Sun	The main goal of the paper is to provide a comprehensive review of the state of the art in DDoS attack detection using machine learning	The study only used a limited set of features to detect DDoS attacks, which may not be sufficient to accurately detect all

			techniques, and to identify the key research challenges and opportunities in this area	possible types of DDoS attacks.
3.	A Comparative Analysis of Machine Learning Algorithms for DDoS Attack Detection	Mohammad A. Al-Faraj, Khaleel M. Al-Salem, and Abdulaziz Al-Shetwi	The main goal of the paper is to contribute to the body of knowledge on DDoS attack detection and to help practitioners make more informed decisions about how to detect and respond to DDoS attacks.	The study used a small dataset with only two types of DDoS attacks, which may not be representative of all possible types of DDoS attacks.

4.	A Novel Method for Detecting DDoS Attacks Based on Random Forest Algorithm	Meng Liu, Shengli Liu, and Huaqi Wang	The paper aims to provide insights and recommendations for researchers and practitioners who are interested in developing or implementing machine learning-based DDoS attack detection systems.	The study did not test the algorithms in a real-world environment, which may have different characteristics and performance compared to the controlled environment of the study
5.	Anomaly-based DDoS Detection Using Deep Learning	Andra Lutu, Valerio Bruschi, and Alberto Dainotti	To identify the strengths and weaknesses of different machine learning algorithms for detecting DDoS attacks.	The approach proposed in the paper may not be scalable to large-scale networks

CHAPTER - 3

3. EXISTING SYSTEM

Machine Learning Techniques used:

3.1. Support Vector Machines (SVMs) :

Support Vector Machines (SVMs) are a group of machine learning algorithms with supervised learning that are used for regression and classification. SVMs works by locating the ideal "hyperplane," or border, that divides the data points into distinct classes. An SVM algorithm looks for a hyperplane that maximises the margin between the two classes in a binary classification task. The margin is the separation between each class's nearest data points and the hyperplane. The biggest margin hyperplane is selected by the SVM algorithm because it is less likely to incorrectly categorize new data points.

A kind of SVM created expressly for classification issues is Support Vector Classification (SVC). SVC works by transforming the input data into a higher-dimensional space where the data can be separated by a hyperplane. The transformation is done using a kernel function that maps the input data into a higher-dimensional space.

3.2. The Restricted Boltzmann Machine (RBM) :

Unsupervised machine learning algorithms for feature extraction and dimensionality reduction include the Restricted Boltzmann Machine (RBM). RBMs are a particular kind of artificial neural network that can autonomously learn a probability distribution over the input data. RBM is a potent unsupervised machine learning technique that has the ability to learn intricate representations of input data and extract crucial characteristics. It is helpful for many applications, including DDoS attack detection, because to its capacity for performing dimensionality reduction.

3.3. The Geometric Mean Decomposition Support Vector Machine (GDSVM) :

A machine learning algorithm called the Geometric Mean Decomposition Support Vector Machine (GDSVM) combines the geometric mean decomposition (GMD) and the Support Vector Machine (SVM) algorithms. For doing classification tasks, GDSVM is especially beneficial for high-dimensional data sets with unbalanced classes. For classification tasks, the GDSVM method, a potent machine learning technique, includes the GMD and SVM algorithms.

It is a helpful tool for many applications, including the detection of DDoS attacks, due to its capacity to handle wide data with classified.

CHAPTER - 4

4. SYSTEM MODEL

Acquiring and preprocessing network traffic data, extracting pertinent features, training machine learning models, assessing model performance, and deploying the models in a real-time environment are all steps in a system model for detecting DDoS attacks.

4.1. Data collection

This involves gathering information about network traffic from multiple devices, including switches, routers, and firewalls. The data may consist of details like flow characteristics, IP addresses, and packet headers.

4.2. Pre-processing

The collected data is pre-processed to extract relevant features and remove noise. This can include techniques such as filtering, normalization, and feature selection.

4.3. Feature extraction

This involves extracting meaningful features from the pre-processed data. This can be done using various techniques such as statistical analysis, machine learning algorithms, and pattern recognition.

4.4. Model training

The extracted features are used to train machine learning models for DDoS attack detection. This can include algorithms such as Random Forest, Cat-Boost, and Gradient

Boosting.

4.5. Model evaluation

The trained models are evaluated using performance metrics as accuracy. This step is important to ensure that the models are effective in detecting DDoS attacks and have low false positive rates.

4.6. Deployment

The trained models are deployed in a real-time environment for continuous monitoring of network traffic. This can include deploying the models on edge devices, such as routers and firewalls, or in a cloud-based environment.

4.7. DATASETS FEATURES :

Datasets had been taken from Kaggle dataset & Features of datasets have been given below ;

4.7.1. Source IP :

A source IP feature dataset is a collection of data that includes various attributes or characteristics of IP addresses that originate traffic on a network. This dataset can be used in network traffic analysis, intrusion detection, and other cybersecurity applications.

4.7.2. Source Port :

The source port is one of the attributes or features that are commonly recorded for each network communication session. The source port number is a 16-bit value that identifies the sending application or service for a particular network communication.

4.7.3. Destination IP :

The destination IP address is another attribute or feature that is commonly recorded for each network communication session. The destination IP address is a 32-bit value that identifies the destination device or host on the network to which the communication is being sent.

4.7.4. Forward IAT min :

Forward IAT min is the minimum interarrival time between packets in the forward direction for a given network communication session. It is a measure of the smallest time interval between two consecutive packets sent by the source device.

4.7.5. Forward IAT max :

Forward IAT max is the maximum interarrival time between packets in the forward direction for a given network communication session. It is a measure of the largest time interval between two consecutive packets sent by the source device.

4.7.6. Backward packets :

Backward packets are a type of network traffic packet that travels from the destination device back to the source device. They can be included as a feature or attribute in a network traffic dataset.

4.7.7. Backward IAT min :

Backward IAT min is another feature or attribute that can be included in a network traffic dataset. IAT stands for Interarrival Time, which is the time elapsed between two consecutive packets in a network communication session. Backward IAT refers to the time elapsed between the transmission of two consecutive packets in the backward direction, i.e., from the destination back to the source.

4.7.8. Backward IAT max :

Backward IAT max is another feature or attribute that can be included in a network traffic dataset. IAT stands for Interarrival Time, which is the time elapsed between two consecutive packets in a network communication session. Backward IAT refers to the time elapsed between the transmission of two consecutive packets in the backward direction, i.e., from the destination back to the source.

4.7.9. Backward IAT Total :

Backward IAT Total is another feature or attribute that can be included in a network traffic dataset. IAT stands for Interarrival Time, which is the time elapsed between two consecutive packets in a network communication session. Backward IAT refers to the time elapsed between the transmission of two consecutive packets in the backward direction, i.e., from the destination back to the source.

4.7.10. Backward IAT Mean :

Backward IAT Mean is another feature or attribute that can be included in a network traffic dataset. IAT stands for Interarrival Time, which is the time elapsed between two consecutive packets in a network communication session. Backward IAT refers to the time elapsed between the transmission of two consecutive packets in the backward direction, i.e., from the destination back to the source.

4.7.11. Backward IAT Standard :

Backward IAT Standard Deviation (or Standard) is another feature or attribute that can be included in a network traffic dataset. IAT stands for Interarrival Time, which is the time elapsed between two consecutive packets in a network communication session. Backward IAT refers to the time elapsed between the transmission of two consecutive packets in the backward direction, i.e., from the destination back to the source.

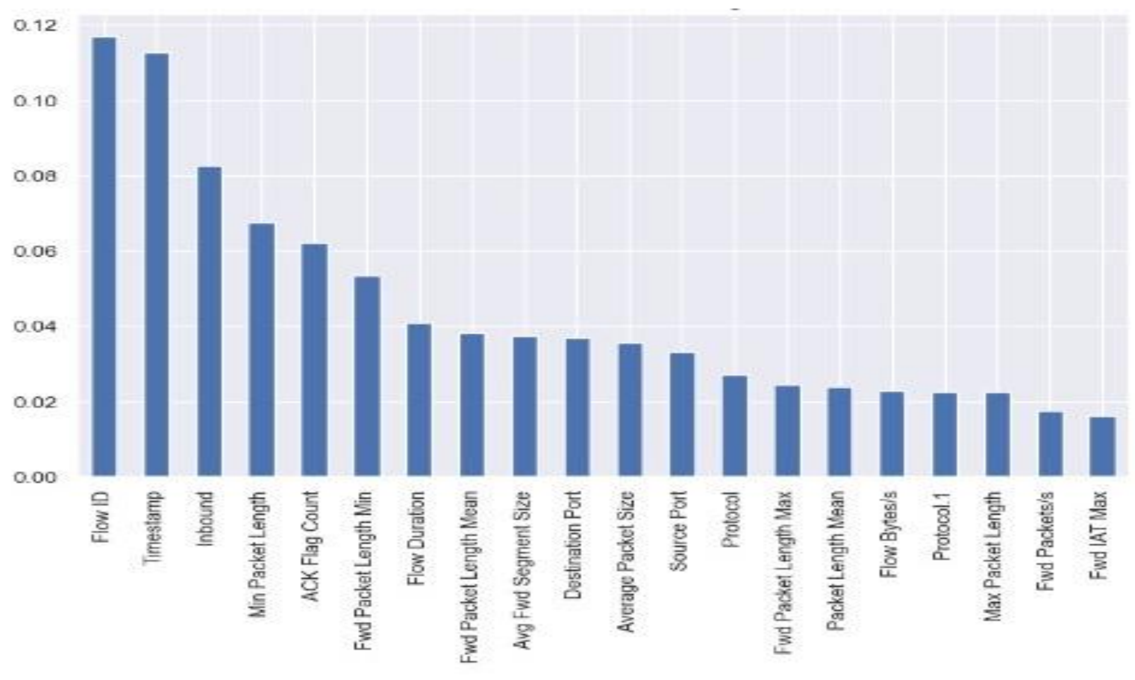


Figure 4.1: Standardised Dataset Features

4.8. EXPLORATORY DATA ANALYSIS :

4.8.1. BENIGN :

The term "benign" is typically used to describe legitimate traffic that is not part of the attack. In other words, benign traffic refers to traffic that is not malicious or harmful and is not part of the DDoS attack.

4.8.2. UDP :

UDP (User Datagram Protocol) is a transport layer protocol that is commonly used for network communication in applications that require low latency and high throughput. To mitigate a UDP flood attack, network administrators may use various techniques such as rate limiting, traffic filtering, or deploying specialized hardware or software solutions that are designed to detect and mitigate DDoS attacks.

4.8.3. UDP LAG :

UDP lag is a term that is sometimes used to describe a delay or slowdown in network traffic caused by a UDP flood attack. In a UDP flood attack, the attacker sends a large number of UDP packets to the target network or server in order to overwhelm its resources and cause it to become unresponsive.

4.8.4. PORT MAP :

Port mapping is a technique used by network administrators to map network traffic flows between different network devices or applications. It involves assigning specific ports on a network device to specific applications or services, so that incoming network traffic can be directed to the appropriate application or service.

4.8.5. SYN :

SYN (synchronization) flooding is a type of DDoS attack that exploits a weakness in the TCP (Transmission Control Protocol) handshake process. In a TCP handshake, the sender (client) sends a SYN packet to the receiver (server), which responds with a SYN-ACK packet, and then the sender responds with an ACK packet to complete the handshake and establish a connection. SYN flooding attacks can also be used to mask other types of attacks, such as malware infections.

4.8.6. NETBIOS :

NetBIOS (Network Basic Input/Output System) is a protocol used by Microsoft Windows operating systems to share files, printers, and other resources over a local area network (LAN). It is a legacy protocol that is no longer widely used, but some organizations may still have systems that rely on it. To mitigate the impact of a NetBIOS-based DDoS attack, network administrators may use various techniques such as traffic filtering, rate limiting, or disabling NetBIOS altogether if it is not needed in their network environment.

4.8.7. LDAP :

LDAP (Lightweight Directory Access Protocol) is a protocol used for accessing and maintaining distributed directory information services over a network. It is commonly used in enterprise environments for centralized authentication and authorization management. To mitigate the impact of an LDAP-based DDoS attack, network administrators may use various techniques such as traffic filtering, rate limiting, or implementing access control policies that restrict unauthorized LDAP queries.

