5-21-2024

USING MACHINE LEARNING ALGORITHM FOR DETECTING DISTRIBUTED DENIAL OF SERVICE ATTACK

Zainab Alblooshi
zaa7513@rit.edu

USING MACHINE LEARNING ALGORITHM FOR DETECTING DISTRIBUTED DENIAL OF SERVICE ATTACK

by

Zainab Alblooshi

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree

of Master of Science in Professional Studies: Data Analytics

Department of Graduate Programs & Research

Rochester Institute of Technology
RIT Dubai

May 21, 2024

RIT

Master of Science in Professional Studies:
Data Analytics

Graduate Thesis Approval

Student Name**: Zainab Alblooshi**
Graduate Capstone Title**: USING MACHINE LEARNING ALGORITHM FOR DETECTING DISTRIBUTED DENIAL OF SERVICE ATTACK**

Graduate Thesis Committee:

Name:	**Dr. Sanjay Modak** **Chair of committee**	**Date:**
Name:	**Dr. Ehsan Warriach** **Mentor**	**Date:**

Acknowledgments

I would like to express my heartfelt thanks to my thesis mentor, Dr. Ehsan, for his constant assistance, perceptive input, and valuable direction throughout the process of writing and researching my thesis. His expertise and encouragement were vital in shaping this academic endeavor. This journey has been a significant and unforgettable period of my academic life, and I express gratitude to all those who contributed to its accomplishment.

Abstract

DDoS attacks, which stand for Distributed Denial of Service attacks, play a significant role in impacting the reliability and availability of online services and networks. Since no system is completely immune to cybersecurity threats, which evolve daily with new techniques, studying this topic is crucial for exploring machine learning methods that can effectively detect DDoS attacks. An approach has been utilized to conduct Exploratory Data Analysis (EDA) to identify patterns suggesting the presence of DDoS attacks. Multiple machine learning models have been employed, such as Random Forest, K-Nearest Neighbors (KNN), XGBoost, and Logistic Regression. These models have undergone training and testing to identify abnormal network activity linked to DDoS attacks. Performance analysis measures, such as accuracy, recall, F1-score, and precision, are used to assess the efficiency of each model. The ML-based solution has demonstrated excellent performance in detecting DDoS attacks, as evidenced by the accurately labeled network traffic examples that determine whether they are legitimate or malicious, resulting in a calculated accuracy from the test results. Moreover, among the models used, the Random Forest and XGBoost models show exceptional accuracy, recall, and F1-score measurements, with an accuracy rate over 99%. On the other hand, while KNN shows praiseworthy performance, Logistic Regression yields somewhat lower accuracy and recall ratings.

Key Words: Distributed Denial of Service (DDoS), Machine Learning, Cy- bersecurity, Random Forest, K-Nearest Neighbors (KNN), XGBoost, Logistic Regression.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Problem Statement

Imagine a future where information is freely shared, enabling us to do multiple things. The extraordinary reality of the internet era is just that! However, great power also comes with great responsibility, and the internet is no different. Similar to how a busy metropolis may draw both tourists and troublemakers, malevolent actors pose a hazard to worldwide online communities.

A distributed denial of service attack is one of the biggest hassles for everyone who uses the internet. It prevents real consumers from accessing the website, which results in website crashes and financial losses for businesses.

The truth is that professionals in cybersecurity are always creating new strategies to fight back. Historically, we have relied on devices like intrusion detection systems and firewalls to keep attackers out. However, these techniques might be tricked by extremely skilled attackers. This is an instance when machine learning is applicable.

The main goal of this research is to investigate whether or not machine learning may act as a last line of defense against DDoS attacks. The plan is to gather a significant amount of data on network traffic, including the number, size, and transit time of messages. Next, we will train unique device mastery models to become adept at identifying DDoS attempts. We will put all of these models to the test, including XGBoost and Random Forest. We'll identify the most suitable method for the task by putting them through their paces and evaluating their overall performance.

1.2 Background of the problem

A Distributed Denial of Service (DDoS) attack occurs when a malicious individual coordinate an attack by flooding the targeted system or network with an enormous amount of traffic, making it unreachable to authorized users. This attack exploits weaknesses in the network infrastructure. Attackers often use botnets, which are networks of compromised devices, to launch attacks on target systems by inundating them with excessive traffic. This overwhelms the systems' capacity to handle legitimate requests, resulting in service disruptions, financial losses, and reputational harm for enterprises. Due to the evolving characteristics of DDoS attacks, it is essential to develop detection technologies that are capable of handling increased demands.

1.3 Project Goals

Addressing cyber threats such as Distributed Denial of Service (DDoS) attacks is an essential focus. The objective of this research is to assist network administrators in addressing these threats. The development of advanced methods for identifying and reducing the impact of DDoS attacks will be achieved via thorough study and new approaches. These state-of-the-art solutions enable administrators to protect networks against malicious traffic overloads. The study investigates advanced techniques for rapidly identifying

- The project goals encompass several key aspects, including:

- A comparison of how well various ML models perform in detecting DDoS attacks.

- Knowledge of the advantages and disadvantages of any assessed machine learning model.

- Suggestions about the best machine learning models to use in DDoS attack detection systems.

- A deeper understanding of how machine learning strengthens cybersecurity defenses against attacks using DDoS.

1.4 Aims and Objectives

The project aims to enhance cybersecurity. It will provide network administrators with intelligent tools to detect and prevent Distributed Denial of Service (DDoS) attacks. The objective is to evaluate the efficacy of machine learning (ML) in detecting and mitigating distributed denial-of-service (DDoS) attacks. The primary objectives are:
The objective is to investigate the ability of machine learning models to detect abnormal network traffic. This might indicate the occurrence of a Distributed Denial of Service (DDoS) attack.

- We will conduct tests on these machine learning models. We will evaluate their proficiency in detecting attacks by using metrics such as precision, recall, F1-score, and accuracy.

- The findings will determine the most effective machine learning models for identifying attacks using DDoS.

- The work has the potential to enhance cyber defenses. It will demonstrate the effectiveness of machine learning in protecting networks from constantly changing cyber threats.

1.5 Research Methodology

Machine learning (ML) is an effective method for detecting and preventing Distributed Denial of Service (DDoS) attacks. This methodology uses specialized computer algorithms to analyze vast quantities of data related to network traffic. It searches for patterns that might indicate the occurrence of an attack. Machine learning has the ability to identify these patterns and respond rapidly, without requiring explicit instructions provided by humans.

Adaptability: Machine learning models have the ability to adjust and conform to changing attack patterns and acquire knowledge from fresh data, enhancing their effectiveness in identifying previously unknown threats.

Scalability: ML algorithms have the capacity to effectively handle substantial amounts of network traffic data, allowing for real-time analysis

Automation: Machine learning-based detection systems have the capability to automatically identify and mitigate DDoS attacks, therefore minimizing the need for user involvement.

Accuracy: Machine learning models may use advanced algorithms to identify tiny irregularities that are indicative of DDoS attacks, leading to improved detection accuracy and a reduction in false positives.

1.6 Limitations of the Study

Data quality: Machine learning algorithms need substantial quantities of well annotated, high-caliber data. Obtaining a comprehensive dataset that encompasses many types of attacks using DDoS may be challenging and may have an impact. The efficacy and precision of the model are strongly linked to the caliber and inclusiveness of the training data.

Chapter 2 : Literature Review

2.1 Literature Review

Nalayini & Katiravan (2022) aimed to determine the various parameters that should be considered for comparing and evaluating DDoS attack detection and prediction. These parameters include accuracy, precision, recall, and the false positive alarm rate, as each of these plays a crucial role in assessing the effectiveness of detection. Accuracy provides a comprehensive view of how well the detection system is performing and how reliable it is in distinguishing between normal and attack traffic. Precision helps reduce false alarms by ensuring that when an alert appears, it indicates an actual attack rather than a false positive. Recall indicates how effectively a system performs in detecting attacks, even when they are hidden or partially obscured. The false positive rate is important for avoiding unnecessary disruptions to normal operations. All these mentioned parameters help understand the advantages and constraints of a detection system, enabling informed decision-making about where and how to use it.

Priyadarshini & Devi (2020) asserted that detecting DDoS attacks is possible using the NAIDA method, an older model that identifies abnormal network traffic in a system using a network sniffer. However, the recommended model involves utilizing a dataset trained on Support Vector Machine (SVM) through the Weka JAR. SVM works based on the types of attacks, namely volume-based attacks, protocol attacks, and application layer attacks. Implementing SVM for detecting attacks enhances the identification of DDoS attacks, categorizing their nature, implementing preventive measures to block them, and generating reports on the identified types of detected worms.

Suresh & Anitha (2011) observed in their research that current detection mechanisms experience limited success. This is attributed to two challenges. The first challenge is that attacks typically employ legitimate requests to flood the target, complicating the distinction between normal and attack traffic. The second challenge is the difficulty of swift real-time detection due to the substantial data flow in computer networks. In their research paper, they also mentioned utilizing chi-square and information gain feature selection mechanisms to identify essential attributes. Once they selected the attributes, various machine learning models such as Naive Bayes, C4.5, SVM, KNN, K-means, and Fuzzy c-means clustering were enhanced for DDOS attack detection. Through their experimentation, they found that Fuzzy c-means clustering provided superior accuracy for attack identification.

Johnson & George (2022) emphasize that the utilization of traffic analytics tools enables the detection of various warning signals indicating a potential DDoS attack. Examples of these warning signals include an excessive amount of traffic originating from a single IP address or a group of IPs, a high volume of traffic from users exhibiting similar behavior (such as using the same device or browser), a sudden surge in requests directed at a specific page or endpoint, and peculiar traffic patterns, such as abrupt spikes during unconventional times or patterns that appear fake (e.g., a spike occurring every 10 minutes). It's important to note that specific indicators of a DDoS attack may vary depending on the nature of the attack.

Robinson & Thomas (2015) conducted a study in which they aimed to evaluate and rank the performance of machine learning algorithms for detecting DDoS attacks. To achieve this, they utilized three distinct datasets and employed various algorithms, including Naïve Bayes, RBF network, Multi-layer Perceptron, etc. Their focus was specifically on key attributes such as False Negative (FN) rate, False Positive (FP) rate, precision, and recall for each algorithm. This emphasis aimed at reducing error types and enhancing precision and recall. Furthermore, the researchers applied a criteria-driven approach using Visual PROMETHEE, which stands for Preference Ranking Organization Methods for Enrichment Evaluation, along with MCDE software.

Gu et al. (2019) discovered that a semi-supervised k-means machine learning algorithm is effective for classifying DDoS attacks. They applied a hybrid feature selection algorithm to enhance detection results, implementing this approach across various datasets, including DARPA, CAIDA, and CICIDS datasets. Additionally, he successfully implemented a real-world scenario dataset by employing a tool to simulate both attacker and normal traffic.

Filho et al. (2019) stated that there is a smart detection for DDOS attacks , as it is main purpose for detecting DDOS attacks that is with both high volume and low volume, it worked as a sensor that it can be installed anywhere on network and it figures out internet traffic by using an MLA strategy . This strategy looks at random bits of data taken from devices on the network through a stream protocol to make predictions.

Pei et al. (2019) reported that a common DDoS attack tool can be employed for implementing local attacks. In this approach, the packet capture tool is used to compare captured attack packets with normal data packets. The authors proposed a new machine learning method, the random forest algorithm model. This involves extracting attack packets for three protocols: TCP flood, UDP flood, and ICMP flood, using the DDoS attack tool. Two essential processes are carried out to extract DDoS attack traffic characteristics: feature extraction and format conversion. Subsequently, the extracted features are utilized as input for machine learning, specifically to train and create the DDoS attack detection model using the random forest algorithm. For testing the model, normal traffic data is mixed with attack traffic data.

Devi et al. (2014) proposed a method to counteract DDOS attacks by employing a prevention system designed to block any harmful traffic that deviates from the service's normal behavior or matches a known attack signature in the database. Additionally, mitigation systems are utilized to push back against attacks, restoring the server's normal functioning by rejecting unintended requests, connections, or reducing data flow from suspicious sources. Furthermore, they introduced a model known as HCF-SVM, which was evaluated alongside Random Forest and Decision Tree models. HCF-SVM demonstrated a 98.99% detection rate, with the added benefit of reducing false positives.

Chen et al. (2018) posited the existence of a new type of cloud known as SDN-based cloud, employed to gain control over network infrastructure and provide Networking as-a-Service (NaaS). They highlighted that notable companies, such as Google, have incorporated SDN into their cloud infrastructure. However, the integration of SDN and Cloud has brought about potential cybersecurity vulnerabilities. Consequently, there is a crucial need to develop an accurate detection method for Distributed Denial of Service (DDoS) attacks in SDN controllers. Furthermore, within the SDN architecture, a pivotal component is the controller, acting as the brain of SDN. They emphasized that if the controller is compromised, the entire network becomes susceptible to attacks. They proposed the utilization of the XGBoost algorithm as an accurate method for detection due to its higher accuracy and lower false positive rates compared to other machine learning algorithms implemented in their research, such as Support Vector Machines (SVM), Gradient Boosted Decision Trees (GBDT), and Random Forest.

Qin et al. (2015) highlighted the efficacy of clustering techniques for enhancing the modeling of different aspects of traffic. They found that employing clustering methods provides valuable insights into the complex and varied nature of traffic patterns. In particular, for their modeling strategy, they chose to incorporate clustering along with an automatic thresholding mechanism.

Bandara et al. (2016) reported that intrusion prevention systems (IPS) and intrusion detection systems (IDS) typically struggle to identify new DDoS attack techniques, as these attacks become increasingly sophisticated each year. To address this challenge, the utilization of machine learning algorithms and pattern recognition is employed to enable systems like IPS and IDS to analyze emerging types of DDoS attacks and implement preemptive measures for mitigation without necessitating user intervention.

Sanjeetha et al. (2021) stated that the utilization of the CatBoost machine learning algorithm results in significantly faster predictions, approximately 8 times quicker than XGBoost. The reason behind this is its proven reliability and effectiveness in detecting DDoS attacks with a 98% accuracy rate and a considerably shorter training duration.

Sofi et al. (2017) underscored the effectiveness of MLP-ANN, also known as Multi-Layer Perceptron Artificial Neural Networks, in detecting DDoS attacks. MLP-ANN operates by learning patterns and relationships in data, proving efficient in tasks like pattern recognition, feature extraction, and real-time monitoring to identify and mitigate DDoS attacks. It is considered a machine learning algorithm that can be combined with other algorithms.

Kadam & Sekhar (2021) aimed to determine a machine learning approach, specifically a hybrid KSVM scheme based on both KNN and SVM algorithms. This approach was built as a secure framework for detecting DDoS attacks. The study demonstrates its effectiveness in enhancing performance and adapting scenarios in VANETs, referring to vehicular networks. VANETs have garnered significant attention from both industry and academia. However, they encounter various challenges, including security, traffic congestion, which have not been adequately addressed in recent years.

Sambangi & Gondi (2020) stated that the issue of the DDoS attack might be considered as a classification problem in machine learning. Specifically, when it comes to cloud computing, the main challenge that security analysts might consider is whether the attack is present or not, due to the computational complexity.

Mihoub et al. (2017) underscored that nowadays, various systems—specifically IoT systems, which stand for Internet of Things systems—are considered one of the most frequent victims of DDoS attacks. In their study, they focused on a main component called a multi-class classifier that adopts the "Looking-Back" approach. "Looking-Back" machine learning techniques are effective in several aspects such as enhancing detection accuracy, providing real-time responses, and enabling adaptive learning.

Khuphiran et al. (2018) highlighted that the use of DFF, a new learning algorithm that stands for Deep Feed Forward, achieves higher accuracy compared to the traditional SVM. With DFF, the accuracy reaches approximately 99.63%, whereas with SVM, it achieves 99.01%. This indicates that DFF is effective when accuracy is a major concern for security analysts.

Sumathi et al. (2022) reported that there is a model called the Intrusion Detection System (IDS), which might be effectively utilized to detect DDoS attacks depending on different machine learning algorithms such as C4.5, SVM, and KNN classifiers, along with a 10-fold cross-validation technique. Additionally, in their study, they used a 10-fold cross-validation technique to select trend features and avoid obtaining biased outputs.

Tayyab et al. (2020) noted that the primary difficulties encountered by ML-based IDS models in identifying ICMPv6-based attacks. Among these difficulties is a real-time performance bottleneck that affects real-time accuracy since it is hard to train models in offline situations and large-scale networks also have scaling challenges, where a cooperative approach based on ensemble learning is required to minimize false positives due to an increase in alarms.

Zhai et al. (2018) highlighted that they utilized a framework called PCA-RNN, which stands for Principal Component Analysis-Recurrent Neural Network, to detect DDoS attacks. This indicates that when comparing PCA-RNN to other DDoS attack detection techniques currently in use, a significant increase in accuracy, sensitivity, precision, and F-score can be achieved. Additionally, another algorithm used is the PCA algorithm, which serves to reduce the dimensions of features, thereby decreasing the complexity of detection time. The PCA algorithm is considered a preprocessing step to enhance the performance of the machine learning model.

Hou et al. (2018) emphasize that the combination of utilizing machine learning and adaptive feature extraction from NetFlow data is effective for detecting DDoS attacks. For example, when used with real-world NetFlow records obtained from a major Internet Service Provider (ISP), the system accurately assessed DDoS attacks, providing valuable information about the frequency, magnitude, and specific services targeted by the attacks. This underscores the importance of using NetFlow analysis, as it helps to extract flow-based features such as the number of packets, bytes, and duration of flows, and also helps to extract pattern-based features that capture the behavior of traffic.

Wang et al. (2020) asserted that Distributed Denial of Service (DDoS) attacks continue to be a major obstacle in the field of network security. While there have been advancements in machine learning-based methods for detection, the crucial task of choosing the most efficient features still remains. Wang observes that current techniques often depend on manually selected characteristics, which may not adequately represent the dynamic nature of network traffic. Wang proposes a resolution to this problem by integrating multilayer perceptrons (MLP) with sequential feature selection. This methodology employs a dynamic method to identify the most significant characteristics throughout the training phase and integrates a feedback system to adjust the detector based on detection mistakes.

Lee et al. (2008) discovered that proactive techniques for detecting DDoS attacks emphasize the need to comprehend the attack's lifecycle. It identifies key characteristics that indicate an incoming attack by evaluating the choices of handlers and agents, communication patterns, and attack patterns. By using cluster analysis, this technique efficiently divides each stage of the attack scenario, allowing for the identification of early indicators prior to the occurrence of the attack.

Ye et al. (2018) investigates the detection of Distributed Denial of Service (DDoS) attacks specifically in the framework of software-defined networking (SDN). Although prior studies have investigated the use of deep learning algorithms to simulate attack behavior, the actual implementation of these methods is still difficult. Ye develops a robust DDoS attack model by combining SVM classification methods and extracting 6-tuple characteristic variables from switch flow tables. The experimental results provide encouraging average accuracy rates, highlighting the potential significance of this method for identifying attacks using DDoS in SDN systems.

Zekri et al. (2017) reported on the creation and validation of a machine learning-based DDoS detection system. The system employs the C4.5 decision tree method and signature detection to identify attacks using DDoS in cloud computing environments both precisely and effectively. The C4.5 algorithm was evaluated against other machine learning methods and demonstrated superior accuracy and detection times.

Wankhede & Kshirsagar (2022) aimed to determine if another machine learning algorithm, the Multi-Layer Perceptron (MLP), could be utilized. In their study, they applied two different algorithms: Random Forest and Multi-Layer Perceptron. It was found that Random Forest is more effective than MLP for detection, achieving higher accuracy.

Saghezchi et al. (2022) observed in their research that there are various challenges in detecting DDoS attacks in industrial CPPSs, which stands for Cyber-Physical Production Systems, when utilizing machine learning techniques. These challenges include: the network traffic data in industrial contexts exhibits a high level of variety and complexity. The research highlighted the challenge of accurately representing such data because of the variety of traffic patterns, which may impact the effectiveness of machine learning models.

Patel et al. (2024) reported the integration of machine learning techniques into advanced firewalls to improve the identification of attacks using DDoS. The researchers discussed the use of several machine learning methods, such as binary decision trees, XGBoost, and the Support Vector Machines (SVM). These algorithms have a significant impact on the functionality of next-generation firewalls. They aid in the identification of traffic patterns by examining traffic volumes, frequency, distribution, and packet headers. In addition, they enhance real-time processing and decision-making by evaluating traffic data in real-time, allowing for rapid decisions on traffic characteristics.

Santos et al. (2019) observed in their research that detecting DDoS attacks in an SDN environment might be difficult due to the three different types of attacks: controller attacks, flow-table attacks, and bandwidth attacks. Therefore, it is important to understand the pattern to detect in each attack.

2.2 Key Takeaways

- Multiple machine learning algorithms have been explored for DDoS attack detection, including Support Vector Machines (SVM), Naive Bayes, Decision Trees, K-Nearest Neighbors (KNN), Fuzzy c-means clustering, Random Forest, and deep learning methods like Multi-Layer Perceptrons (MLP) and Deep Feed Forward networks. Each has its strengths and weaknesses in terms of accuracy, speed, and handling different types of attack patterns.

- Efficient feature selection is vital for improving model performance. Methods such as chi-square, information gain, and hybrid algorithms have been used to determine the most significant characteristics for identifying DDoS assaults. Effective feature selection reduces complexity and enhances the speed and accuracy of detection.

- Continuous updates and learning from fresh data are necessary to react to the dynamic nature of DDoS assaults. Several researches have investigated the use of semi-supervised learning and adaptive algorithms to effectively respond to constantly changing assault techniques.

- Obstacles like as the need for substantial computing resources, the necessity for a large amount of training data, and the potential vulnerability to adversarial attacks pose major challenges. In order to enhance the practicality of using these technologies, it is essential to tackle these difficulties.

Chapter 3: Project Description

Multiple stages may be used to provide a detailed project description for this project focused on establishing a machine learning-based system for detecting and mitigating DDoS attacks. It offers a comprehensive overview of the many phases of the project, starting with the gathering of data to the implementation of the model. It provides a well-defined plan of the activities that need to be completed, covering fundamental elements such as data preparation, exploratory data analysis, feature engineering, model training, evaluation, hyperparameter tuning, and model selection, all of which are vital for the effective execution of this project.

3.1 Data Collection

Employ a Kaggle resource to acquire data that includes diverse properties like packet counts, byte counts, duration, flow characteristics, and protocol information. The dataset should additionally include a target variable that indicates whether the traffic is classed as Attacks or Normal.

3.2 Data Preprocessing

Conduct data cleaning and preprocessing to remove any irrelevant or unwanted data, address any missing values, convert the data types of the features, and manage any outliers in the data.

3.3 EDA (Exploratory Data Analysis)

Conduct a thorough analysis of the dataset using both univariate and bivariate analytical methods to uncover underlying patterns and features that indicate DDoS assaults.

3.4. Feature Engineering

Identify and extract relevant features from the dataset to accurately represent the unique characteristics of both normal and malicious network traffic.

3.5 Model Training:

Employ the preprocessed dataset to train machine learning models such Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), XG-Boost, and Logistic Regression.

3.6 Model Evaluation

Evaluate the performance of each model by quantifying criteria like as accuracy, precision, recall, and F1-score.

3.7 Hyperparameter Tuning:

Enhance the performance of the models by adjusting the hyperparameters using techniques like grid search cross-validation.

3.8 Model Selection

Select the most efficient model according to assessment criteria and deploy it to detect and mitigate DDoS assaults.