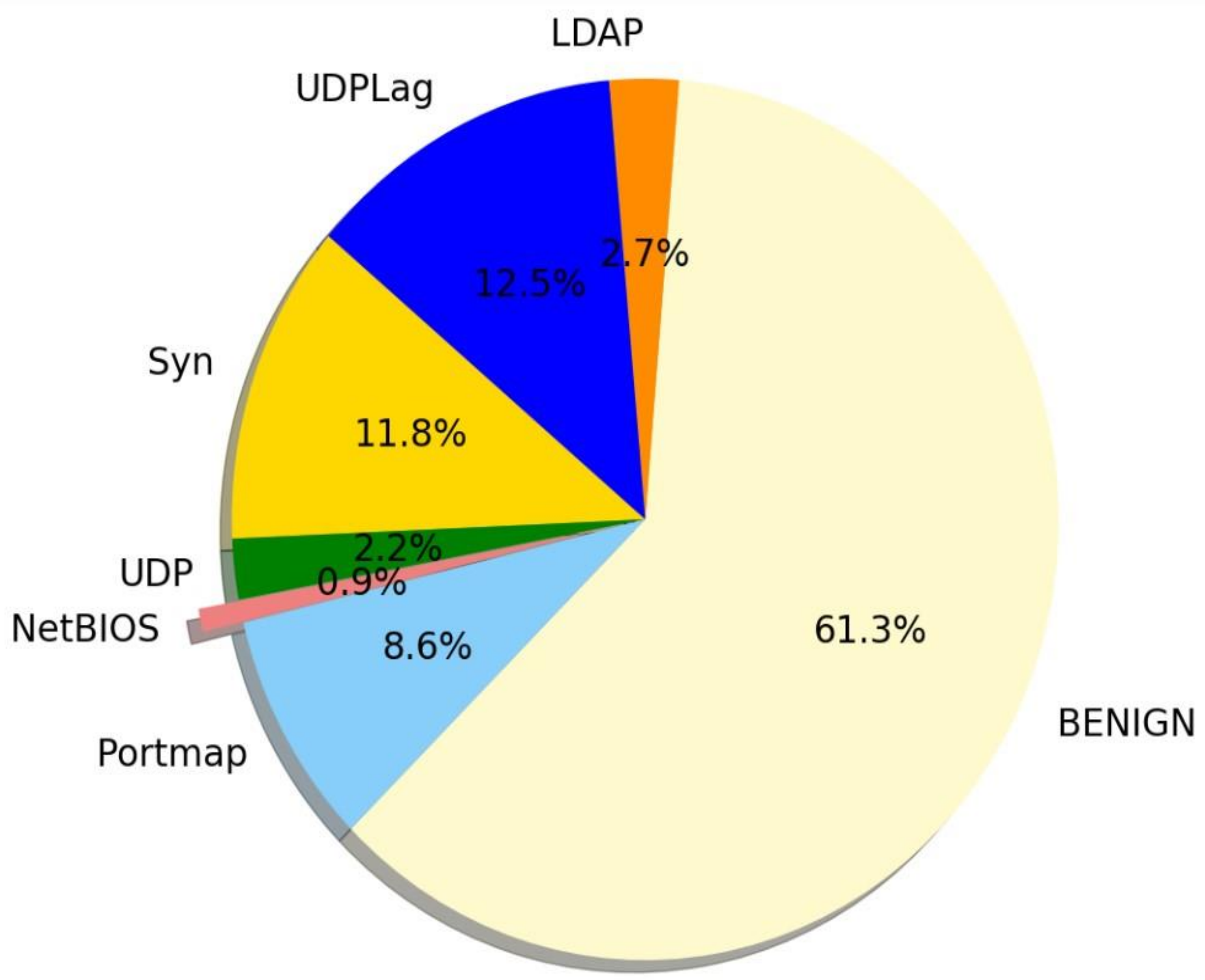


Figure 4.2: Exploratory Data Analysis

CHAPTER - 5

5.1. CORRELATION MATRIX:

A correlation matrix in machine learning is a table that displays the pairwise correlations between several variables in a dataset. The correlation coefficient between any two variables is represented by each cell in the matrix. The degree and direction of the linear link between two variables are measured by the correlation coefficient. For a number of reasons, a correlation matrix can be helpful in machine learning. For instance, it can be used to determine which variables have a strong correlation with one another, which may indicate duplicate or unimportant features. The effectiveness of machine learning models can be enhanced by removing highly correlated characteristics, which can aid in reducing overfitting. Statistical software programs like Python's NumPy and Pandas libraries can be used to compute correlation matrices.

It's worth noting that correlation does not necessarily imply causation, and a high correlation between two variables does not necessarily mean that one variable causes the other. Correlation matrices should be interpreted carefully and in conjunction with other analyses to draw meaningful conclusions about the relationships between variables.

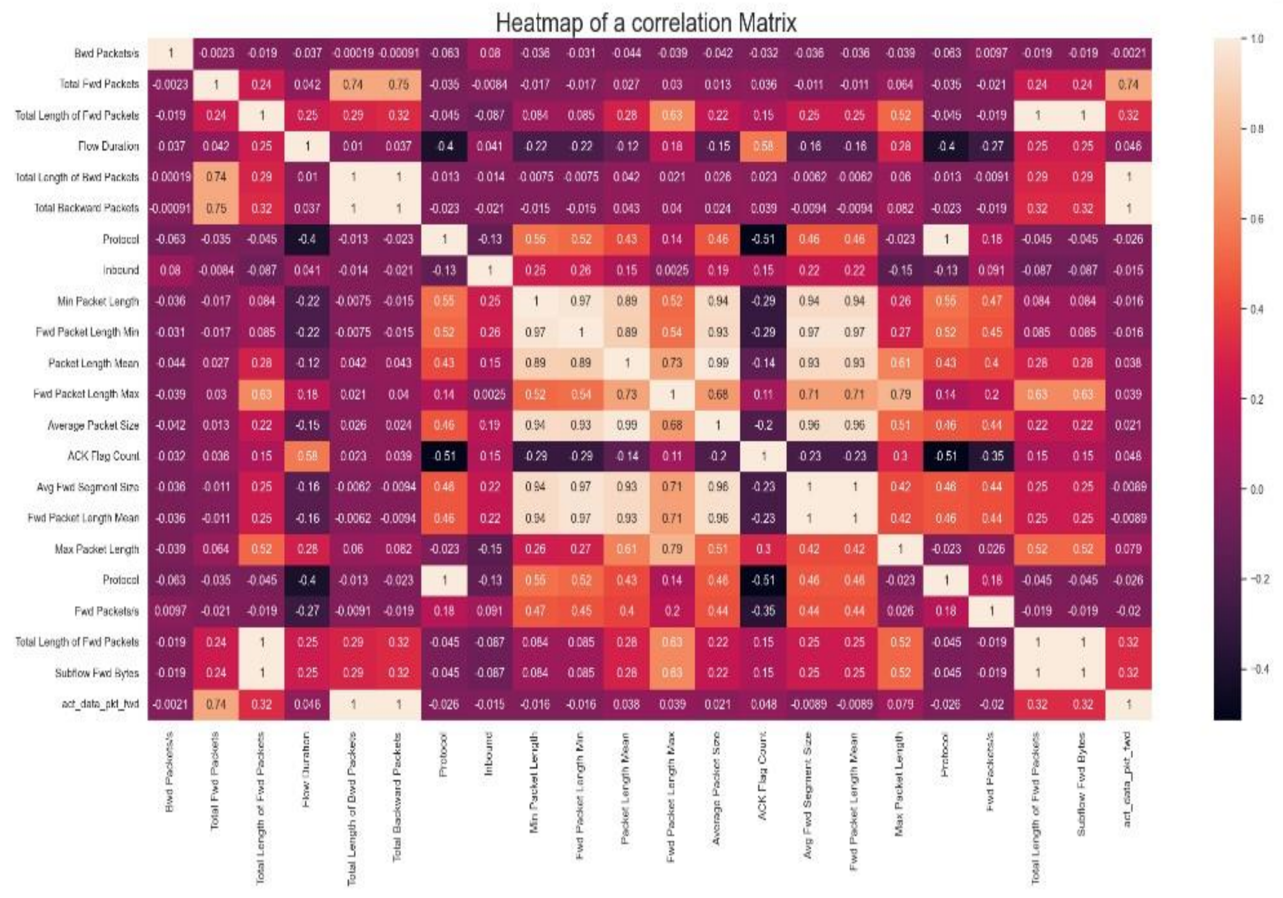


Figure 5.1: Heatmap of a correlation Matrix

5.2. CONFUSION MATRIX:

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the total number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix, as shown below, with 4 values:

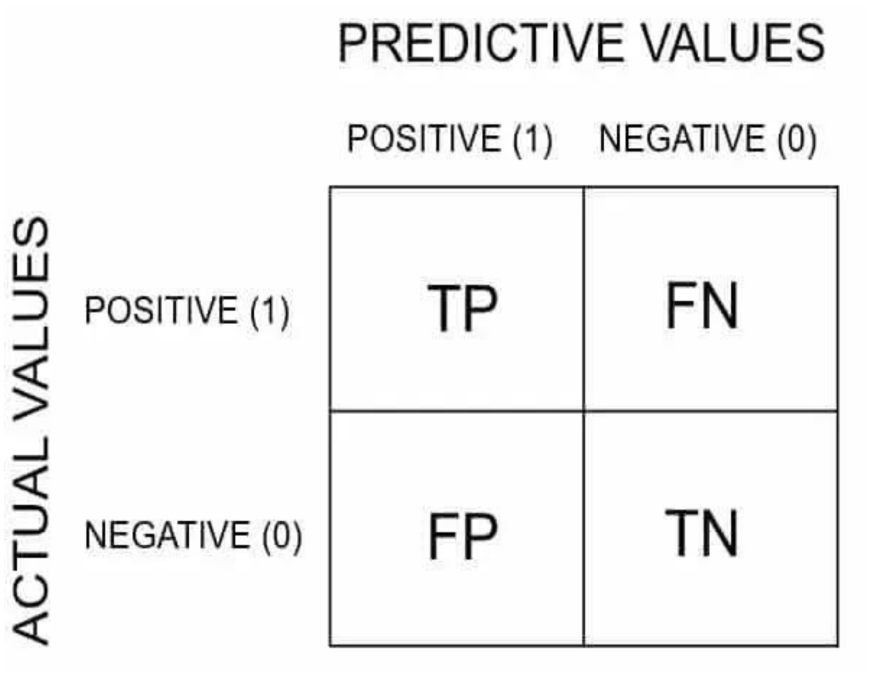


Figure 5.2: Confusion Matrix

The target variable has two values: Positive or Negative

The columns represent the actual values of the target variable

The rows represent the predicted values of the target variable

IMPORTANT TERMS IN A CONFUSION MATRIX

True Positive (TP):

- The predicted value matches the actual value, or the predicted class matches the actual class.

- The actual value was positive, and the model predicted a positive value.

True Negative (TN):

- The predicted value matches the actual value, or the predicted class matches the actual class.
- The actual value was negative, and the model predicted a negative value.

False Positive (FP) – Type I Error:

- The predicted value was falsely predicted.
- The actual value was negative, but the model predicted a positive value.
- Also known as the type I error.

False Negative (FN) – Type II Error:

- The predicted value was falsely predicted.
- The actual value was positive, but the model predicted a negative value.
- Also known as the type II error.

5.3. PRINCIPLE COMPONENT ANALYSIS(PCA):

PCA, or Principal Component Analysis, is a commonly used technique in machine learning for reducing the dimensionality of high-dimensional data.

In PCA, the goal is to transform a dataset with many features into a smaller set of new features, called principal components, which capture the maximum amount of variation in the original dataset. The principal components are calculated in such a way that the first principal component captures the most variation, the second captures the second-most, and so on.

PCA is useful in machine learning because it can help reduce the number of features in a dataset, making it easier to work with and less prone to overfitting. By removing redundant or irrelevant features, PCA can also improve the performance of machine learning algorithms and reduce the computational resources required to train them.

PCA can be implemented using various techniques, including Singular Value Decomposition (SVD) and Eigenvalue Decomposition (EVD). There are also many libraries and frameworks available in various programming languages, such as scikit-learn in Python, that provide easy-to-use functions for implementing PCA.

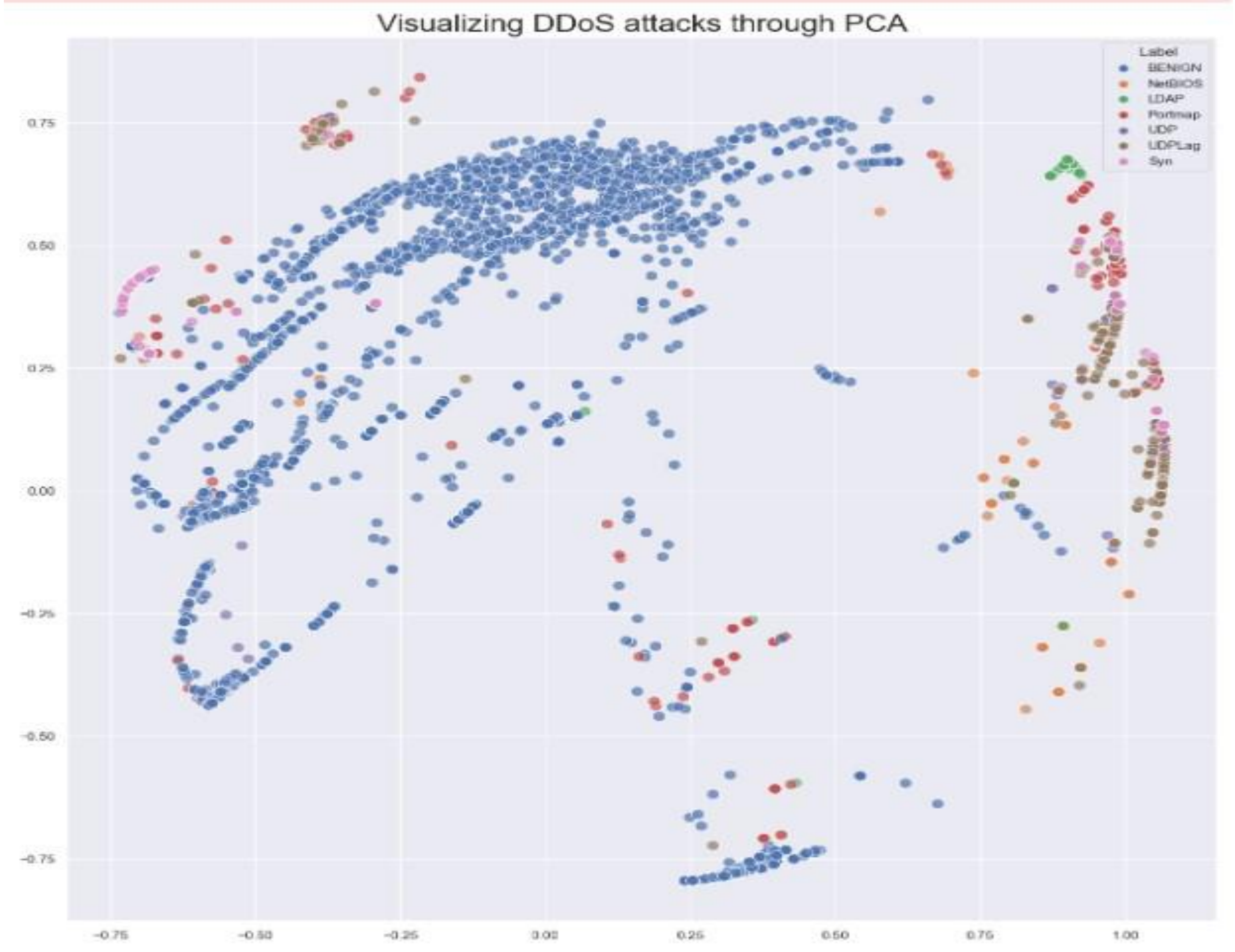


Figure 5.3: Visualization DDoS attacks through PCA

5.4. t-SNE or t-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING:

t-SNE, or t-Distributed Stochastic Neighbor Embedding, is a machine learning technique used for visualization and dimensionality reduction of high-dimensional data. It was first introduced in a paper by Laurens van der Maaten and Geoffrey Hinton in 2008.

t-SNE is particularly useful for visualizing complex datasets with many features, such as images or text, by reducing their dimensionality while still preserving their relationships. It works by modeling the similarities and differences between pairs of data points in high-dimensional space and then mapping them to a lower-dimensional space, such as a two-dimensional or three-dimensional plot.

The main advantage of t-SNE over other dimensionality reduction techniques is that it can preserve the local structure of the data, meaning that data points that are close to each other in high-dimensional space will still be close to each other in the lower-dimensional space. This makes it easier to visually identify clusters or patterns in the data.

t-SNE is implemented in various programming languages and libraries, such as scikit-learn in Python, and is commonly used in machine learning applications such as image recognition, natural language processing, and bioinformatics. However, it should be noted that t-SNE can be computationally intensive and may not be suitable for very large datasets.

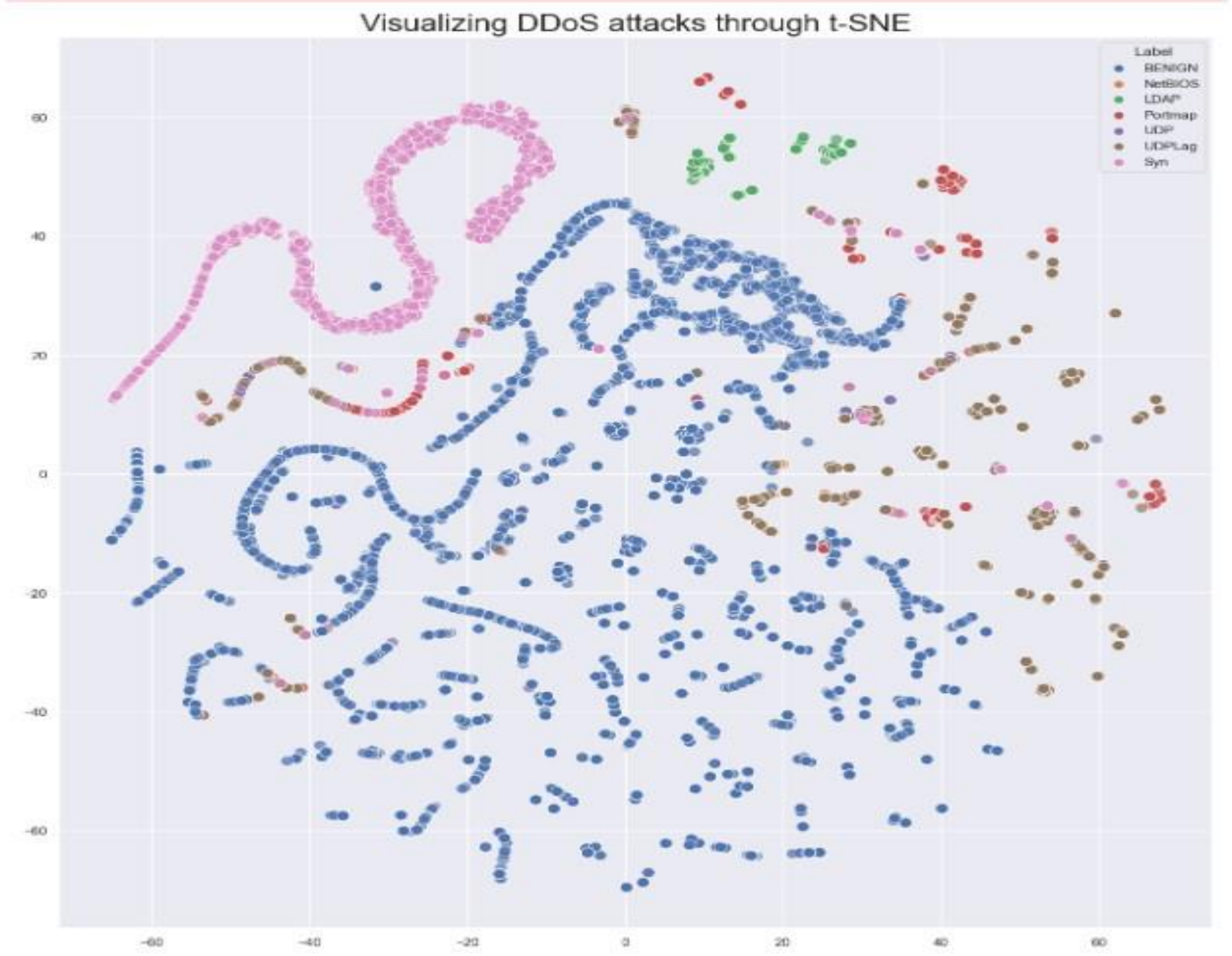


Figure 5.4: Visualization DDoS attacks through t-SNE

5.5. AGGLOMERATIVE CLUSTERING:

Agglomerative clustering is a machine learning technique used for clustering similar data points together based on their proximity to each other in a high-dimensional space. It is a type of hierarchical clustering, which means that it builds a tree-like structure of clusters, starting with each data point as its own cluster and then progressively merging clusters until all data points are in a single cluster.

Agglomerative clustering works by iteratively merging the closest pairs of clusters based on a similarity metric such as Euclidean distance or cosine similarity until all data points belong to a single cluster. The result is a dendrogram, which is a tree-like diagram that shows the hierarchy of clusters and the order in which they were merged.

There are several different linkage criteria that can be used to determine which pairs of clusters to merge at each iteration, including single linkage, complete linkage, and average linkage. Single linkage merges the closest pair of data points in two clusters, complete linkage merges the furthest pair of data points in two clusters, and average linkage merges the pair of clusters with the lowest average distance between all data points in the two clusters.

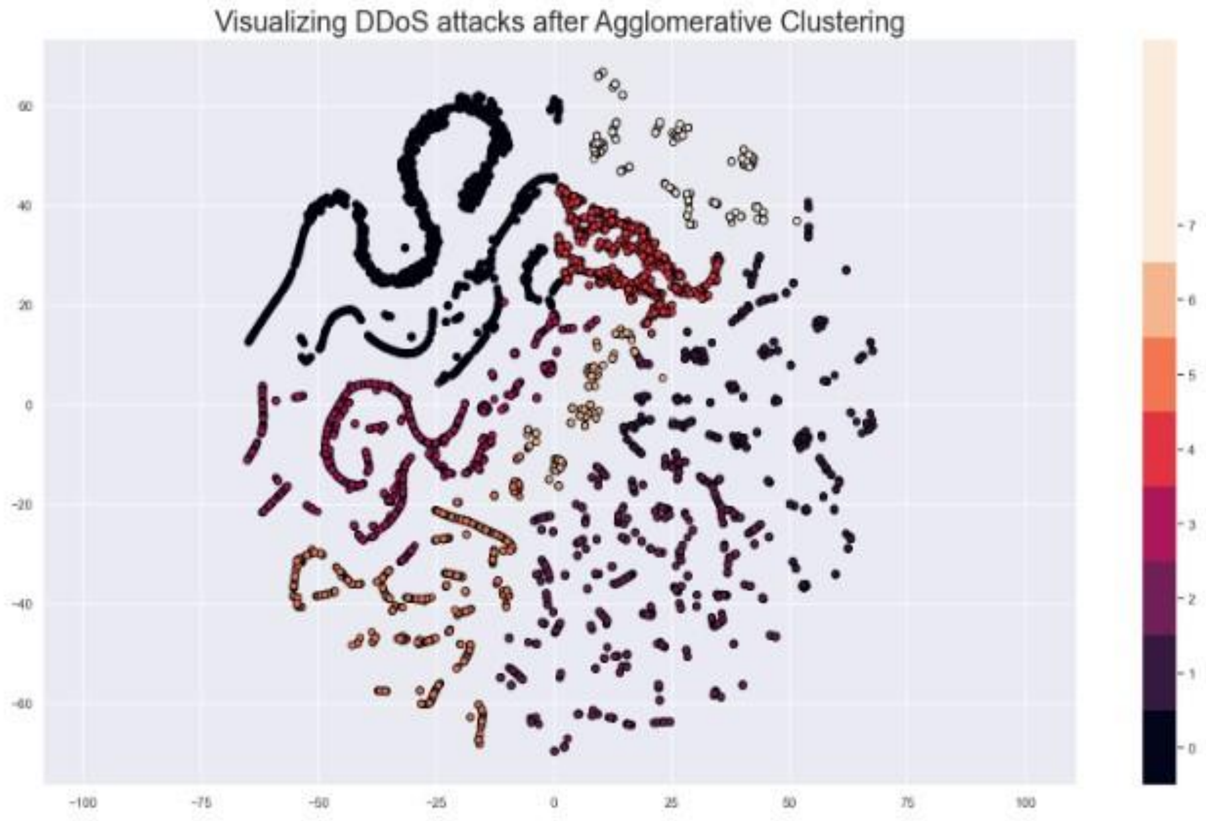


Figure 5.5: Visualization DDoS attacks after Agglomerative clustering

Agglomerative clustering is commonly used in machine learning for exploratory data analysis and pattern recognition, as well as in applications such as image segmentation and bioinformatics. It can also be used as a pre-processing step for other machine learning algorithms by reducing the dimensionality of the data and identifying groups of related features.

5.6. Flow Diagram of Proposed System

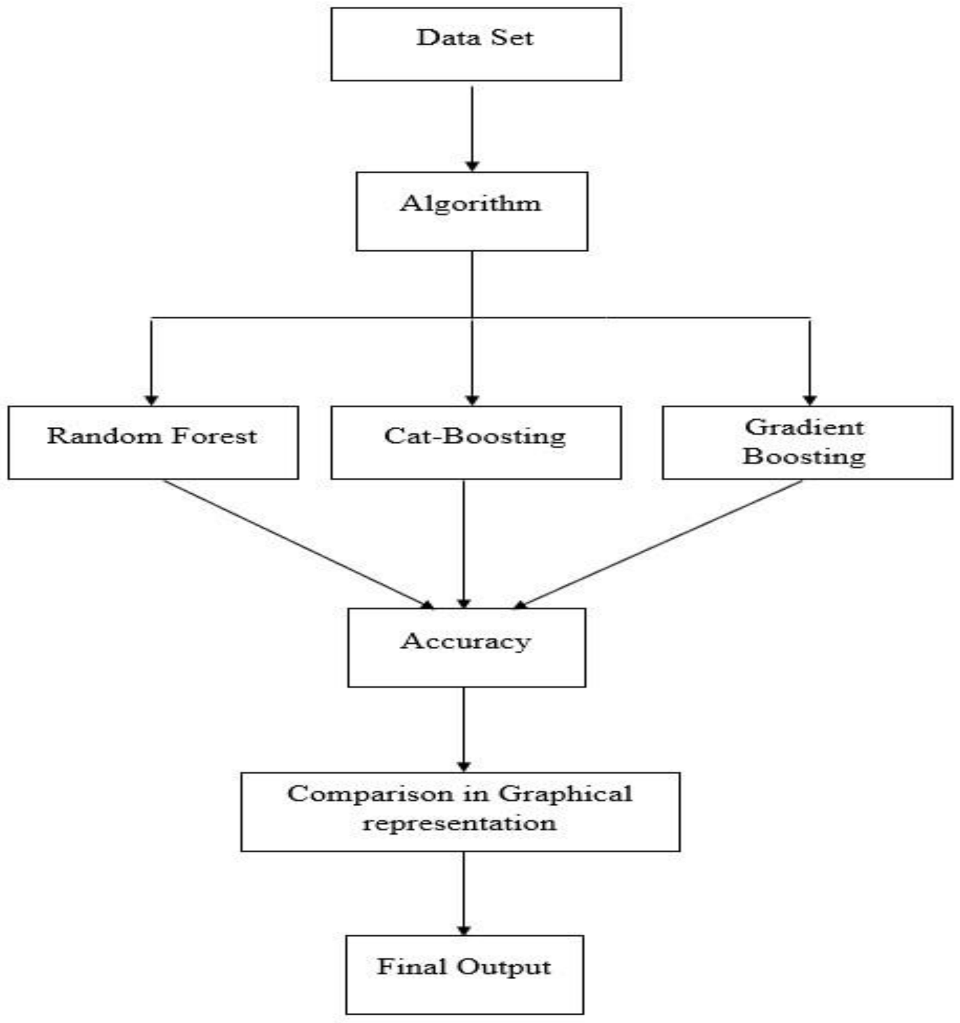


Figure 5.6: Flow Diagram

CHAPTER - 6

6. PROPOSED SYSTEM

6.1. Random Forest

A well-liked machine learning method called Random Forest is utilized for both classification and regression problems. Several decision trees are combined in this ensemble learning technique to produce predictions. The robust and adaptable Random Forest method has the accuracy and generalization ability to handle huge and complicated datasets.

The technique builds a forest of decision trees, each of which is trained using a randomly chosen portion of the training data and features. Each decision tree in the set is trained using a random subset of the input data, and the algorithm uses a random subset of the available features at each split point. This randomization enhances the model's accuracy by lowering overfitting. The Random Forest algorithm aggregates all of the forest's trees' predictions during the prediction phase to get the final forecast. For classification issues, the algorithm determines the mode of all the anticipated classes, and for regression problems, the mean or median of the expected values.

When compared to other machine learning methods, Random Forest provides a number of benefits. It can handle high-dimensional data with numerous features and missing values, and feature scaling is not necessary. Additionally, it can offer feature importance scores that can be used to comprehend the relative weights given to various features during the prediction process. Overall, Random Forest is a powerful and flexible machine learning algorithm that can be applied to a wide range of tasks and is suitable for both beginners and experts in machine learning.