Chapter 4 : Analysis

4.1 Detailed description

Sections on data preparation, exploratory data analysis, feature engineering, model training, assessment, hyperparameter tuning, and model selection will cover each project stage.

4.2 Data Loading and Exploration

The dataset entitled 'dataset sdn.csv' was imported into the Python environment using the Pandas package during this step. An examination was conducted to get a deeper understanding of the structure and characteristics of the dataset, which would aid in doing additional analysis. The dataset was imported into a Pandas DataFrame called 'data' using the pd.read_csv() method.

This method processes the CSV file and generates a DataFrame that encapsulates the dataset. Upon importing the dataset, a first examination was performed to get insight into its structure and contents. The following actions were executed:

4.3 Displaying the first few rows

The initial rows of the dataset were presented using the head() function, which offered an overview of its structure and the contents of every column.

4.4 Checking the dataset dimensions

The DataFrame shape property determined the dataset's rows and columns, indicating its size. The original dataset shape was (104345,23), where 23 represents the number of columns or and 104345 represents the records.

4.5 Inspecting column information

For each dataset column, the info() method displayed its name, data type, and quantity of non-null entries.

4.6 Distribution in Target Class

The 'label' column was subjected to the unique() method to identify the distinct classes found in the dataset. The 'label' column in this example consists of two distinct classes: 0 (Benign) and 1 (Malicious). The counts() method was employed to determine the frequency of every class in the 'label' column, offering an understanding of the distribution of classes.

4.7 Summary statistics

For numerical columns in the dataset, the describe method generated summary statistics like mean, median, minimum, maximum, and quartiles. The statistical overview of numerical columns is shown in Figure 4.1.

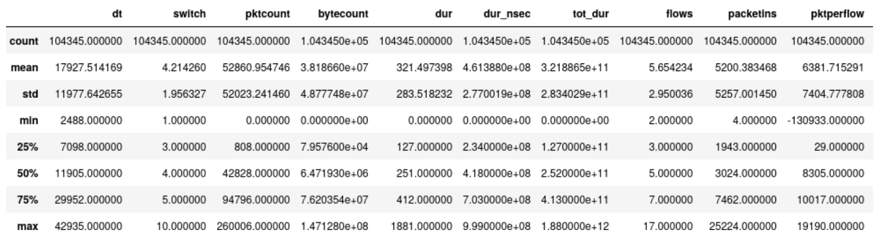
4.8 Identification of Numeric and Object Columns

In order to get started with the analysis, the dataset was partitioned into two distinct segments based on the data types: numeric and object. The `select_dtypes()` method was utilized to specifically identify columns with certain data types. The columns that included numbers were designated for numerical analysis, while the columns that contained objects were designated for either categorical or textual analysis. The outcomes of this phase provided significant insight into the structure, content, and attributes of the dataset.

4.9 Exploratory Data Analysis (EDA) through Univari- ate Analysis

The dataset entitled 'dataset sdn.csv' was imported into the Python environment using the Pandas package during this step. An examination was conducted to get a deeper understanding of the structure and characteristics of the dataset, which would aid in doing additional analysis. The dataset was imported into a Pandas DataFrame called 'data' using the pd.read_csv() method.

This method processes the CSV file and generates a DataFrame that encapsulates the dataset. Upon importing the dataset, a first examination was performed to get insight into its structure and contents. The following actions were executed:



	dt	switch	pktcount	bytecount	dur	dur_nsec	tot_dur	flows	packetins	pktperflow
count	104345.000000	104345.000000	104345.000000	1.043450e+05	104345.000000	1.043450e+05	1.043450e+05	104345.000000	104345.000000	104345.000000
mean	17927.514169	4.214260	52860.954746	3.818660e+07	321.497398	4.613880e+08	3.218865e+11	5.654234	5200.383468	6381.715291
std	11977.642655	1.956327	52023.241460	4.877748e+07	283.518232	2.770019e+08	2.834029e+11	2.950036	5257.001450	7404.777808
min	2488.000000	1.000000	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	2.000000	4.000000	-130933.000000
25%	7098.000000	3.000000	808.000000	7.957600e+04	127.000000	2.340000e+08	1.270000e+11	3.000000	1943.000000	29.000000
50%	11905.000000	4.000000	42828.000000	6.471930e+06	251.000000	4.180000e+08	2.520000e+11	5.000000	3024.000000	8305.000000
75%	29952.000000	5.000000	94796.000000	7.620354e+07	412.000000	7.030000e+08	4.130000e+11	7.000000	7462.000000	10017.000000
max	42935.000000	10.000000	260006.000000	1.471280e+08	1881.000000	9.990000e+08	1.880000e+12	17.000000	25224.000000	19190.000000

Figure 4. 1: Summary Statistics of Data

Univariate Analysis

Univariate analysis entails scrutinizing individual variables within the dataset to comprehend multiple aspects, such as their characteristics, outliers, and distribution.

1- Histogram for Numerical Columns

A visual depiction of the dataset's numerical variable distribution is provided by histograms. To comprehend the frequency distribution of every numerical column and spot trends or abnormalities, histograms were created for each of them in this study

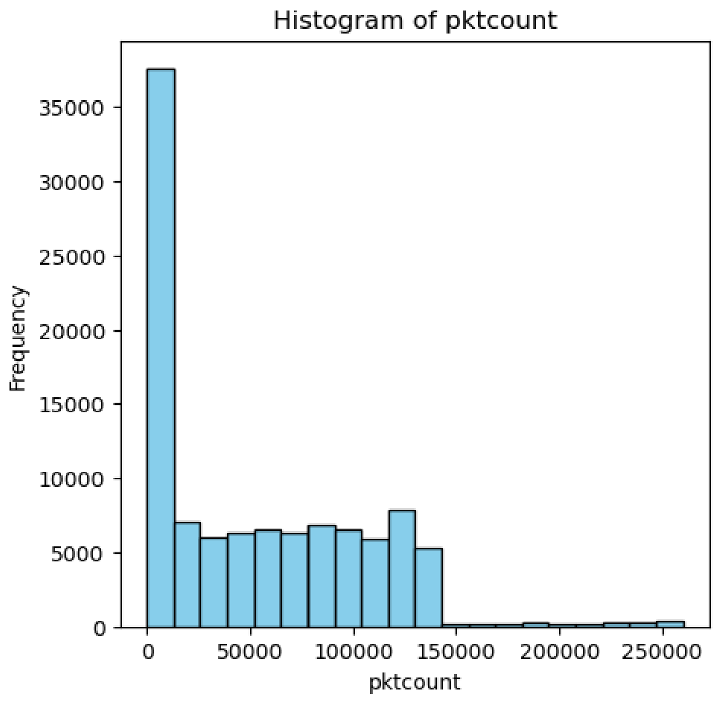


Figure 4. 2: Histogram of packet count

- The histogram shown in Figure 4.2 demonstrates that most observations in 'pktcount' have a comparatively small number of packets, with a prominent peak appearing at the lower end of the value range.

- In certain cases, there are fewer packets; however, these packets exhibit much larger counts. This indicates the presence of likely outliers or unexpected trends.

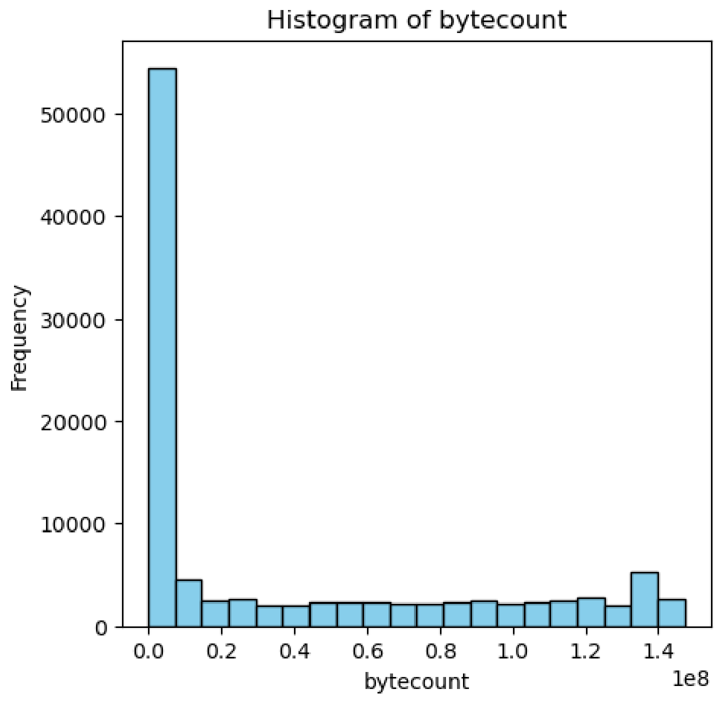


Figure 4. 3: Histogram of Bytes count

- A right-skewed distribution is shown by the histogram of "bytecount" in figure 4.3, where the bulk of observations have smaller byte counts.

- A long tail towards higher byte counts might be an indication of outliers or anomalies in the data, since it implies the existence of a small number of cases with noticeably higher byte counts.

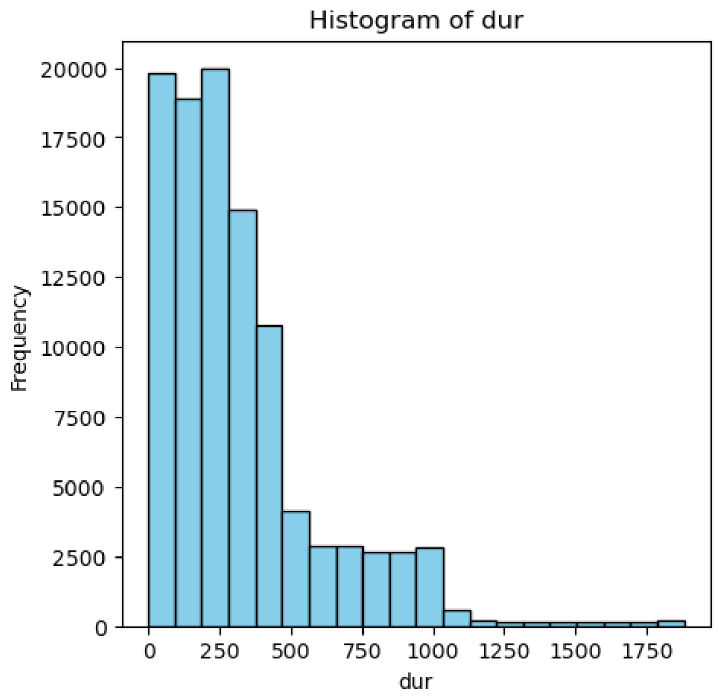


Figure 4. 4: Histogram of duration

- The histogram in Figure 4.4 displays the distribution of the variable 'dur' (duration). It shows a fairly even distribution, with data spread throughout a broad range of duration values.

- There is no obvious peak or skewness in the distribution, suggesting an even distribution of duration values throughout the sample.

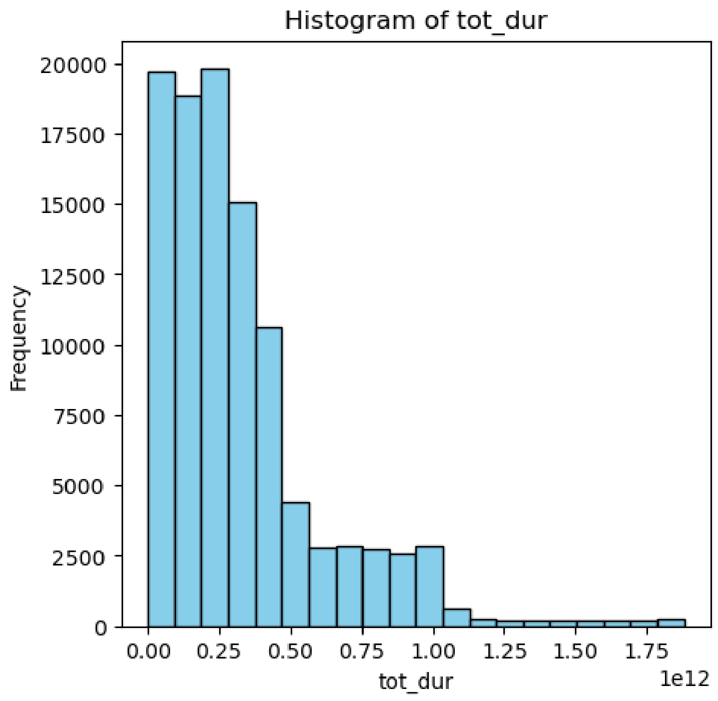


Figure 4. 5: Histogram of Total duration

- The histogram of 'tot dur' in figure 4.5, which represents the total duration, shows a distribution that is comparable to 'dur'. The observations are uniformly spread over the range of total duration values.