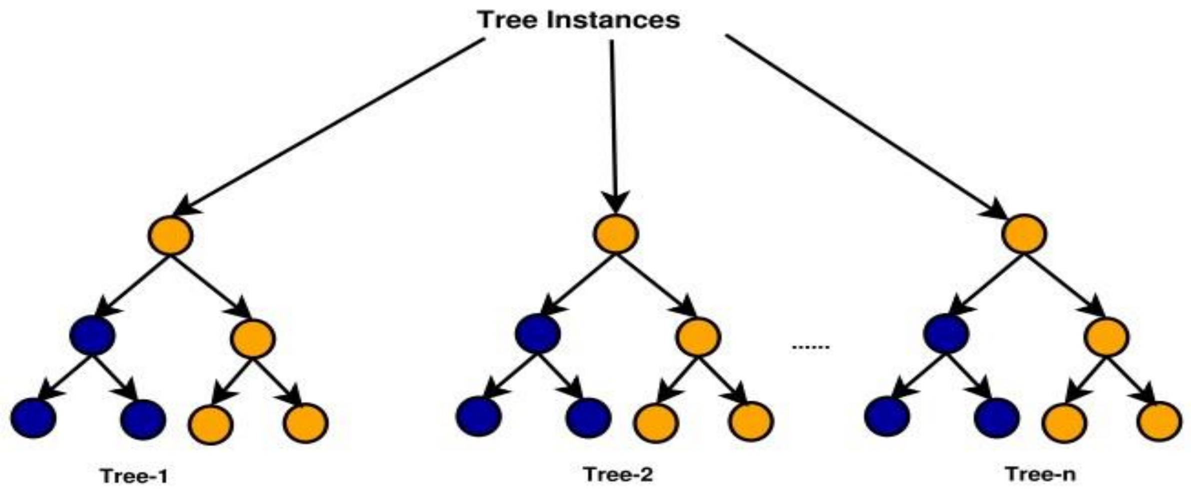
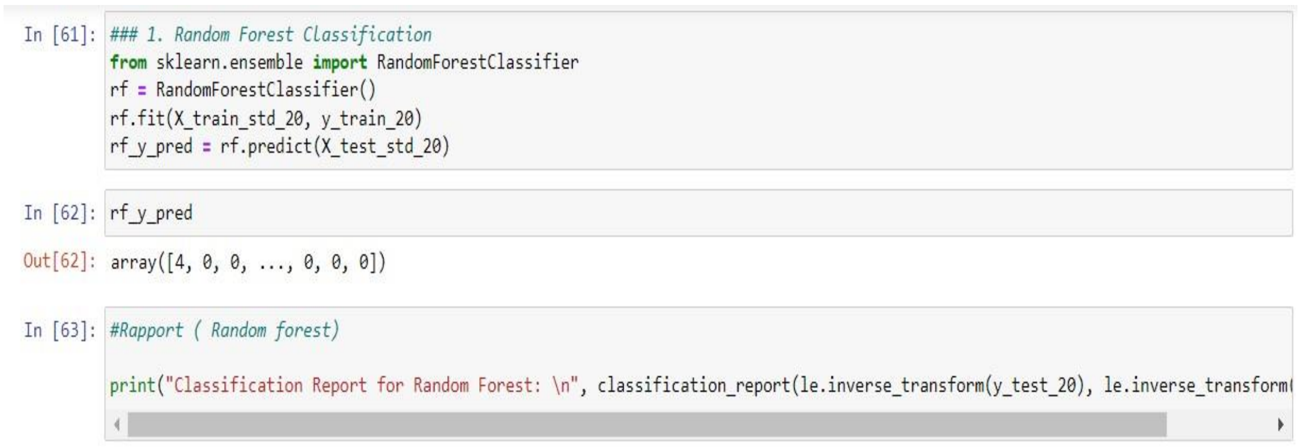
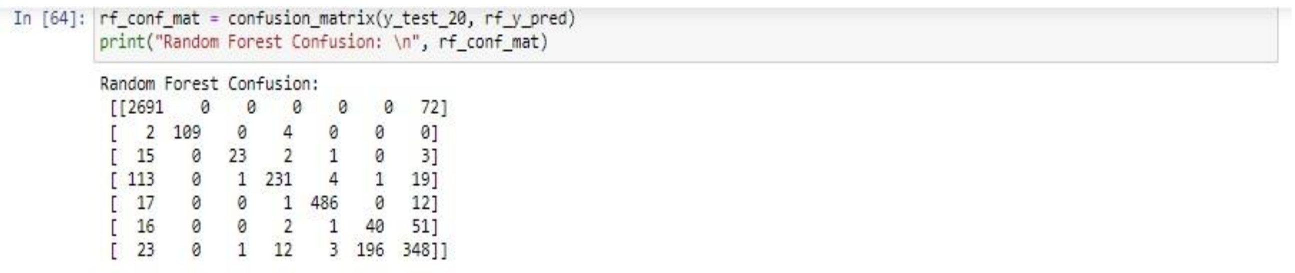


Figure 6.1: Random Forest

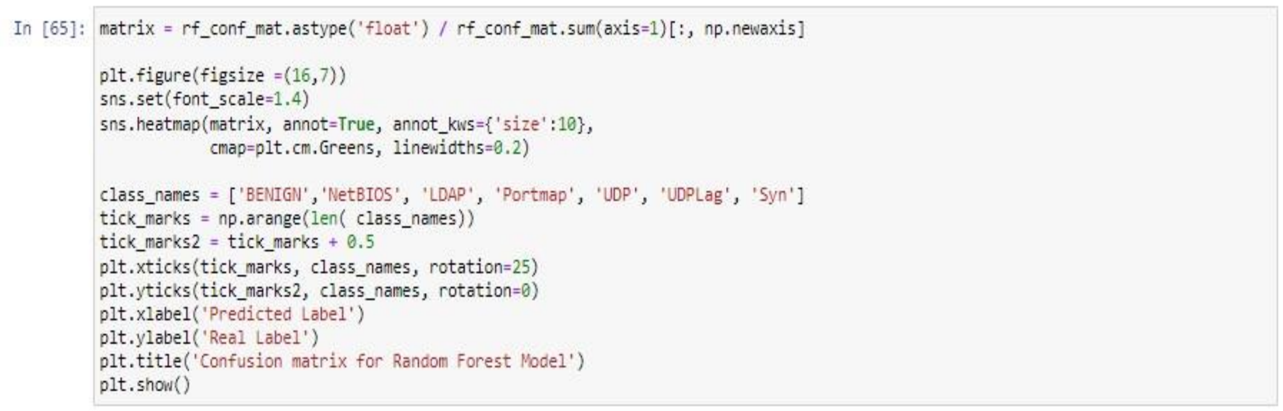
Coding and its Accuracy :



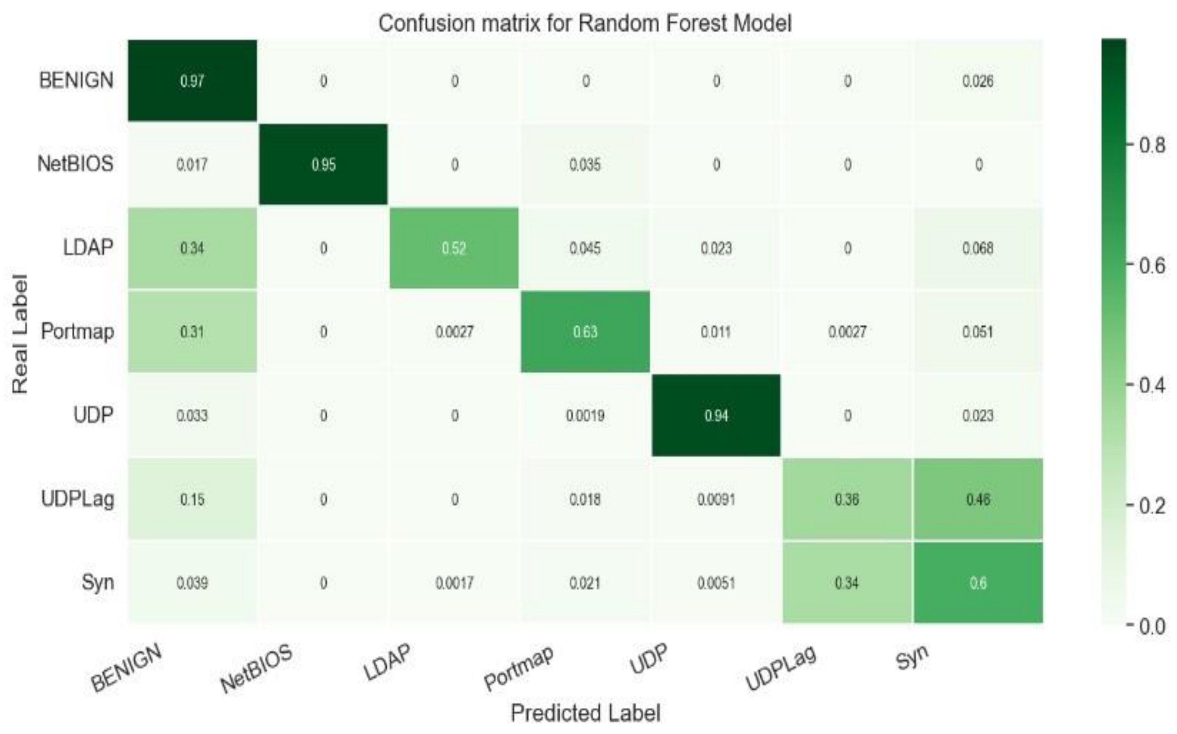
```
In [61]: ### 1. Random Forest Classification
         from sklearn.ensemble import RandomForestClassifier
         rf = RandomForestClassifier()
         rf.fit(X_train_std_20, y_train_20)
         rf_y_pred = rf.predict(X_test_std_20)

In [62]: rf_y_pred

Out[62]: array([4, 0, 0, ..., 0, 0, 0])

In [63]: #Rapport ( Random forest)

         print("Classification Report for Random Forest: \n", classification_report(le.inverse_transform(y_test_20), le.inverse_transform(
```



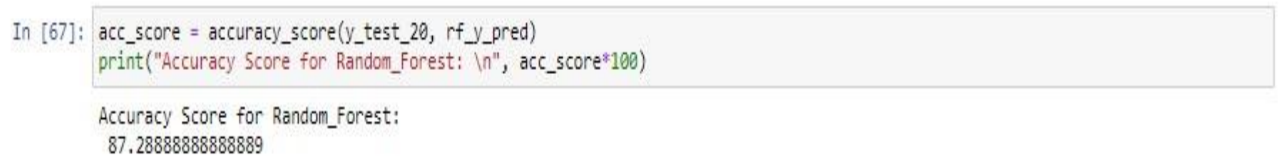
```
In [64]: rf_conf_mat = confusion_matrix(y_test_20, rf_y_pred)
         print("Random Forest Confusion: \n", rf_conf_mat)

         Random Forest Confusion:
          [[2691    0    0    0    0    0   72]
          [   2  109    0    4    0    0    0]
          [  15    0   23    2    1    0    3]
          [ 113    0    1  231    4    1   19]
          [  17    0    0    1  486    0   12]
          [  16    0    0    2    1   40   51]
          [  23    0    1   12    3  196  348]]
```



```
In [65]: matrix = rf_conf_mat.astype('float') / rf_conf_mat.sum(axis=1)[:, np.newaxis]

         plt.figure(figsize =(16,7))
         sns.set(font_scale=1.4)
         sns.heatmap(matrix, annot=True, annot_kws={'size':10},
                     cmap=plt.cm.Greens, linewidths=0.2)

         class_names = ['BENIGN','NetBIOS', 'LDAP', 'Portmap', 'UDP', 'UDPLag', 'Syn']
         tick_marks = np.arange(len( class_names))
         tick_marks2 = tick_marks + 0.5
         plt.xticks(tick_marks, class_names, rotation=25)
         plt.yticks(tick_marks2, class_names, rotation=0)
         plt.xlabel('Predicted Label')
         plt.ylabel('Real Label')
         plt.title('Confusion matrix for Random Forest Model')
         plt.show()
```



	BENIGN	NetBIOS	LDAP	Portmap	UDP	UDPLag	Syn
BENIGN	0.97	0	0	0	0	0	0.026
NetBIOS	0.017	0.95	0	0.035	0	0	0
LDAP	0.34	0	0.52	0.045	0.023	0	0.068
Portmap	0.31	0	0.0027	0.63	0.011	0.0027	0.051
UDP	0.033	0	0	0.0019	0.94	0	0.023
UDPLag	0.15	0	0	0.018	0.0091	0.36	0.46
Syn	0.039	0	0.0017	0.021	0.0051	0.34	0.6



```
In [67]: acc_score = accuracy_score(y_test_20, rf_y_pred)
         print("Accuracy Score for Random_Forest: \n", acc_score*100)

         Accuracy Score for Random_Forest:
          87.28888888888889
```

Figure 6.1.1: Coding and Accuracy

6.2. Cat-Boosting

The gradient boosting framework, on which Cat-Boost is based, involves iteratively adding weak learners to a model in order to increase its predictive potential. However, Cat-Boost employs a cutting-edge strategy known as "ordered boosting" in contrast to conventional gradient boosting algorithms to handle category variables more skillfully. Using an ordering mechanism, ordered boosting transforms categorical variables into numerical values while maintaining the relationships between the categories.

Application areas where Cat-Boost has been utilized successfully include fraud detection, predicting customer turnover, and picture categorization. Cat-Boost can be used to categorize network traffic data and precisely identify DDoS attacks in the context of DDoS attack detection. It is a helpful tool for analysis because it can handle category variables and handle missing data makes it a useful tool for analyzing network traffic data, which often contains a mix of numerical and categorical variables.