- Similar to the term 'dur', the distribution of total duration values does not exhibit a clear peak or skewness, indicating a balanced distribution.

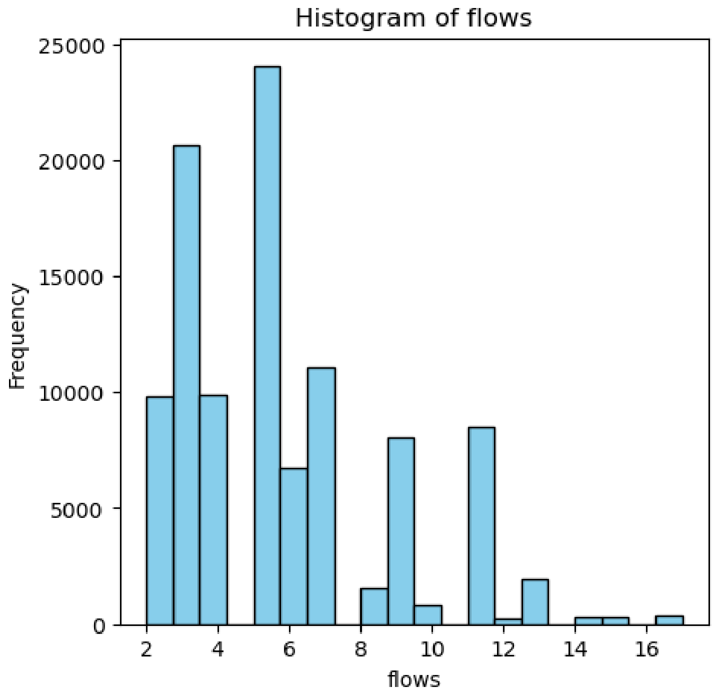


Figure 4. 6: Histogram of flows

- The histogram in the following figure 4.6 displays a distribution that is tilted to the right, with most observations having lower flow counts.

- Some occurrences had greater flow counts, indicating possible deviations or abnormalities in the data distribution.

Analysis:

The distribution and characteristics of the numerical variables in the dataset are very well-represented by histograms. It is crucial to comprehend the distribution of variables like "pktcount," "bytecount," "dur," and "flows" in order to spot patterns, outliers, or anomalies that might affect further modeling.

2- Bar Plot for Categorical Columns

For the purpose of visualizing the frequency distribution of categorical variables, bar plots are deployed. For the purpose of this research, bar plots were created for every category column in order to get an understanding of the distribution of categories and to determine which categories were dominant or common within every column.

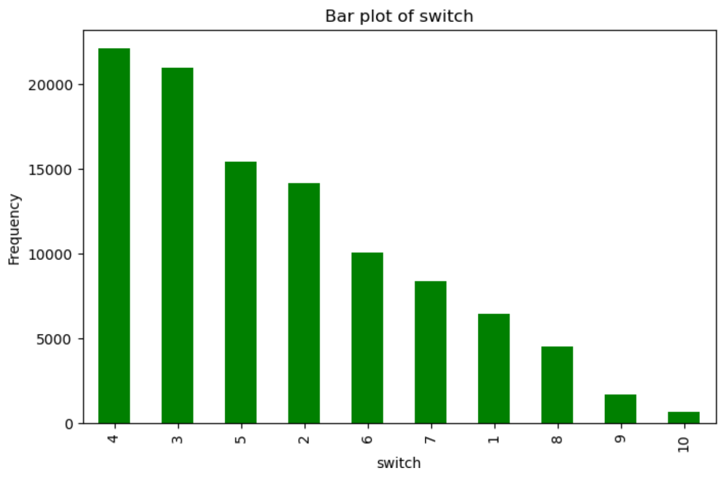


Figure 4. 7: Bar plot of Switch

- Figure 4.7 shows a balanced distribution of'switch' across categories (4 and 3), indicating a similar representation of switch status in the dataset. The dataset exhibits a minimal occurrence of switch categories 9 and 10. The remaining switches exhibit a moderate level of data range.

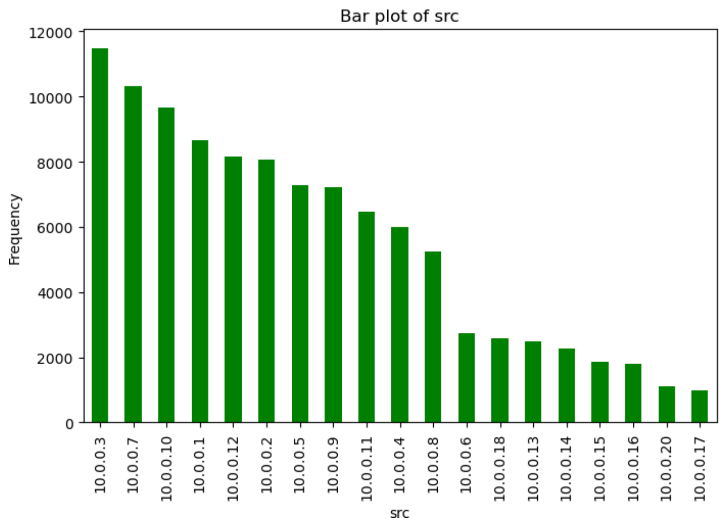


Figure 4. 8: Bar plot of Source IP

- The bar plot of the variable 'src' displays the distribution of source addresses in the dataset.

- There are several source addresses in the dataset, with just one IP address accounting for a large number of observations

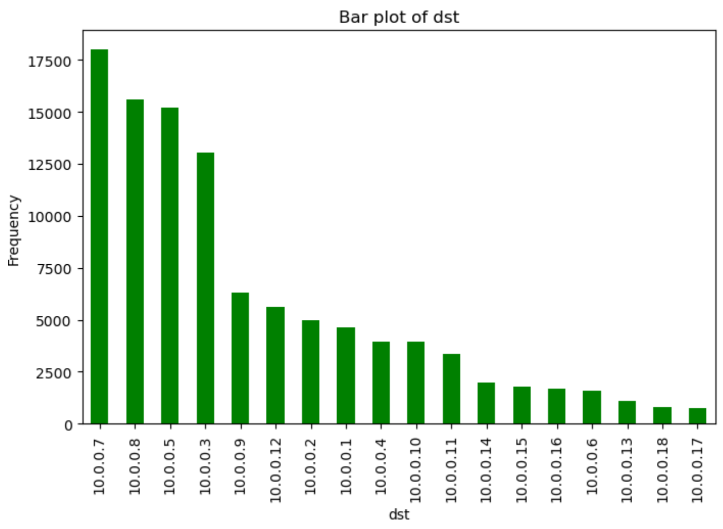


Figure 4. 9: Bar plot of Destination IP

- In Figure 4.9, the bar plot of 'dst' displays the frequency distribution of destination addresses in the dataset.
- The dataset displays a wide variety of destination dresses, much as the'src' column, with only the address shown in the plot dominating the distribution.

-

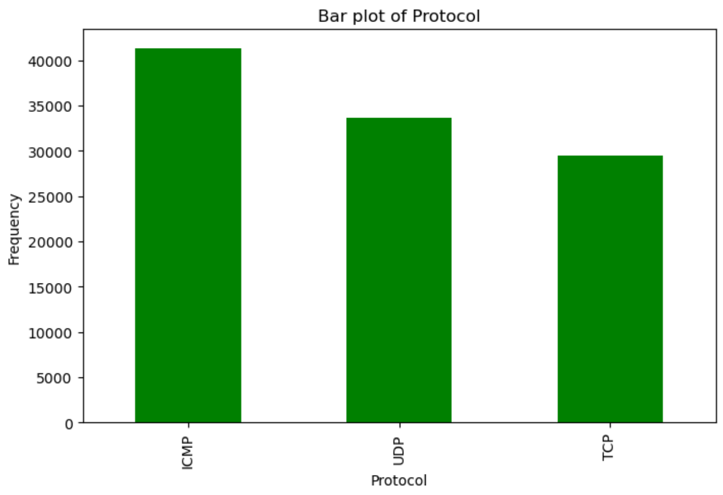


Figure 4. 10: Bar plot of Protocol

- A better understanding of the distribution of network protocols within the dataset may be gained from the bar plot of 'Protocol' that is shown in figure 4.10.

- The Transmission Control Protocol (TCP) is the protocol that is used the least, then the User Datagram Protocol (UDP) and the Internet Control Message Protocol (ICMP), which shows that these protocols are the most prevalent in network communication that is captured by the dataset.

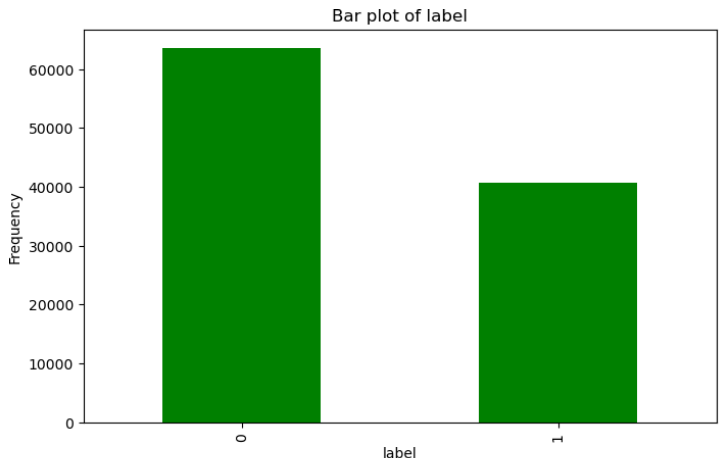


Figure 4. 11: Bar plot of Label

- The distribution of labels (0 and 1) that reflect benign and malicious occurrences, respectively, is shown in figure 4.11

- Given that the number of benign examples (label 0) is greater than the number of malicious instances (label 1), it is evident that the target class in the dataset is distributed in an imbalanced manner. When it comes to the performance of classification models that have been trained on this dataset, this will have a significant influence.

3- Box Plot for Identifying Outliers

Visualizing numerical data using box plots helps spot outliers. Box plots were created for every numerical column to detect outliers and extreme values.

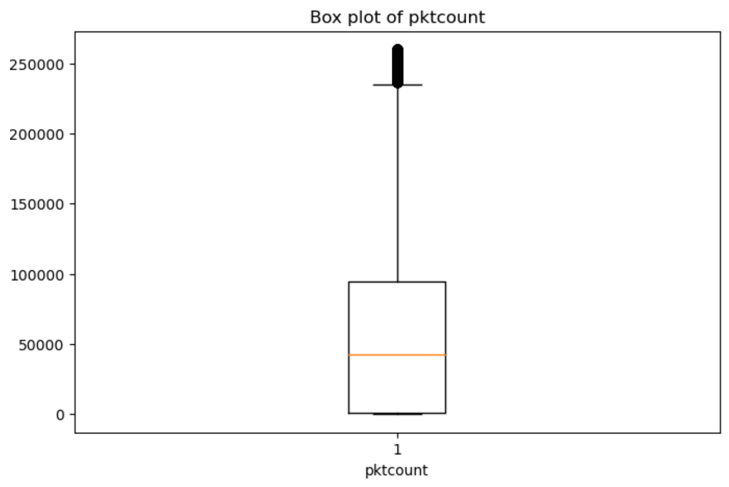


Figure 4. 12: Box plot of Packet Count

- The distribution of packet counts throughout the dataset is shown in figure 3.12 by the box plot of the 'pktcount' variable.