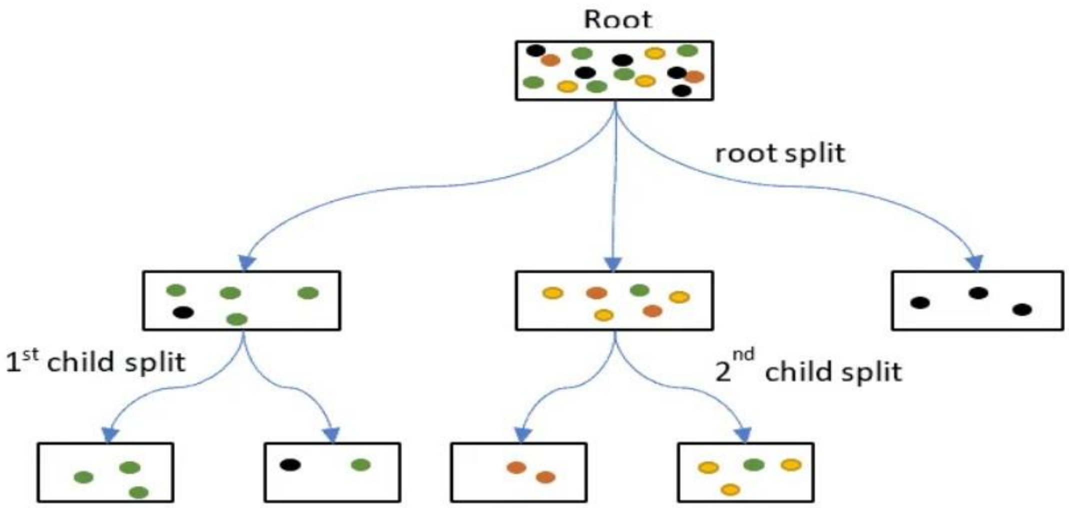
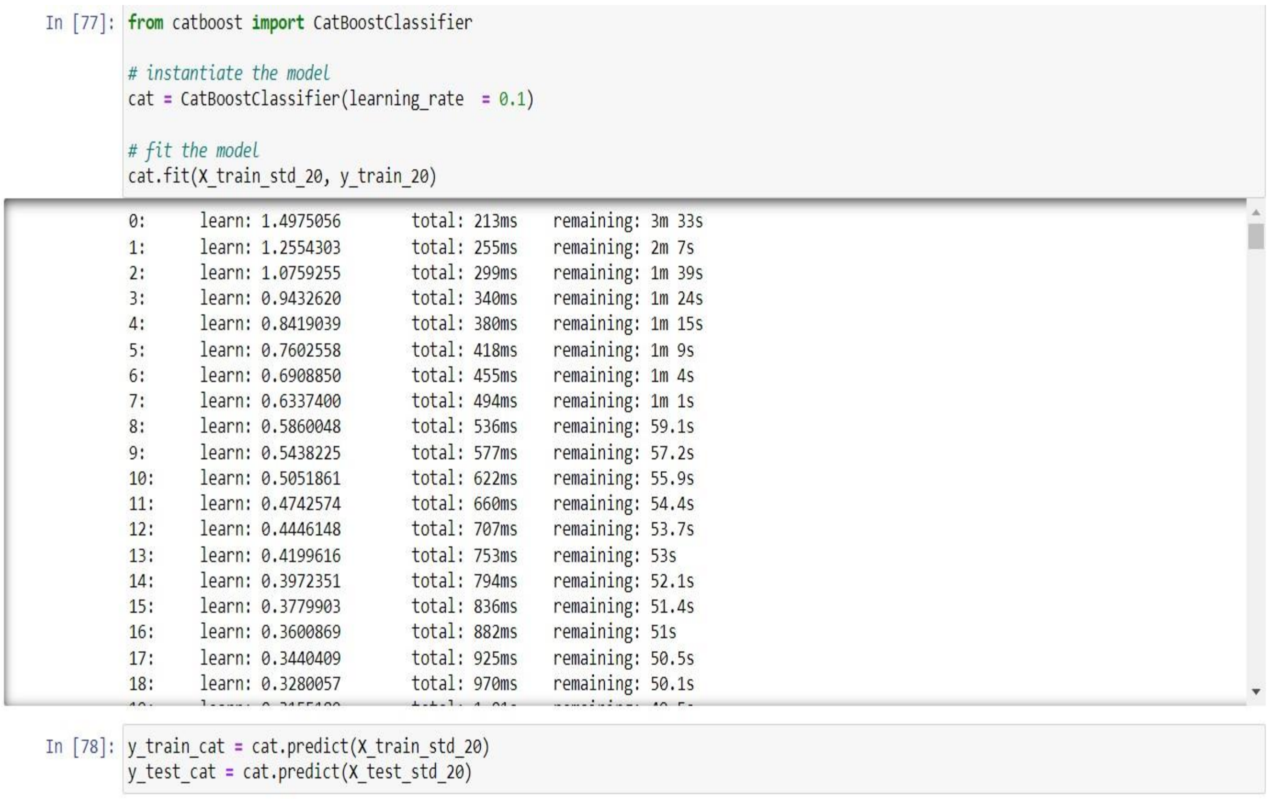


Figure 6.2: Cat-Boosting

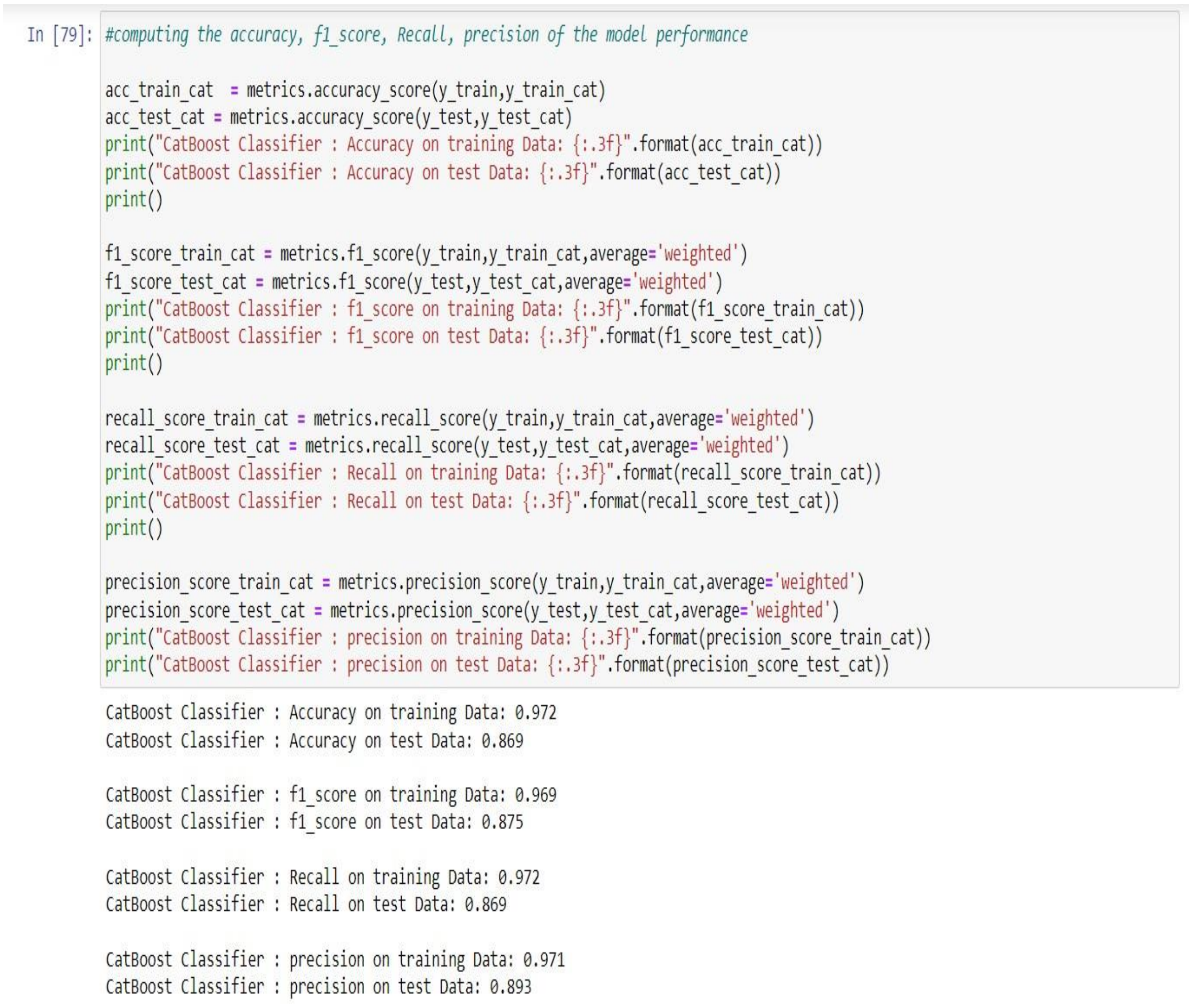
Coding and its Accuracy :



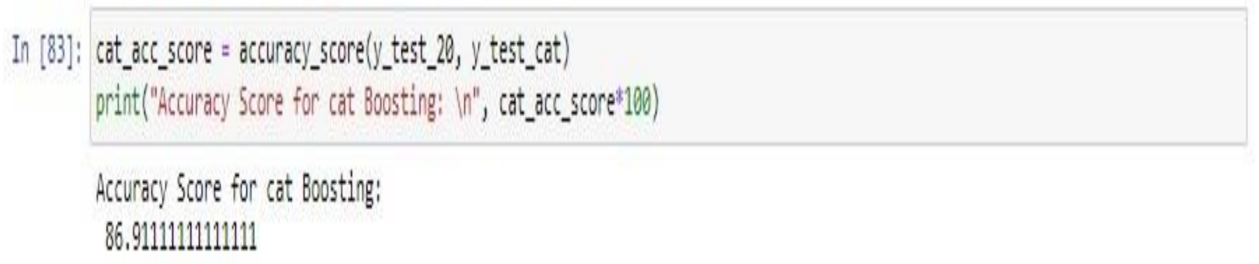
```
In [77]: from catboost import CatBoostClassifier

         # instantiate the model
         cat = CatBoostClassifier(learning_rate  = 0.1)

         # fit the model
         cat.fit(X_train_std_20, y_train_20)
```

```
0:      learn: 1.4975056        total: 213ms    remaining: 3m 33s
1:      learn: 1.2554303        total: 255ms    remaining: 2m 7s
2:      learn: 1.0759255        total: 299ms    remaining: 1m 39s
3:      learn: 0.9432620        total: 340ms    remaining: 1m 24s
4:      learn: 0.8419039        total: 380ms    remaining: 1m 15s
5:      learn: 0.7602558        total: 418ms    remaining: 1m 9s
6:      learn: 0.6908850        total: 455ms    remaining: 1m 4s
7:      learn: 0.6337400        total: 494ms    remaining: 1m 1s
8:      learn: 0.5860048        total: 536ms    remaining: 59.1s
9:      learn: 0.5438225        total: 577ms    remaining: 57.2s
10:     learn: 0.5051861        total: 622ms    remaining: 55.9s
11:     learn: 0.4742574        total: 660ms    remaining: 54.4s
12:     learn: 0.4446148        total: 707ms    remaining: 53.7s
13:     learn: 0.4199616        total: 753ms    remaining: 53s
14:     learn: 0.3972351        total: 794ms    remaining: 52.1s
15:     learn: 0.3779903        total: 836ms    remaining: 51.4s
16:     learn: 0.3600869        total: 882ms    remaining: 51s
17:     learn: 0.3440409        total: 925ms    remaining: 50.5s
18:     learn: 0.3280057        total: 970ms    remaining: 50.1s
```

```
In [78]: y_train_cat = cat.predict(X_train_std_20)
         y_test_cat = cat.predict(X_test_std_20)
```



```
In [79]: #computing the accuracy, f1_score, Recall, precision of the model performance

         acc_train_cat  = metrics.accuracy_score(y_train,y_train_cat)
         acc_test_cat = metrics.accuracy_score(y_test,y_test_cat)
         print("CatBoost Classifier : Accuracy on training Data: {:.3f}".format(acc_train_cat))
         print("CatBoost Classifier : Accuracy on test Data: {:.3f}".format(acc_test_cat))
         print()

         f1_score_train_cat = metrics.f1_score(y_train,y_train_cat,average='weighted')
         f1_score_test_cat = metrics.f1_score(y_test,y_test_cat,average='weighted')
         print("CatBoost Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_cat))
         print("CatBoost Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_cat))
         print()

         recall_score_train_cat = metrics.recall_score(y_train,y_train_cat,average='weighted')
         recall_score_test_cat = metrics.recall_score(y_test,y_test_cat,average='weighted')
         print("CatBoost Classifier : Recall on training Data: {:.3f}".format(recall_score_train_cat))
         print("CatBoost Classifier : Recall on test Data: {:.3f}".format(recall_score_test_cat))
         print()

         precision_score_train_cat = metrics.precision_score(y_train,y_train_cat,average='weighted')
         precision_score_test_cat = metrics.precision_score(y_test,y_test_cat,average='weighted')
         print("CatBoost Classifier : precision on training Data: {:.3f}".format(precision_score_train_cat))
         print("CatBoost Classifier : precision on test Data: {:.3f}".format(precision_score_test_cat))

         CatBoost Classifier : Accuracy on training Data: 0.972
         CatBoost Classifier : Accuracy on test Data: 0.869

         CatBoost Classifier : f1_score on training Data: 0.969
         CatBoost Classifier : f1_score on test Data: 0.875

         CatBoost Classifier : Recall on training Data: 0.972
         CatBoost Classifier : Recall on test Data: 0.869

         CatBoost Classifier : precision on training Data: 0.971
         CatBoost Classifier : precision on test Data: 0.893
```



```
In [83]: cat_acc_score = accuracy_score(y_test_20, y_test_cat)
         print("Accuracy Score for cat Boosting: \n", cat_acc_score*100)

         Accuracy Score for cat Boosting:
          86.91111111111111
```

Figure 6.2.1. Coding and Accuracy

6.3. Gradient Boosting

Building a series of decision trees in which each new tree is trained to anticipate the residual error of the prior tree is how the method operates. The approach iteratively enhances the model's performance during training by using gradient descent to optimize a loss function. In order to minimize the loss function, it, in other words, modifies the weights of the decision trees based on the difference between the expected output and the actual output.

Gradient Boosting's strength rests in its capacity to merge weak prediction models into a powerful and precise model. Each decision tree in the series gains knowledge from the residual mistakes of the previous tree, which lowers the model's bias and variance and produces better accuracy. There are several advantages to using Gradient Boosting. It can handle missing values, outliers, and skewed data, and can also handle categorical and numerical data. Additionally, it can provide feature importance scores, which can help in understanding the importance of each feature in the prediction process. Gradient Boosting can be computationally costly and prone to overfitting, though, if the number of trees or their depth is excessively high. To obtain best performance, hyperparameter adjustment is crucial.

Overall, Gradient Boosting is a robust and popular machine learning technique that may be utilized for a wide range of tasks. Many data scientists and analysts find it to be an invaluable tool because of its capacity to manage complicated and heterogeneous data and generate precise predictions.