- The box plot identifies outliers as data points outside the whiskers.

- This dataset contains outliers with abnormally high packet counts relative to the bulk of observations.

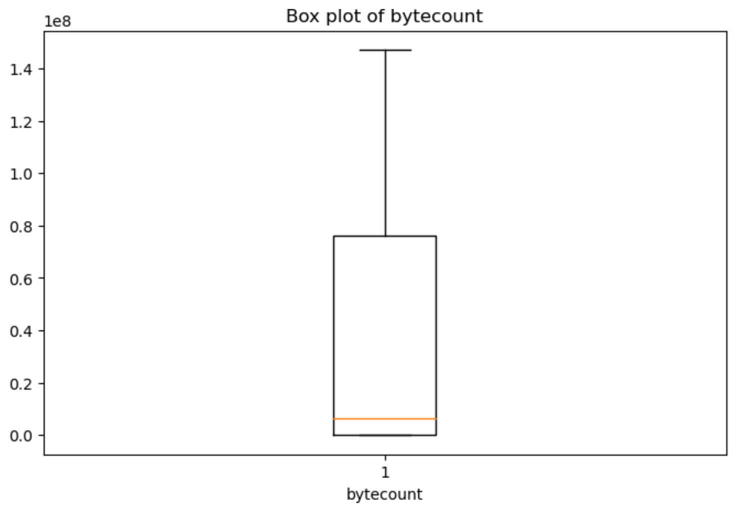


Figure 4. 13: Box plot of Byte Count

- Figure 4.13 'bytecount' box plot shows the dataset's byte counts. No outliers beyond the whisker indicate uncommon values.

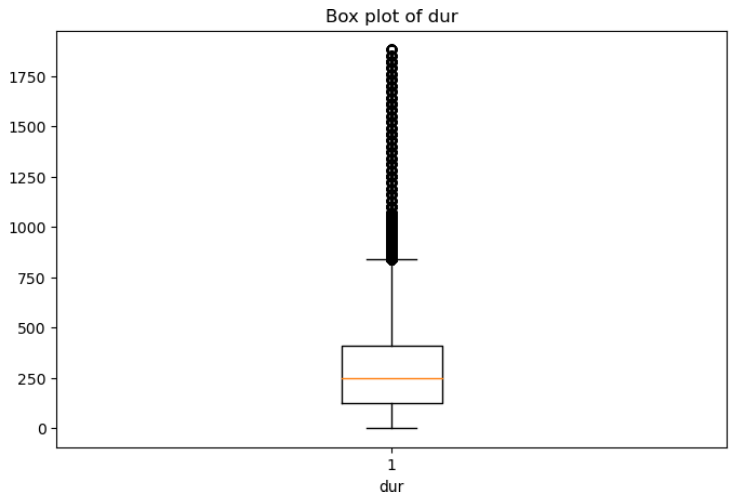


Figure 4. 14: Box plot of Duration

- Figure 4.14's "box plot" of "dur" (duration) shows how the durations are spread out across the data set.

- The box plot illustrates that the duration values are mostly spread out evenly, with only a few outliers being found.

- Outliers in "dur" are observations with lengths that are longer than the overall amount of data.

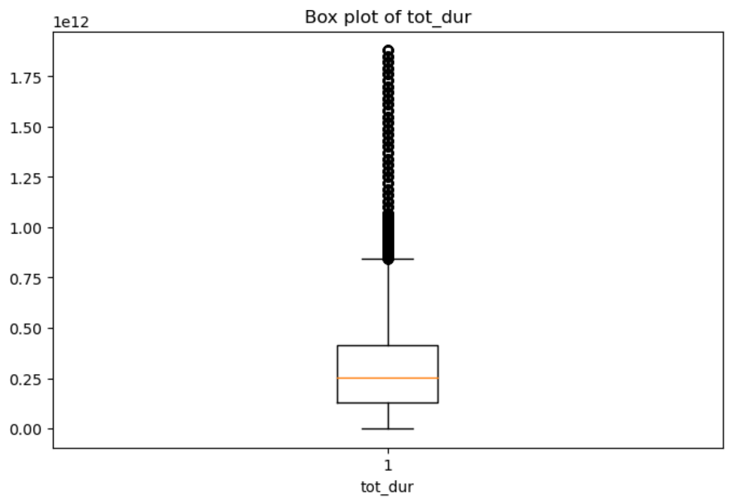


Figure 4. 15: Box plot of Total Duration

- The box plot of 'tot dur' (total duration) that can be seen in figure 4.15 provides an illustration of the distribution of total durations that are included within the dataset.

- Comparable to the 'dur' , the box plot exhibits a distribution of total duration values that is largely stable, with the exception of a few outliers that are identified.

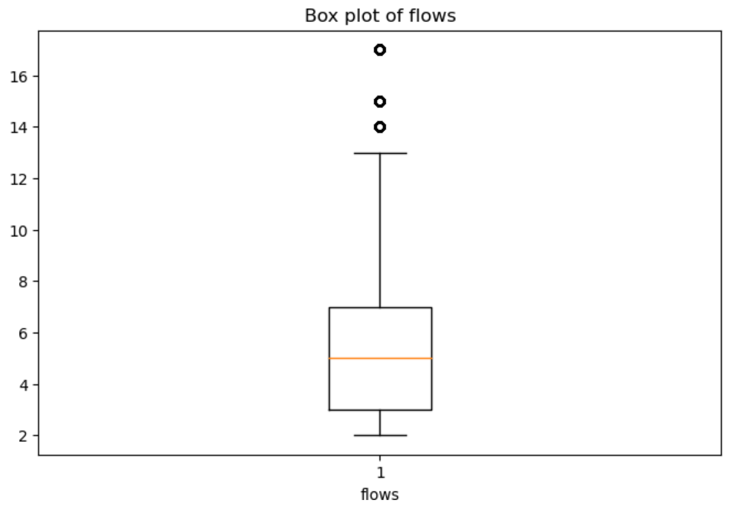


Figure 4. 16: Box plot of Flows

- The distribution of flow counts throughout the dataset is shown in figure 3.16 by means of a box plot of the 'flows' variable.

- It is clear from the box plot that there are outliers that extend beyond the whiskers, which indicates there are cases with very high flow counts.

- One possible interpretation of these outliers is that they are unusual variations in flow counts in comparison to the bulk of data.

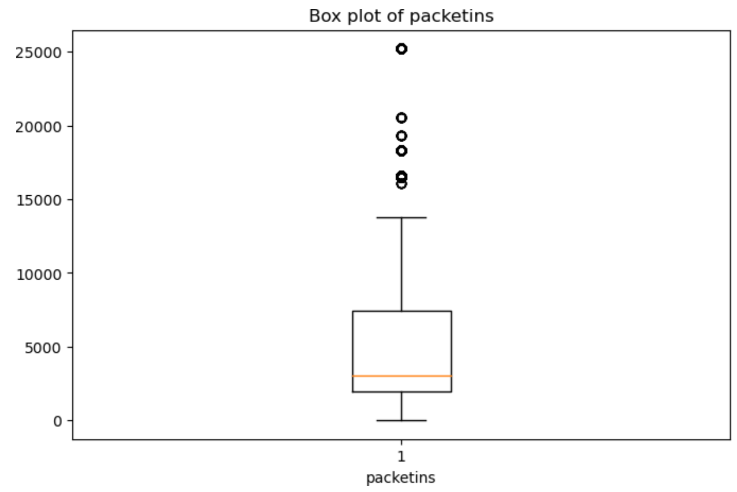


Figure 4. 17: Box plot of packet-In

- The distribution of packet insertions throughout the dataset is shown by the box plot of "packetins" in figure 4.17.

- The box plot displays outliers outside of the whiskers, which are indicative of excessive packet insertion counts, much as other features.

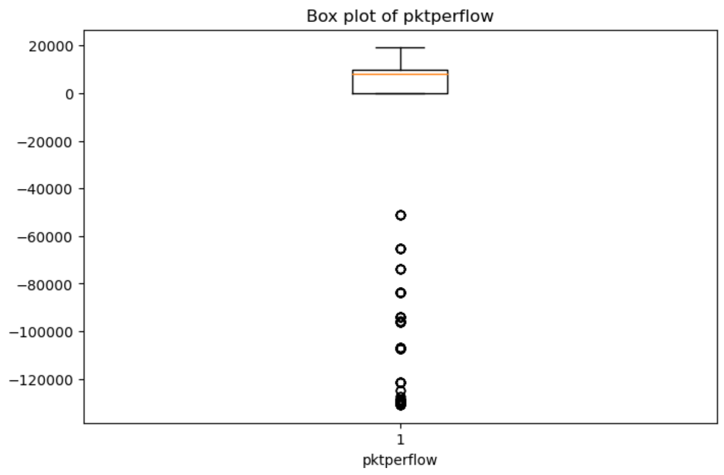


Figure 4. 18: Box plot of Packet Per Flow

- Figure 4.18's "pktperflow" box plot shows the distribution of packet per flow counts throughout the dataset.

- The box plot displays outliers that extend over the whiskers, indicating instances of very low packet counts each flow.

- Outliers in the "pktperflow" column indicate occurrences when the packet-per-flow rate is unusual in relation to the bulk of observations.

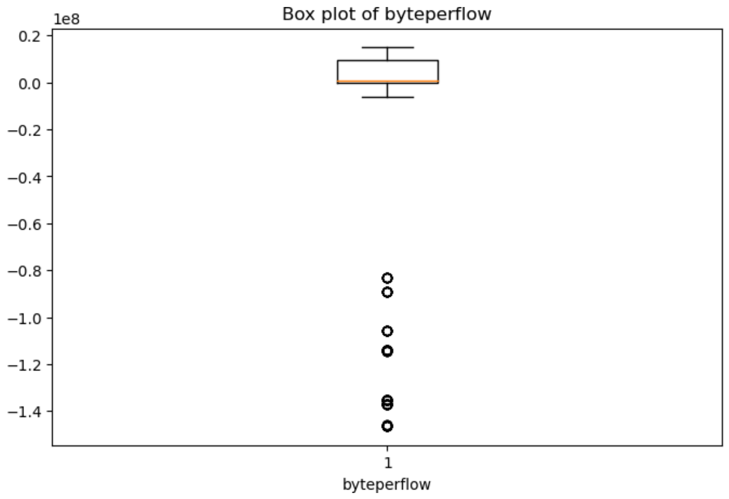


Figure 4. 19: Box plot of Byte Per Flow

- The box plot in Figure 4.19 depicts the distribution of byte per flow counts in the dataset.

- The box plot displays outliers beyond the whiskers, showing occurrences with very low byte per flow counts, much like other features.

- Outliers in the 'byteperflow' variable indicate occurrences that have significantly lower byte per flow rates relative to the majority of the dataset.