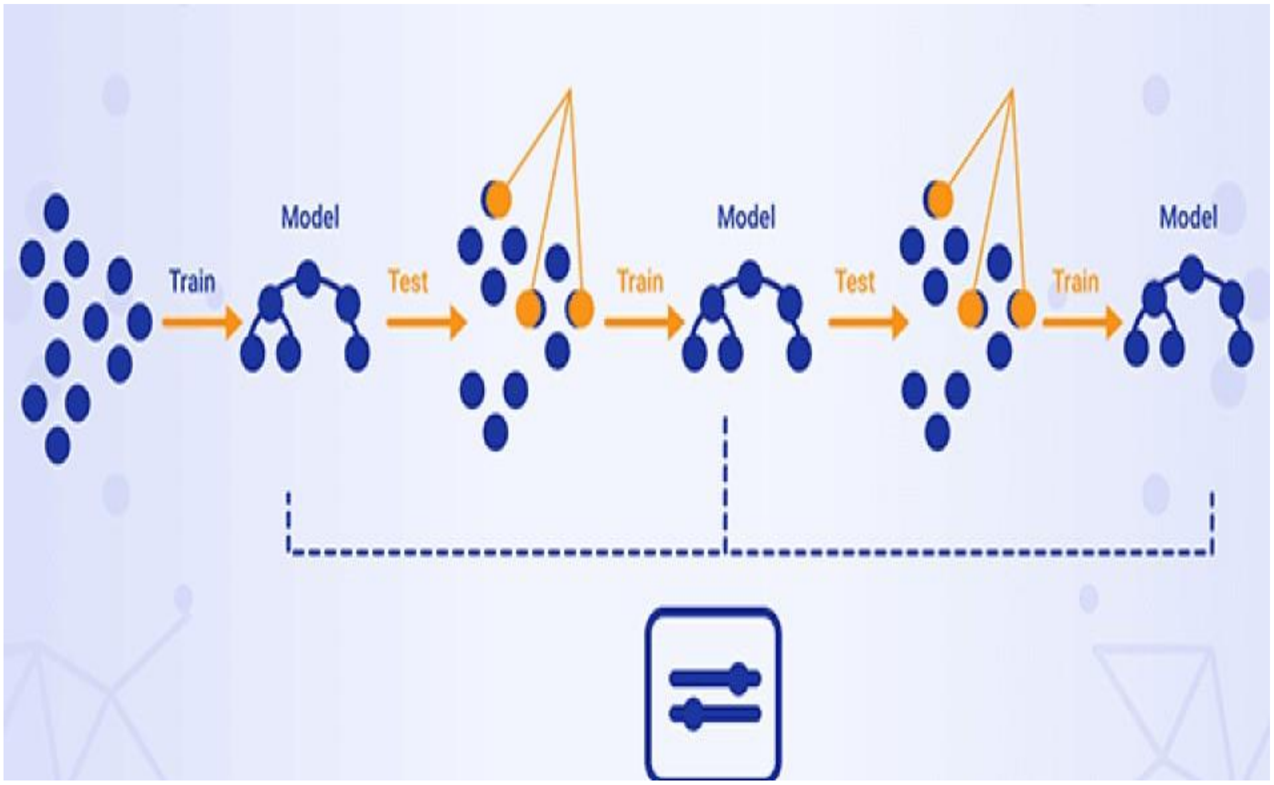
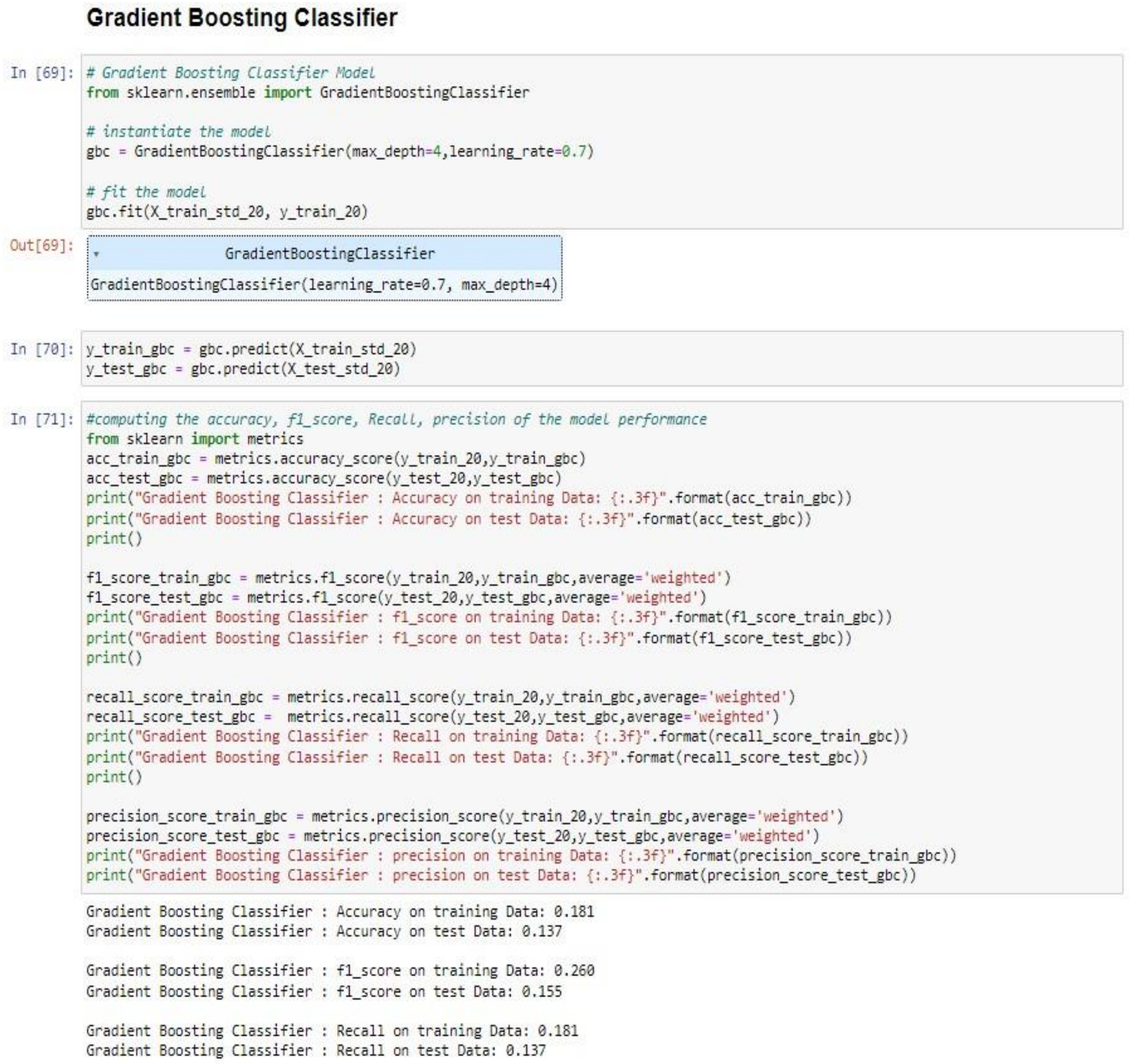


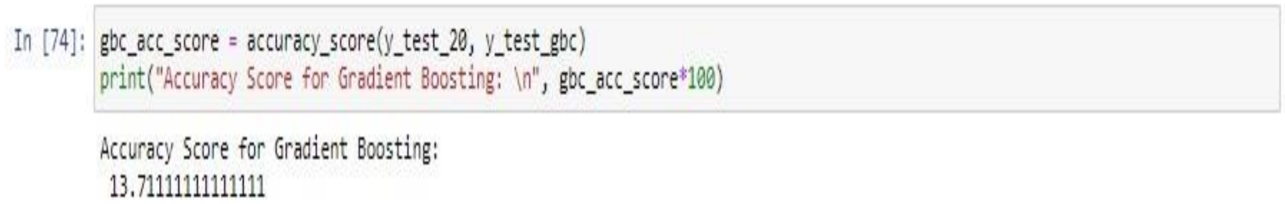
Figure 6.3: Gradient Boosting

Coding and its Accuracy :



Gradient Boosting Classifier

```
In [69]: # Gradient Boosting Classifier Model
         from sklearn.ensemble import GradientBoostingClassifier

         # instantiate the model
         gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)

         # fit the model
         gbc.fit(X_train_std_20, y_train_20)

Out[69]:              GradientBoostingClassifier
         GradientBoostingClassifier(learning_rate=0.7, max_depth=4)

In [70]: y_train_gbc = gbc.predict(X_train_std_20)
         y_test_gbc = gbc.predict(X_test_std_20)

In [71]: #computing the accuracy, f1_score, Recall, precision of the model performance
         from sklearn import metrics
         acc_train_gbc = metrics.accuracy_score(y_train_20,y_train_gbc)
         acc_test_gbc = metrics.accuracy_score(y_test_20,y_test_gbc)
         print("Gradient Boosting Classifier : Accuracy on training Data: {:.3f}".format(acc_train_gbc))
         print("Gradient Boosting Classifier : Accuracy on test Data: {:.3f}".format(acc_test_gbc))
         print()

         f1_score_train_gbc = metrics.f1_score(y_train_20,y_train_gbc,average='weighted')
         f1_score_test_gbc = metrics.f1_score(y_test_20,y_test_gbc,average='weighted')
         print("Gradient Boosting Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_gbc))
         print("Gradient Boosting Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_gbc))
         print()

         recall_score_train_gbc = metrics.recall_score(y_train_20,y_train_gbc,average='weighted')
         recall_score_test_gbc =  metrics.recall_score(y_test_20,y_test_gbc,average='weighted')
         print("Gradient Boosting Classifier : Recall on training Data: {:.3f}".format(recall_score_train_gbc))
         print("Gradient Boosting Classifier : Recall on test Data: {:.3f}".format(recall_score_test_gbc))
         print()

         precision_score_train_gbc = metrics.precision_score(y_train_20,y_train_gbc,average='weighted')
         precision_score_test_gbc = metrics.precision_score(y_test_20,y_test_gbc,average='weighted')
         print("Gradient Boosting Classifier : precision on training Data: {:.3f}".format(precision_score_train_gbc))
         print("Gradient Boosting Classifier : precision on test Data: {:.3f}".format(precision_score_test_gbc))

         Gradient Boosting Classifier : Accuracy on training Data: 0.181
         Gradient Boosting Classifier : Accuracy on test Data: 0.137

         Gradient Boosting Classifier : f1_score on training Data: 0.260
         Gradient Boosting Classifier : f1_score on test Data: 0.155

         Gradient Boosting Classifier : Recall on training Data: 0.181
         Gradient Boosting Classifier : Recall on test Data: 0.137
```



```
In [74]: gbc_acc_score = accuracy_score(y_test_20, y_test_gbc)
         print("Accuracy Score for Gradient Boosting: \n", gbc_acc_score*100)

         Accuracy Score for Gradient Boosting:
          13.7111111111111
```

Figure 6.3.1. Coding and Accuracy

CHAPTER - 7

7. RESULT AND OUTPUT

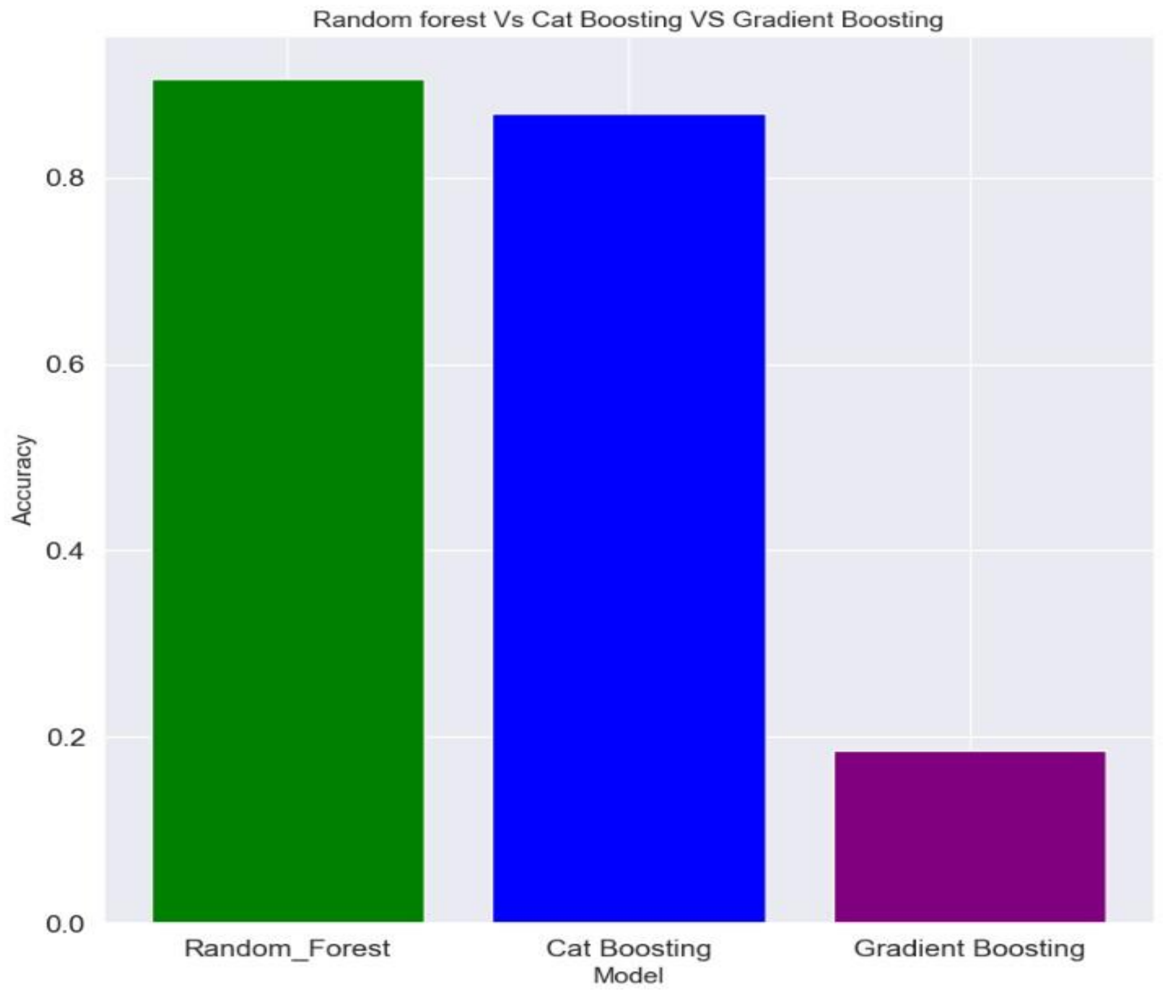


Figure 7.1 Result and Graph

In this proposed system, We have trained the Random Forest algorithm which has high accuracy than comparing other algorithms like Cat Boosting and Gradient Boosting.