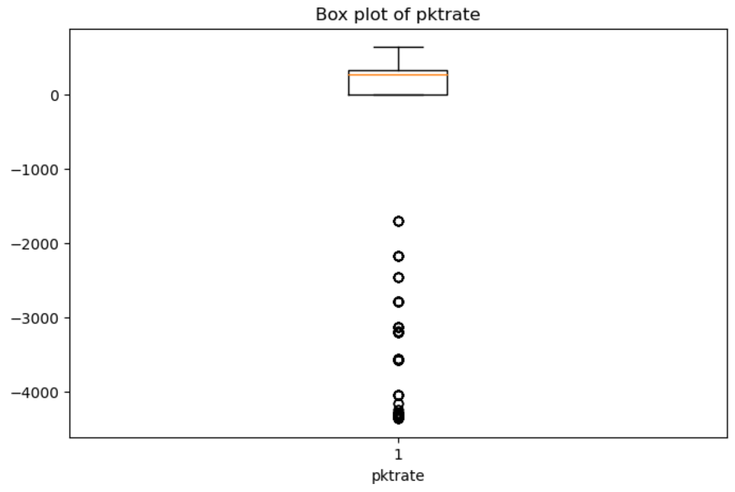


Figure 4. 20: Box plot of Packet Rate

- The box plot in figure 4.20 shows the distribution of packet rates in the dataset.
- The box plot displays outliers that are located outside the whiskers, showing instances with very low packet rates.
- Outliers in the 'pktrate' variable indicate cases with packet transmission rates that are significantly different from the bulk of data.

Analysis:

Box plots provide useful insights into the distribution of numerical features and aid in determining the presence of outliers or excessive values in the dataset. Outliers in features like 'pktcount', 'bytecount', 'flows', 'packetins', 'pk-tperflow', 'byteperflow', and 'pktrate' may suggest instances with aberrant behavior or characteristics.

4- Kernel Density Estimate (KDE) Plot

Kernel Density Estimate (KDE) plots assess a continuous random variable's probability density function easily. This research used KDE plots for every numerical column to examine data distribution and discover patterns or structures.

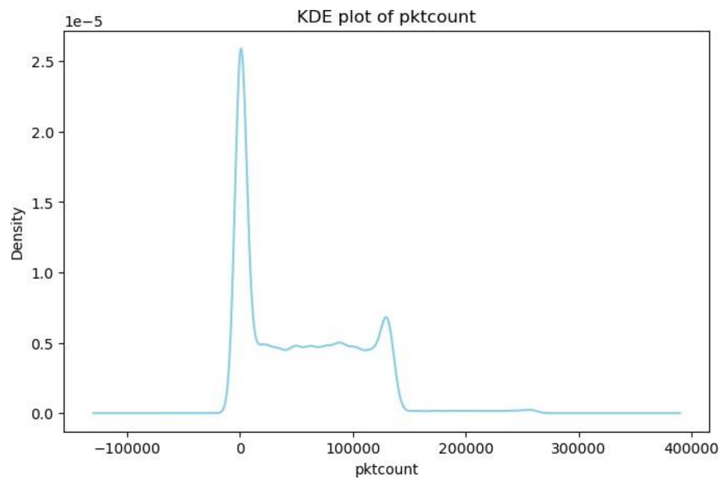


Figure 4. 21: Density Estimation of Packet Count

- The KDE plot of the variable 'pktcount' displays the distribution of packet counts throughout the dataset.

- The figure exhibits a maximum point in the lower range of values, suggesting a greater concentration of observations with lower packet counts.

- As the number of packets increases, the distribution slowly declines, indicating a lower density of observations.

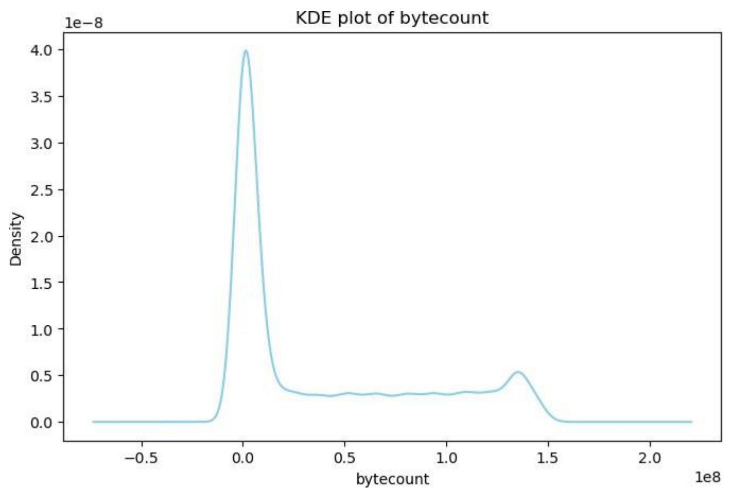


Figure 4. 22: Density Estimation of Byte Count

- The KDE plot of the variable 'bytecount' displays the distribution of byte counts in the dataset.

- As with 'pktcount', the figure peaks at lower byte counts, suggesting a larger density of observations with lesser transmission.

- The density falls steadily as the number of bytes increases, indicating a decline in the frequency of observations with larger byte transmission.

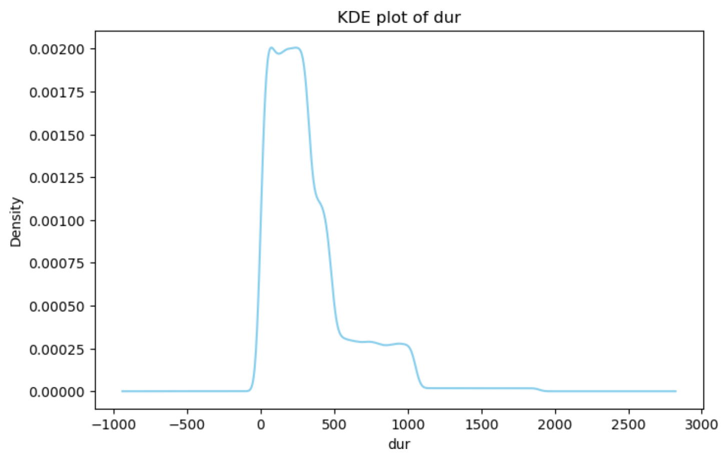


Figure 4. 23:Density Estimation of Duration

- An illustration of the distribution of durations throughout the dataset is shown by the KDE plot of the variable 'dur' (duration).

- The graph shows that the duration values are distributed in a smooth manner, with the exception of a few apparent peaks on the lowest points.

- This distribution indicates that the durations within the dataset are distributed in a manner that is rather uniform, and there are no noticeable patterns or structures that the dataset contains.

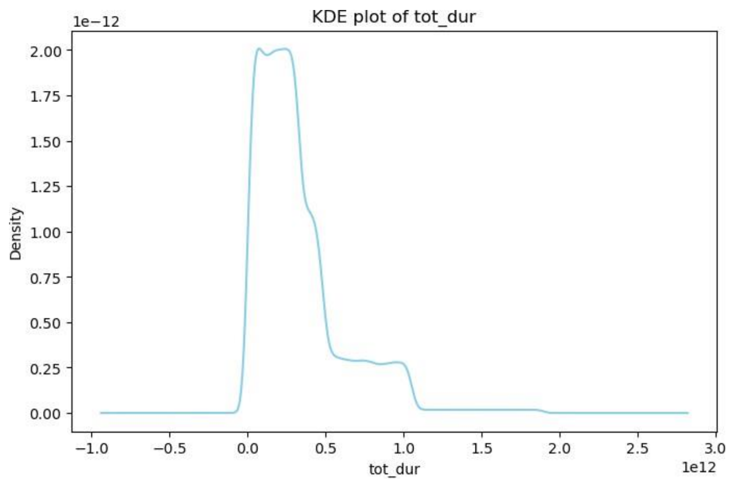


Figure 4. 24: Density Estimation of Total Duration

- A representation of the distribution of total durations within the dataset is shown by the KDE plot of "tot dur," which stands for "total duration.

- The plot displays a smooth distribution of total duration values, with clear peaks and valleys, much as the 'dur'

- The data set does not exhibit any noteworthy patterns or structures, and its distribution suggests that the total ratio values are distributed in a manner that is reasonably equal over the whole dataset.

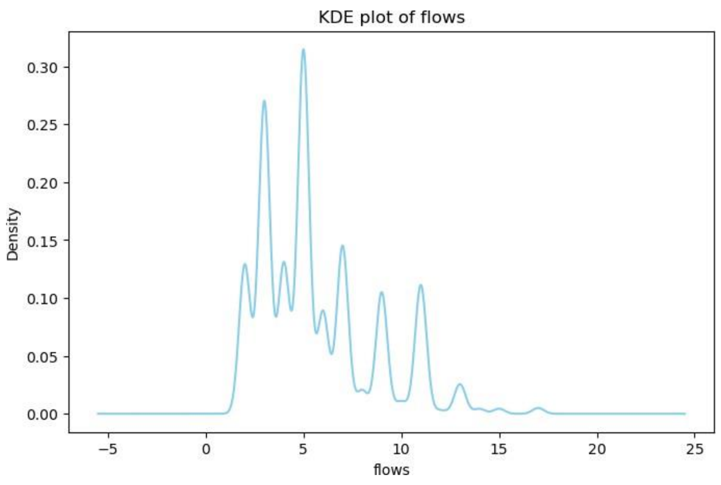


Figure 4. 25: Density Estimation of Flows

- The KDE plot of 'flows' provides a representation of the distribution of flow counts throughout the dataset.

- A peak may be seen in the plot around flow counts that are lower, which indicates that there is a larger density of observations with flow rates that are lower.

- The frequency distribution steadily drops as flow counts increase, which suggests that there is a reduction in the frequency of observations with increasing flow rates.

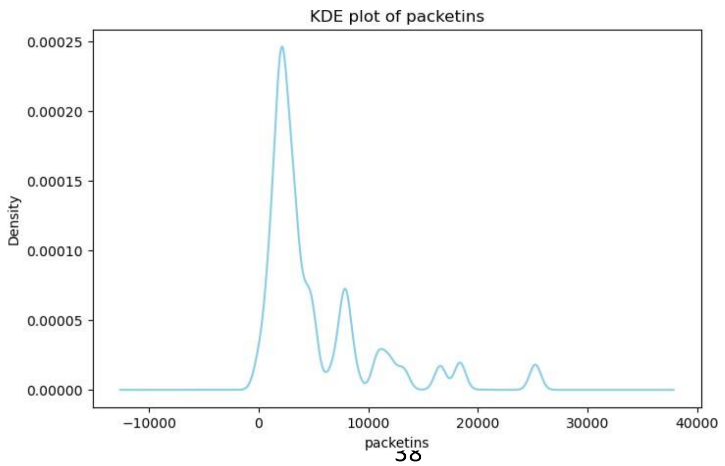


Figure 4. 26: Density Estimation of Packet In's

- The KDE plot of 'packetins' displays the distribution of packet insertions in the dataset.
- A peak in the graphic indicates a larger density of observations with lower packet insertion rates, like other features.
- As packet insertion rates rise, the density falls, indicating a reduction in observation frequency.

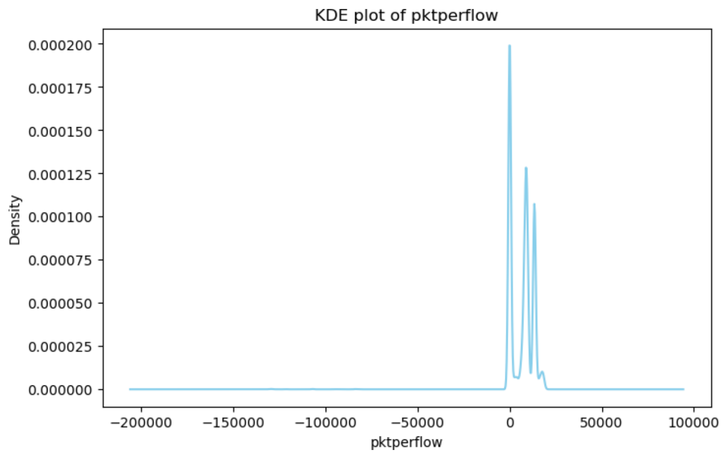


Figure 4. 27: Density Estimation of Packet Per Flow

- The KDE plot of 'pktperflow' displays the distribution of packet counts throughout the dataset.
- The figure shows a peak at higher packet per flow rates, indicating a larger density of observations with higher counts.

- As packet per flow rates rise, density declines, indicating fewer observations with greater counts.

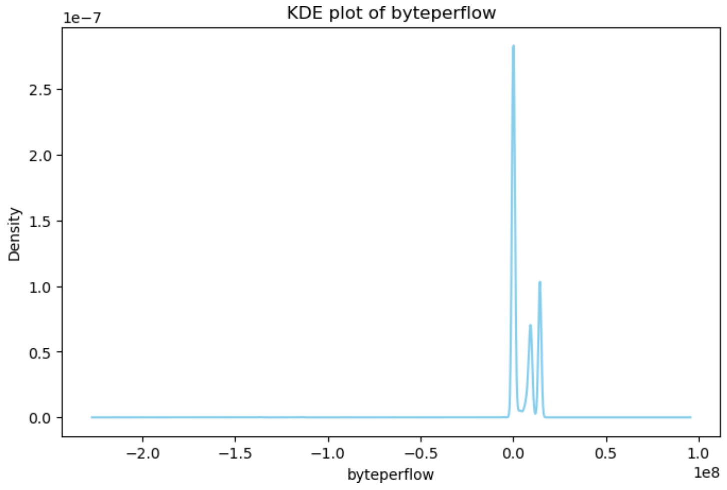


Figure 4. 28: Density Estimation of Byte Per Flow

- This KDE plot displays the distribution of byte per flow counts in the dataset.
- The figure shows a peak at higher byte per flow rates, suggesting a larger density of observations with higher counts.
- As byte per flow rates rise, density declines, suggesting fewer observations with greater counts.

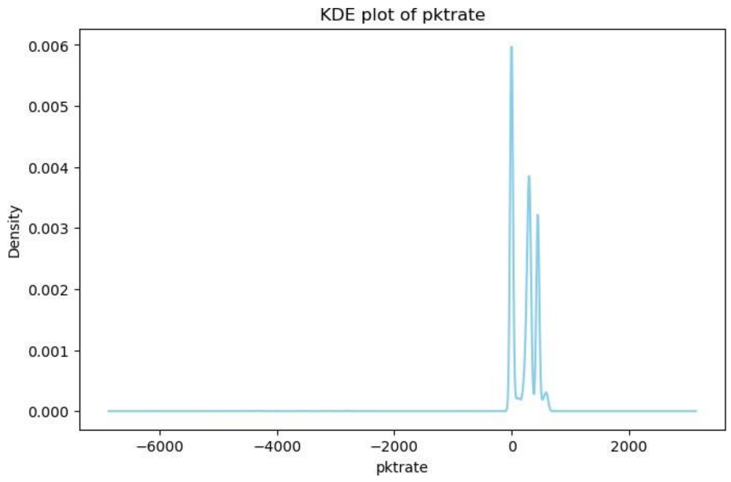


Figure 4. 29: Density Estimation of Packet Rate

- The KDE plot of 'pktrate' shows the distribution of packet rates in the dataset.
- The figure shows a peak at lower packet transmission rates, supporting a larger observation density.
- As packet rates rise, density falls, indicating fewer observations at greater rates.

Analysis:

Kernel Density Estimate (KDE) plots visualize numerical feature distributions by predicting their probability density functions easily. KDE plot patterns and structures reveal dataset characteristics and help explain feature data distribution.

5- Pie Chart for Exploring the Distribution of Categorical Variables

Pie charts are used to represent the distribution of categorical information in a visual manner. To get insight into the distribution of protocols in the dataset, we generate a pie chart just for the 'Protocol' column.

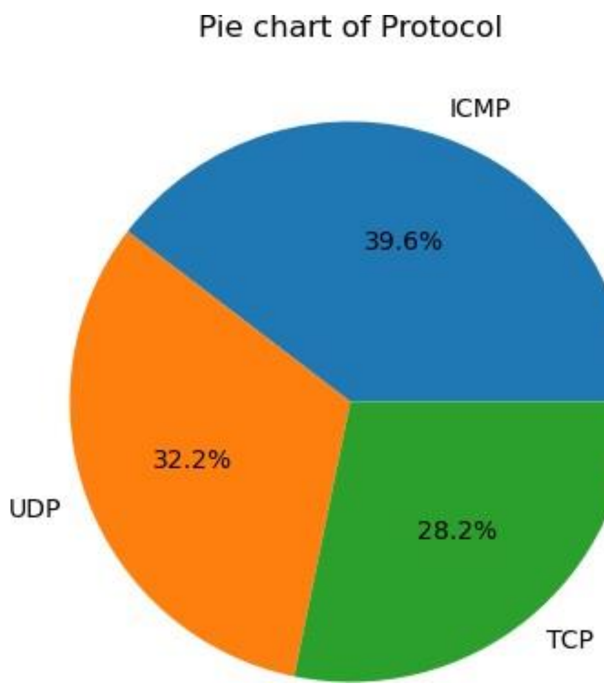


Figure 4. 30: Protocol Distribution in Dataset

Observations:

- The pie chart in figure 4.30 shows the distribution of protocols within the dataset. It demonstrates the proportion for every protocol type, with ICMP having the greatest percentage, accounting for almost 40%.

- This graphic facilitates comprehension of the comparative occurrence of every protocol in the dataset.

- The pie graphic indicates that TCP is the least common protocol, representing about 28% of the sample.

4.10 Exploratory Data Analysis (EDA) through Bi-variate Analysis

Bivariate analysis examines the relationships between two variables in a dataset, providing insights into how these variables affect or interact with one another. This analysis is essential for analyzing the hidden patterns and connections in the data.

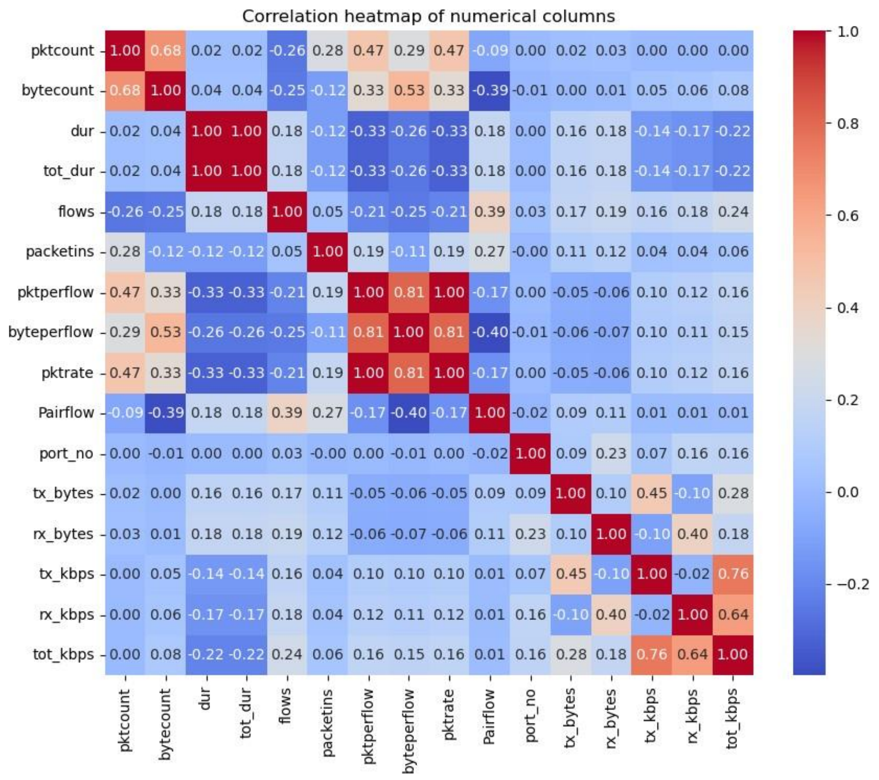


Figure 4. 31: Correlation heatmap of numerical columns

- Figure 4.31 correlation heatmap detailed the dataset's numerical column associations. The correlation coefficient between two numerical variables is shown in each heatmap cell. A correlation coefficient around 1 shows a significant positive connection, meaning one variable rises as the other rises. A correlation value around -1 suggests a high negative connection, indicating that one variable grows as the other decreases.

Pairplot for Selected Numerical Columns

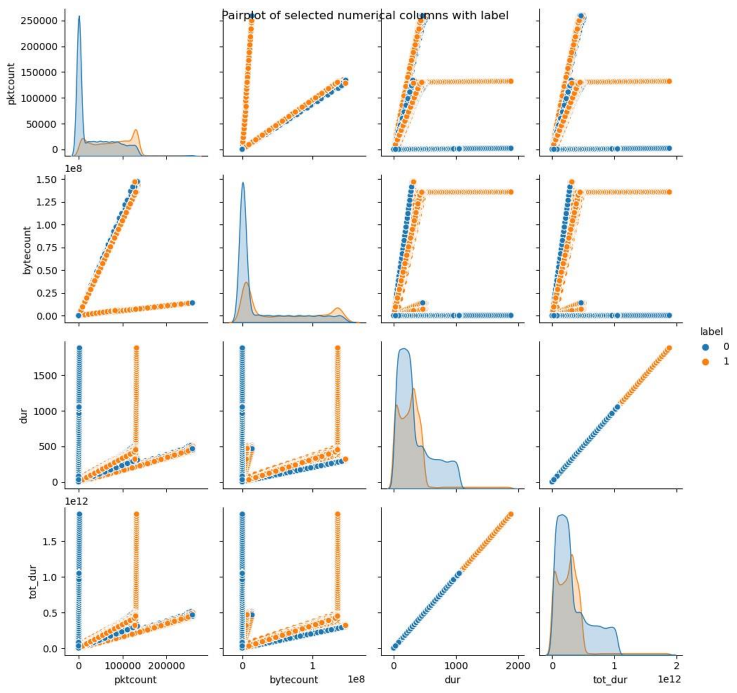


Figure 4. 32: Pairplot of selected numerical columns

- "pktcount," "bytecount," "dur," and "tot dur" are the four numerical columns that are shown in Figure 4.32's pairplot, which provides a visual representation of the relationships that exist between the chosen pairings of these columns. Every scatterplot in the pairplot illustrates the correlation between two quantitative variables, with one variable shown on the x-axis and another displayed on the y-axis.

- The connection between two numerical variables is represented by each scatterplot in the pairplot. One of the variables is drawn on the x-axis, while another variable is displayed on the y-axis. In addition to this, the diagonal panels provide the distribution of each individual numerical column, which offers insights into the distributions of the columns themselves as well as probable outliers.