So we conclude that our trained Random Forest model as best model for DDoS detection. Cat Boosting can also be used for DDoS detection since it has accuracy of 86.91

Table 7.1: Algorithm and Accuracy

Algorithm	**Accuracy**
Random Forest	**87.288888888889**
Cat Boosting	**86.911111111111**
Gradient Boosting	**13.711111111111**

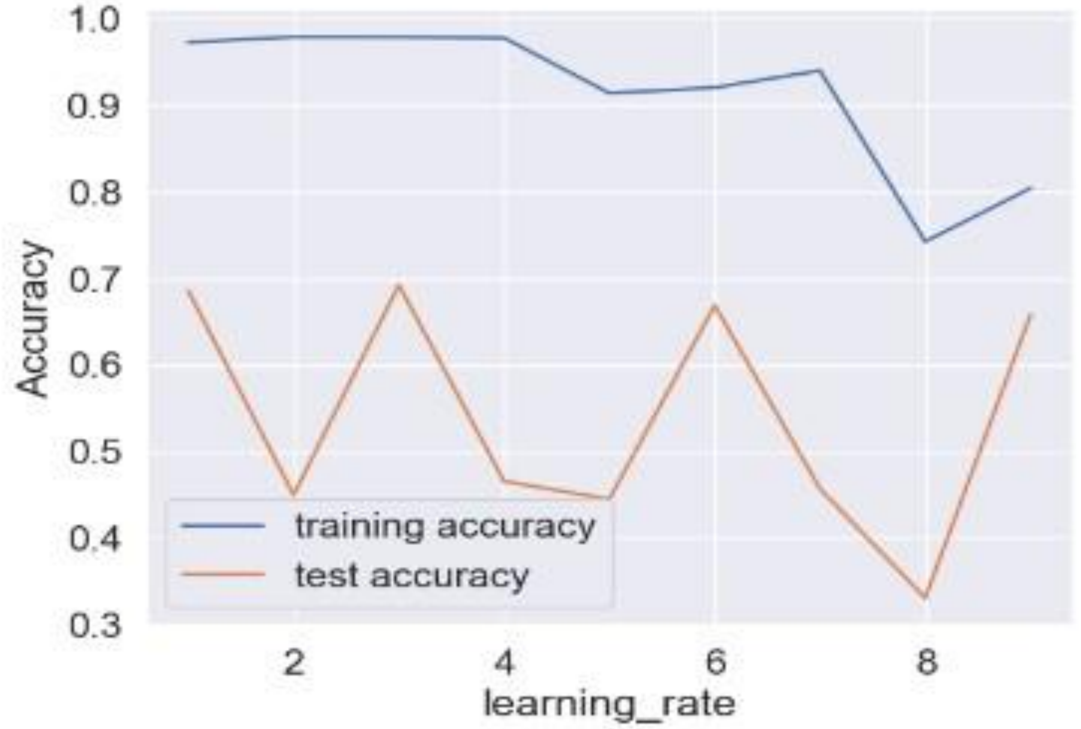


Figure 7.2: Gradient Boosting Graph

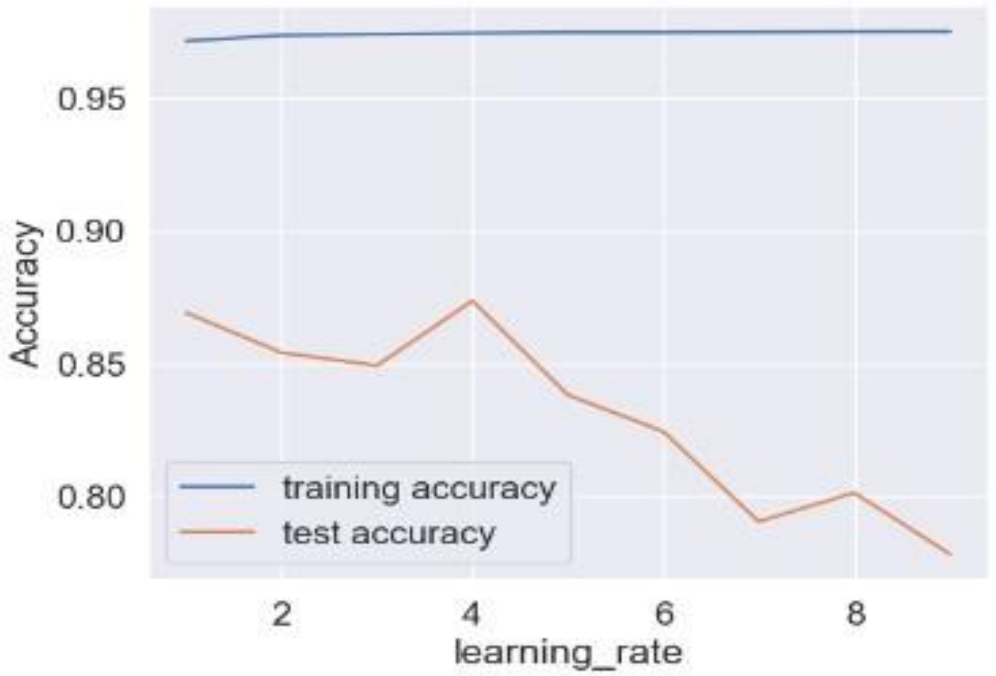


Figure 7.3: Cat-Boosting Graph

7.1. Merits and Demerits of Work :

Merits :

- In this proposed system, used boosting algorithm over bagging algorithm
- Boosting algorithm works sequentially whether bagging algorithm works parallel.
- So it's check sequentially and reduce the errors more than bagging algorithm.

Demerits :

- By using Boosting algorithm, it is hard to implement in real-time due to the increased complexity of the algorithm.

CHAPTER - 8

8. CONCLUSION

DDoS attacks pose a substantial threat to modern networks, and prompt detection is essential to avoiding severe service and infrastructure interruptions. The ability of machine learning algorithms to quickly evaluate vast amounts of network traffic data and precisely identify patterns and abnormalities related to attacks makes them a promising tool for DDoS attack detection. DDoS attacks pose a substantial threat to modern networks, and prompt detection is essential to avoiding severe service and infrastructure interruptions. The ability of machine learning algorithms to rapidly evaluate huge amounts of network traffic data and precisely identify trends and attacks makes them a promising tool for DDoS attack detection. Several important advancements in machine learning and network security are anticipated to have a significant impact on DDoS attack detection in the future. New machine learning techniques and models that can efficiently manage remote networks and diverse data sources are likely to be developed as a result of the expanding use of cloud computing and edge computing.

These new algorithms must be able to process enormous amounts of data in real-time and recognise tiny patterns and anomalies that could be signs of DDoS attacks. Deep learning and neural network developments will probably make DDoS attack detection more sophisticated and precise. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs), two types of deep learning algorithms, have demonstrated promising performance in image and speech recognition applications.

8.1. Future Work :

- The future implementation will focuses on implementing greater accuracy
- To make a real time system analysis to curb the denial of service

8.2. REFERENCES

1. S. Garg, K. Kaur, N. Kumar and J. J. P. C. Rodrigues, "Hybrid Deep-Learning-Based Anomaly Detection Scheme for Suspicious Flow Detection in SDN: A Social Multimedia Perspective," in IEEE Transactions on Multimedia, vol. 21, no. 3, pp. 566-578, March 2019, doi: 10.1109/TMM.2019.2893549.

2. M. Zekri, S. E. Kafhali, N. Aboutabit and Y. Saadi, "DDoS attack detection using machine learning techniques in cloud computing environments," 2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech), Rabat, Morocco, 2017, pp. 1-7, doi: 10.1109/CloudTech.2017.8284731.

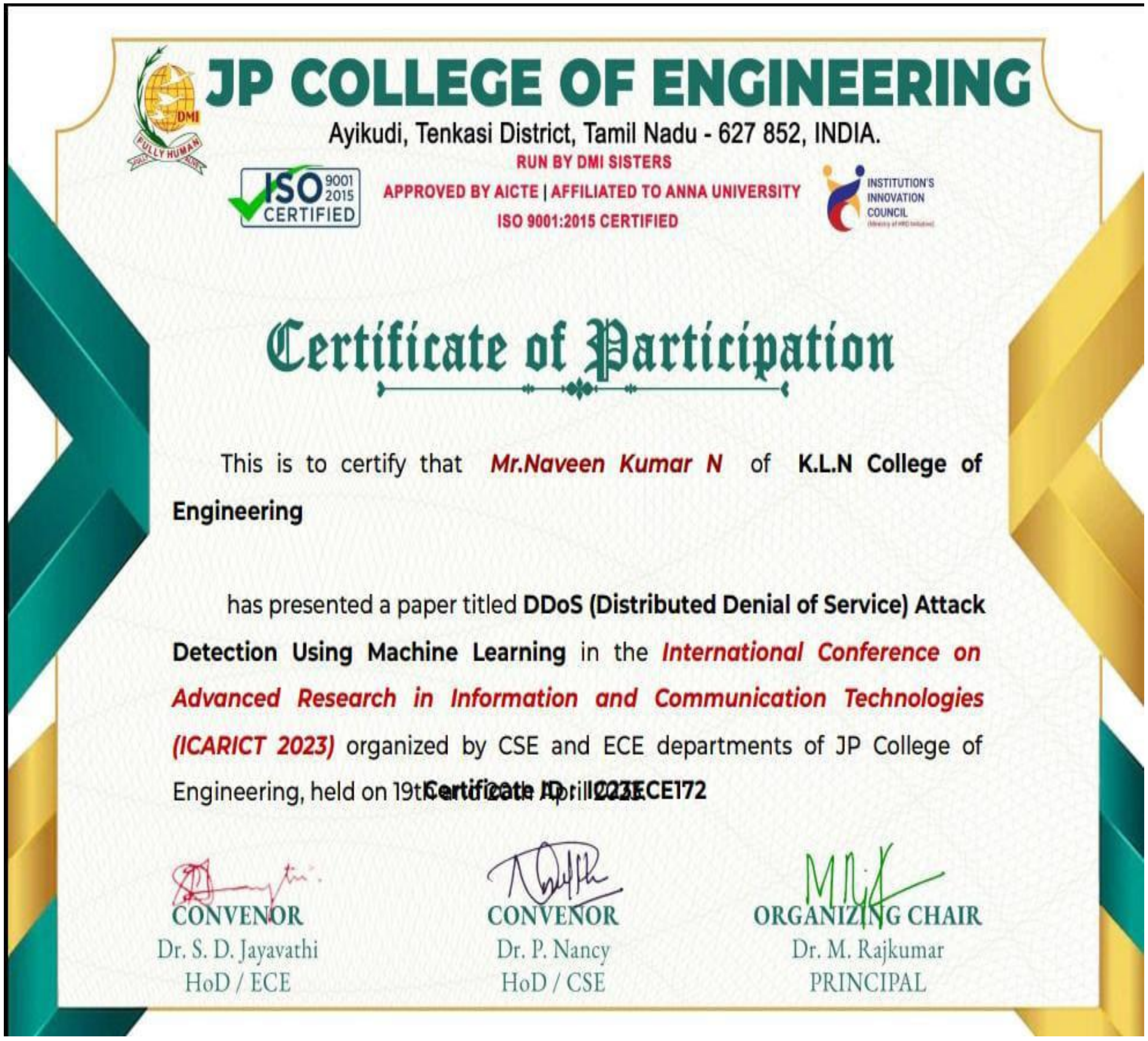
3."A Novel Method for Detecting DDoS Attacks Based on Random Forest

Algorithm" by Meng Liu, Shengli Liu, and Huaqi Wang, in IEEE Access, 2019.

4. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," in IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41-50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.

5. Hussain, Q. Du, B. Sun and Z. Han, "Deep Learning-Based DDoS-Attack Detection for Cyber–Physical System Over 5G Network," in IEEE Transactions on Industrial Informatics, vol. 17, no. 2, pp. 860-870, Feb. 2021, doi: 10.1109/TII.2020.2974520.

6." Iqbal, H., Ali, S., & Aslam, F. (2021). DDoS Attack Detection using CatBoost Algorithm with Feature Selection and Dimensionality Reduction. International Journal of Advanced Computer Science and Applications (IJACSA), 12(2), 18-23. doi: 10.14569/IJACSA.2021.0120203



JP COLLEGE OF ENGINEERING

Ayikudi, Tenkasi District, Tamil Nadu - 627 852, INDIA.

RUN BY DMI SISTERS

APPROVED BY AICTE | AFFILIATED TO ANNA UNIVERSITY

ISO 9001:2015 CERTIFIED

ISO 9001 2015 CERTIFIED

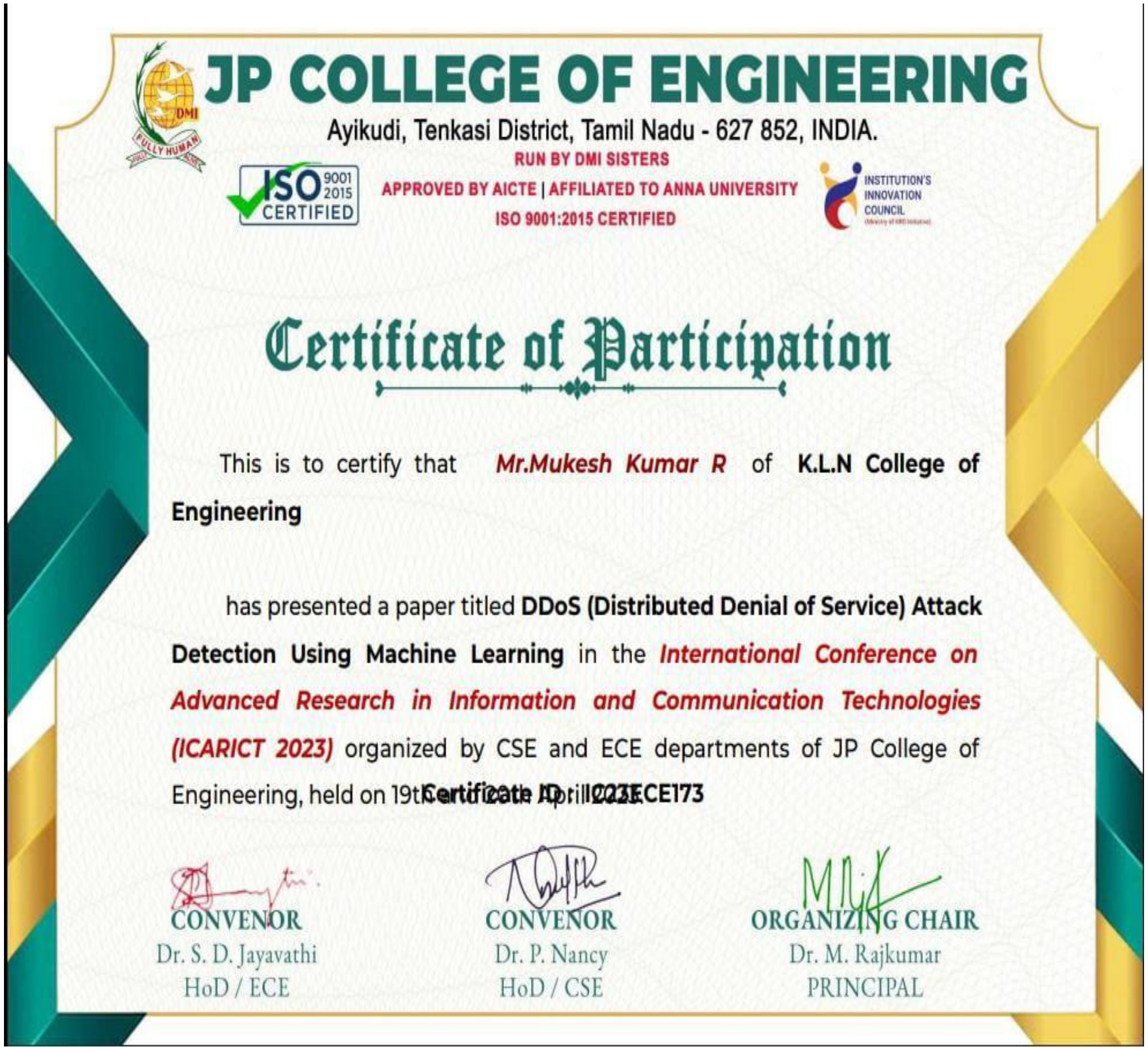
INSTITUTION'S INNOVATION COUNCIL

Certificate of Participation

This is to certify that *Mr.Naveen Kumar N* of **K.L.N College of Engineering**

has presented a paper titled **DDoS (Distributed Denial of Service) Attack Detection Using Machine Learning** in the ***International Conference on Advanced Research in Information and Communication Technologies (ICARICT 2023)*** organized by CSE and ECE departments of JP College of Engineering, held on 19th and 20th April 2023. **Certificate ID : IC23ECE172**

[illegible]	[illegible]	[illegible]
CONVENOR	CONVENOR	ORGANIZING CHAIR
Dr. S. D. Jayavathi	Dr. P. Nancy	Dr. M. Rajkumar
HoD / ECE	HoD / CSE	PRINCIPAL



JP COLLEGE OF ENGINEERING

Ayikudi, Tenkasi District, Tamil Nadu - 627 852, INDIA.

RUN BY DMI SISTERS

APPROVED BY AICTE | AFFILIATED TO ANNA UNIVERSITY

ISO 9001:2015 CERTIFIED

ISO 9001 2015 CERTIFIED

INSTITUTION'S INNOVATION COUNCIL

Certificate of Participation

This is to certify that *Mr.Mukesh Kumar R* of **K.L.N College of Engineering**

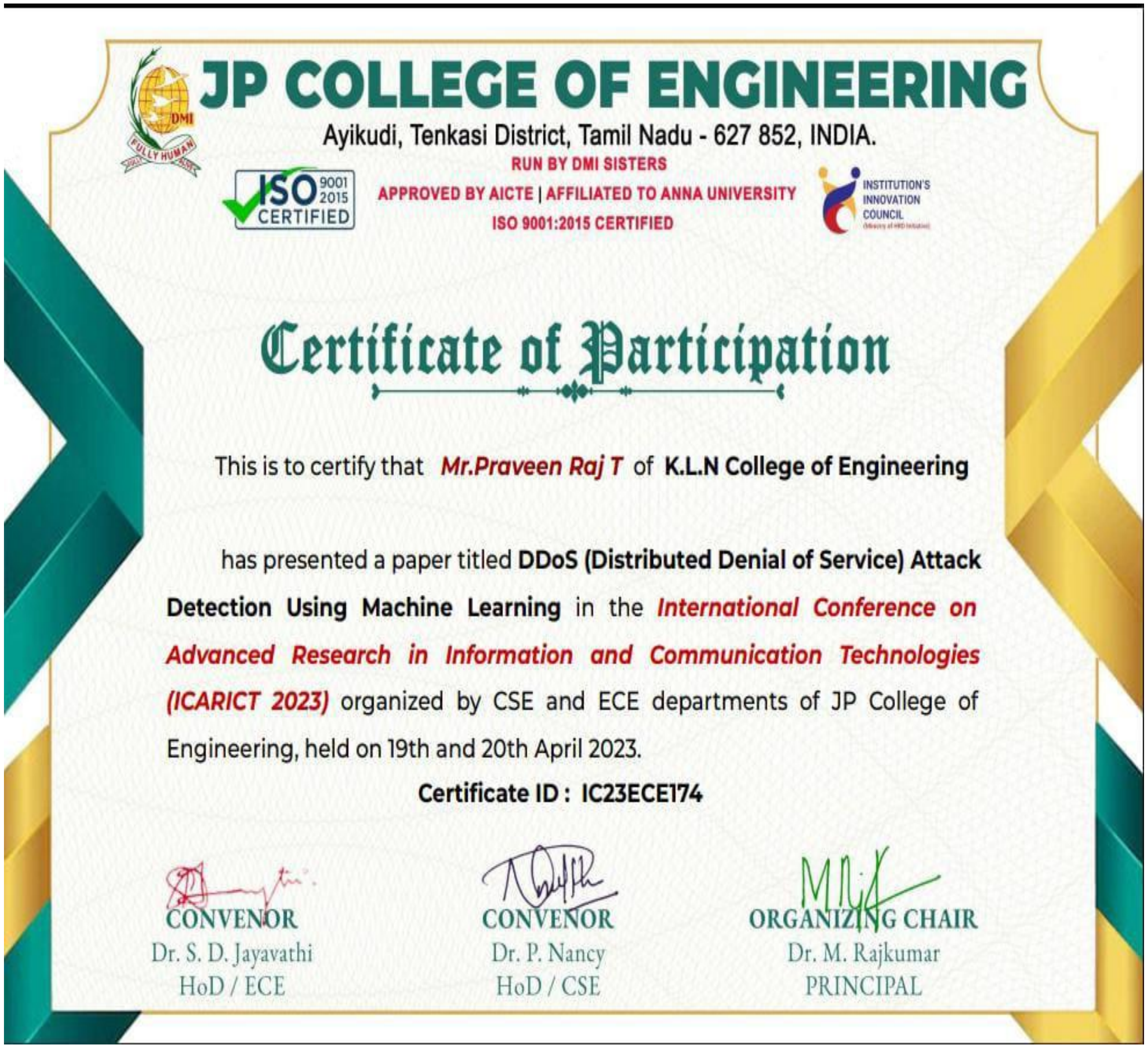
has presented a paper titled **DDoS (Distributed Denial of Service) Attack Detection Using Machine Learning** in the ***International Conference on Advanced Research in Information and Communication Technologies (ICARICT 2023)*** organized by CSE and ECE departments of JP College of Engineering, held on 19th and 20th April 2023.

Certificate ID : IC23ECE173

[illegible]
CONVENOR
Dr. S. D. Jayavathi
HoD / ECE

[illegible]
CONVENOR
Dr. P. Nancy
HoD / CSE

[illegible]
ORGANIZING CHAIR
Dr. M. Rajkumar
PRINCIPAL



JP COLLEGE OF ENGINEERING

Ayikudi, Tenkasi District, Tamil Nadu - 627 852, INDIA.

RUN BY DMI SISTERS

ISO 9001 2015 CERTIFIED

APPROVED BY AICTE | AFFILIATED TO ANNA UNIVERSITY

ISO 9001:2015 CERTIFIED

INSTITUTION'S INNOVATION COUNCIL

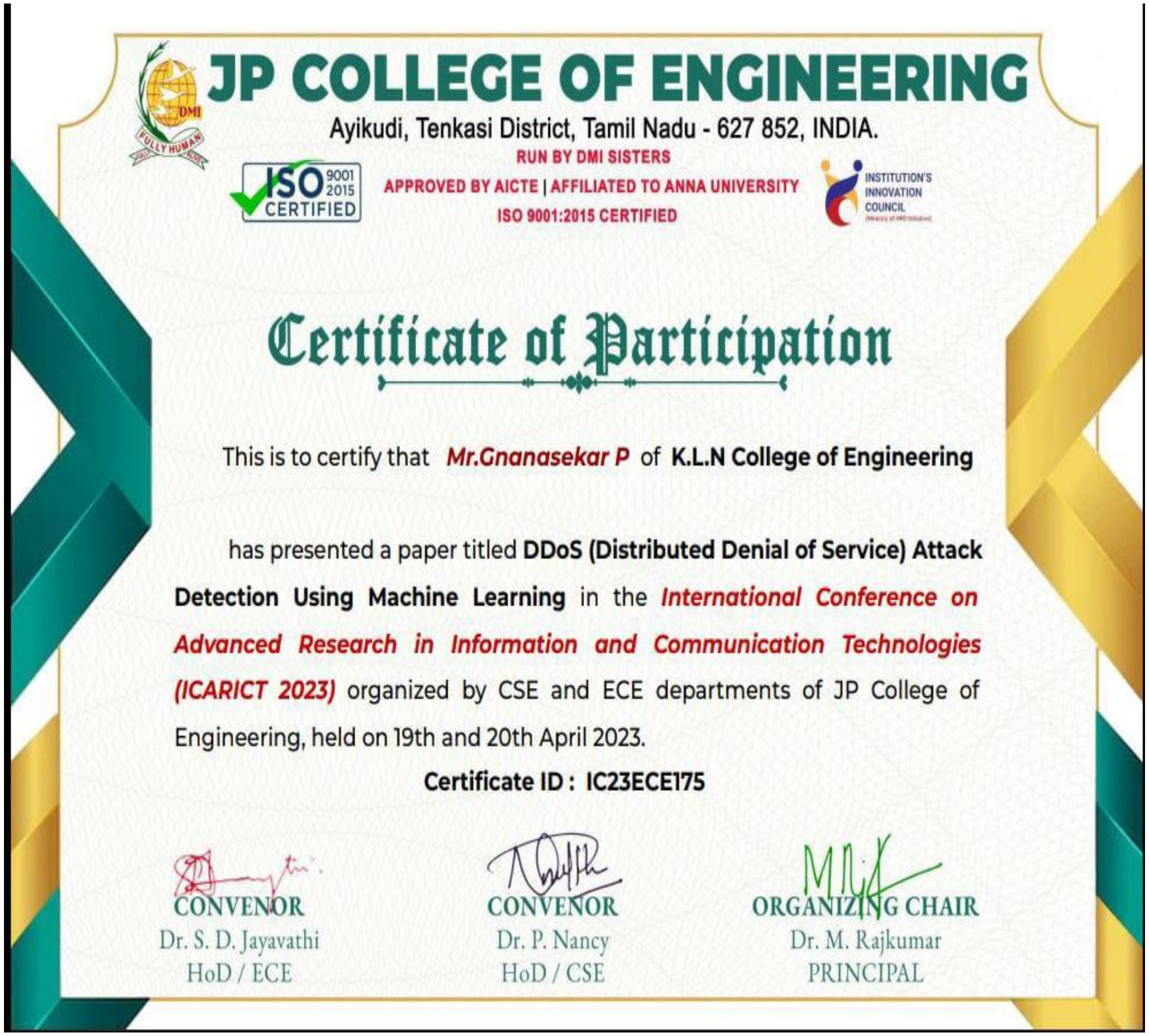
Certificate of Participation

This is to certify that *Mr.Praveen Raj T* of **K.L.N College of Engineering**

has presented a paper titled **DDoS (Distributed Denial of Service) Attack Detection Using Machine Learning** in the *International Conference on Advanced Research in Information and Communication Technologies (ICARICT 2023)* organized by CSE and ECE departments of JP College of Engineering, held on 19th and 20th April 2023.

Certificate ID : IC23ECE174

[illegible]	[illegible]	[illegible]
CONVENOR	CONVENOR	ORGANIZING CHAIR
Dr. S. D. Jayavathi	Dr. P. Nancy	Dr. M. Rajkumar
HoD / ECE	HoD / CSE	PRINCIPAL



JP COLLEGE OF ENGINEERING

Ayikudi, Tenkasi District, Tamil Nadu - 627 852, INDIA.

RUN BY DMI SISTERS

APPROVED BY AICTE | AFFILIATED TO ANNA UNIVERSITY

ISO 9001:2015 CERTIFIED

ISO 9001 2015 CERTIFIED

INSTITUTION'S INNOVATION COUNCIL

Certificate of Participation

This is to certify that ***Mr.Gnanasekar P*** of **K.L.N College of Engineering**

has presented a paper titled **DDoS (Distributed Denial of Service) Attack Detection Using Machine Learning** in the ***International Conference on Advanced Research in Information and Communication Technologies (ICARICT 2023)*** organized by CSE and ECE departments of JP College of Engineering, held on 19th and 20th April 2023.

Certificate ID : IC23ECE175

[illegible]	[illegible]	[illegible]
CONVENOR	**CONVENOR**	**ORGANIZING CHAIR**
Dr. S. D. Jayavathi	Dr. P. Nancy	Dr. M. Rajkumar
HoD / ECE	HoD / CSE	PRINCIPAL