4.11 Data Pre-Processing

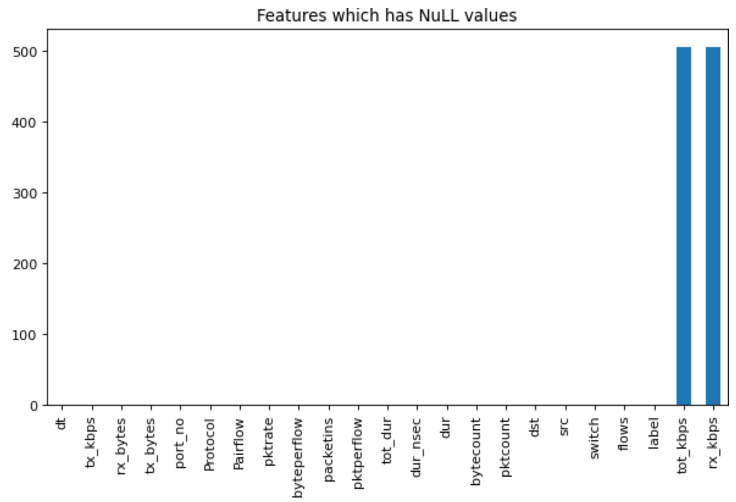


Figure 4. 33: Features with Null Values

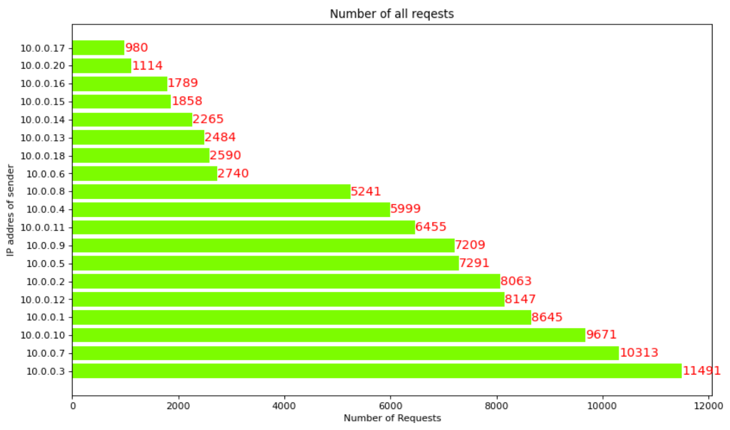


Figure 4. 34: Number of Requests from All IP Addresses

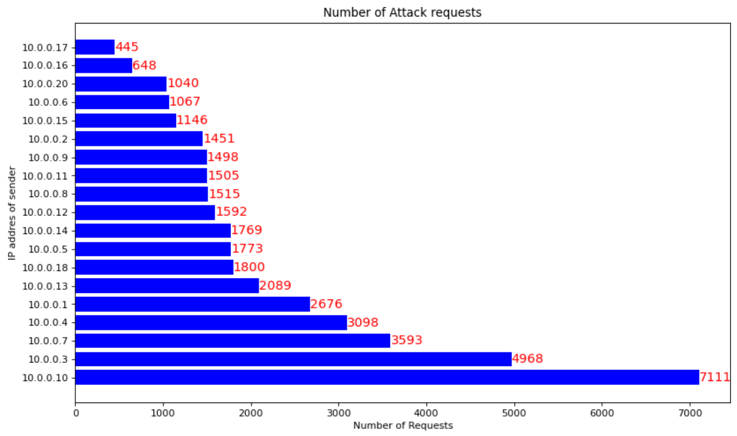


Figure 4. 35: Number of Attack Requests

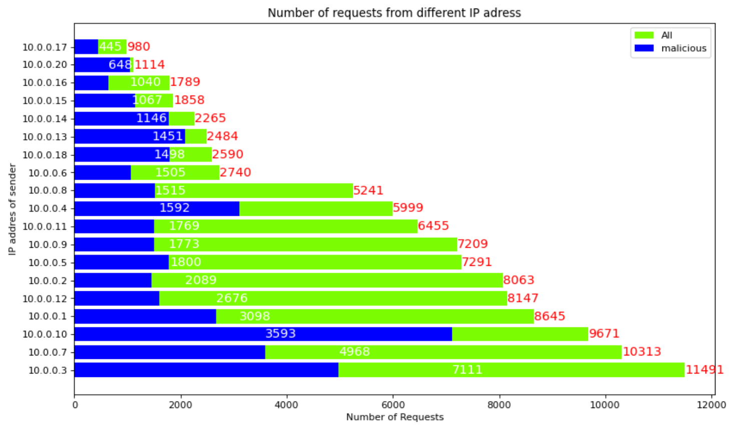


Figure 4. 36: Comparison of Requests between All and Malicious IP Addresses

In the process of data analysis and modeling, the pre-processing of data is an essential phase, Before anything else, it is essential to fix any missing values. It is possible to get insight into the level of missingness throughout several columns via the visualization of features that have null values for themselves.

On display in Figure 4.33 is a bar plot that illustrates the features that include null values. In addition, we use the isnull().sum() function to determine the precise number of null values that are present for every column. The missing values in certain columns, such as "rx kbps" and "tot kbps," are then imputed by replacing them with the mean values of the respective columns. This process is repeated until all of the data columns are complete.Following this, the distribution of IP addresses within the dataset is analyzed, with a particular emphasis placed on the number of requests that originate from certain IP addresses. Visualization is a useful tool for determining whether or not there are any patterns or irregularities in the distribution of network traffic. The purpose of this comparison is to identify any substantial differences that may exist between the request frequencies of all IP addresses and those that are related with suspicious behavior inside network.

The number of requests coming from all IP addresses is shown in Figure 4.34, whereas the number of requests coming solely from IP addresses linked with malicious behavior is shown in Figure 4.35. By doing an analysis of these visualizations, we are able to recognize any trends or irregularities in the distribution of requests

The comparison of request frequencies between all IP addresses and those associated with malicious activity is shown in a thorough manner in Figure 4.36.

4.12 Feature Engineering and Selection

Feature engineering and selection are crucial data preparation steps for enhancing machine learning models. These techniques refine and curate the data set in order to choose its most relevant attributes for model training.

4.12.1 Data Splitting

The first stage involves dividing the dataset into separate components: the feature matrix X and the target variable y. The target variable, often referred to as 'label', is the key element in classification tasks, indicating whether a network activity is harmless or harmful. Simultaneously, columns that are considered useless for the modeling process, such as 'rx kbps', 'tot kbps', and 'dt', are excluded from the feature set.

Subsequently, categorical variables are subjected to ordinal encoding to convert them into a numerical representation that is suitable for machine learning techniques. Afterwards, the data is divided into training and testing sets using stratified splitting, with a ratio of 75-25. This division allows for the training of the model using the provided training data and the subsequent assessment of its performance on new, unseen test data.

4.12.2 ML Model Training

Machine learning models are used to categorize network activity as either benign or malicious, depending on the collected attributes. We use four distinct models for training: Random Forest, K-Nearest Neighbors (KNN), XGBoost, and Logistic Regression. Every model has specific features that make it appropriate for this particular challenge.

4.12.3 Random Forest

Random Forest is a type of ensemble learning that uses more than one decision tree to generate predictions. It doesn't get too good at fitting data and works well with data that has a lot of dimensions. Random Forest can be used in this case because it can handle a lot of features and figure out complicated relationships between those features and the goal variable.

Hyperparameters:

n _estimators stands for the number of trees in the forest. Higher numbers can make things work better, but they may also make training take longer.

4.12.4 K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm is a straightforward and intuitive classification method. It operates by identifying the k closest data points in the training set and categorizing the test data according to the most common class among its neighboring points. The KNN algorithm is appropriate for our situation due to its ability to handle non-linear decision boundaries and its lack of assumptions about the underlying distribution of the data.

Hyperparameters:

n_neighbors: The number of neighbors to consider. Increasing this value may enhance robustness, but it can also lead to excessive smoothness. We use a value of k equal to 3 for our dataset and issue classification.

4.12.5 XGBoost

XGBoost is a very efficient and scalable implementation of gradient boosting machines. The algorithm operates by iteratively incorporating decision trees to reduce the loss function. XGBoost is an ideal choice for this challenge because to its ability to effectively manage enormous datasets and provide accurate predictions.

Hyperparameters:

Different hyperparameters, including learning rate, maximum tree depth, and regularization parameters, were adjusted to enhance performance.

4.12.6 Logistic Regression

Logistic Regression is a linear classification approach that estimates the likelihood of a binary outcome. Due to its computing efficiency and interpretability, this method is well-suited for situations involving huge datasets.

Hyperparameters:

The regularization parameter, C, was changed to mitigate overfitting.

4.12.7 Model Evaluation

We assess the efficacy of each machine learning model by using diverse criteria, such as accuracy.

Model	**Accuracy**
Random Forest	0.998
KNN	0.963
XGBoost	1.000
Logistic Regression	0.850

Table 4. 1: Model Evaluation Result

The results shown in table 4.1 demonstrate that the Random Forest and XGBoost models attained the maximum level of accuracy, with scores of 0.998 and 1.000, respectively. These models exhibit exceptional performance in categorizing network activity as either benign or malicious. The KNN model demonstrated strong performance, with an accuracy rate of 0.963. Nevertheless, Logistic Regression demonstrated worse accuracy in comparison to the other models, attaining a score of 0.850.

In summary, the findings suggest that the Random Forest, XGBoost, and KNN models are very effective in identifying malicious network behaviors in this dataset. Our technique involves a thorough approach to addressing the complexities of network intrusion detection. We start by doing data preparation, where we carefully address any missing values and encode categorical variables to ensure that the dataset is prepared for modeling. Feature engineering and selection are crucial processes in which we carefully choose essential features and divide the data into training and testing sets for future model training and assessment. By using a diverse range of machine learning methods such as Random Forest, KNN, XGBoost, and Logistic Regression, we develop powerful models that can accurately differentiate between harmless and harmful network behavior.

In the next part, we will examine the findings generated from our trained models and conduct a thorough comparative study. Through careful examination of diverse performance indicators such as accuracy, precision, recall, and F1-score, our objective is to acquire profound understanding of the efficacy of each algorithm and choose the most appropriate technique for network intrusion detection. This thorough assessment will provide useful insights into the merits and limitations of each model, informing future efforts to improve the security and resilience of network systems.

Chapter 5 : Results

The following classification reports give a comprehensive examination of the performance of each machine learning model in categorizing network activity as either benign (0) or malicious (1).

5.1 Random Forest Classification Report

	Precision	Recall	F1-score	Support
Benign (0)	1.00	1.00	1.00	15807
Malicious (1)	1.00	1.00	1.00	10280

Table 5. 1: Random Forest Classification Report

Precision, recall, and F1-score of 1.00 for both classes were excellent for the Random Forest model. This shows that the model categorized all benign and malicious network events. The Random Forest model's accuracy was 1.00, proving its ability to discriminate the two groups.

5.2 KNN Classification Report

	Precision	Recall	F1-score	Support
Benign (0)	0.97	0.97	0.97	15807

Malicious (1)	0.95	0.95	0.95	10280

Table 5. 2: KNN Classification Report

The KNN model performed well, with accuracy, recall, and F1-scores of 0.97, 0.97, and 0.97 for benign and 0.95, 0.95, and 0.95 for malicious. A 0.96 accuracy rate indicates good KNN model effectiveness in categorizing network activity.

5.3 XGBoost Classification Report

	Precision	Recall	F1-score	Support
Benign (0)	1.00	1.00	1.00	15807
Malicious (1)	1.00	1.00	1.00	10280

Table 5. 3: XGBoost Classification Report

Like the Random Forest model, the XGBoost model has flawless precision, recall, and F1-score of 1.00 for both classes. This shows that the XGBoost model correctly identified all benign and malicious internet activity. The XGBoost model detected malicious activity with 1.00 accuracy.

5.4 Logistic Regression Classification Report

	Precision	Recall	F1-score	Support
Benign (0)	0.86	0.90	0.88	15807
Malicious (1)	0.84	0.77	0.80	10280

Table 5. 4: Logistic Regression Classification Report

Logistic Regression performed somewhat worse than Random Forest and XGBoost. The benign class had 0.86, 0.90, and 0.88 precision, recall, and F1-score, whereas the malicious class had 0.84, 0.77, and 0.80. The Logistic Regression model classified network activity moderately with an accuracy of 0.85.

After analyzing the outcomes derived from our trained models, it is clear that each approach has unique advantages and disadvantages when used to network intrusion detection. The Random Forest model exhibits outstanding accuracy, precision, and recall, reaching classification performance that is close to flawless.

The KNN model, albeit less accurate than Random Forest, nonetheless demonstrates commendable performance across all measures. The simplicity and convenience of implementation of this option make it suitable for contexts that have low processing resources. However, the XGBoost model is notable for its exceptional accuracy and adaptability, demonstrating perfect classification performance in all areas. The software's capacity to manage intricate information and adjust to shifting circumstances makes it highly suitable for dynamic network environments with developing threat landscapes. However, the Logistic Regression model, while providing satisfactory performance, is not as accurate or effective as its rivals in terms of accuracy and recall. However, its simplicity and ability to be understood make it a desirable option for situations where the transparency and ease of understanding of the model are of utmost importance.

It can be inferred that each model offers distinct benefits, addressing various needs and limitations in network intrusion detection. Organizations may make educated decisions on the most suitable model for enhancing network security by carefully considering the trade-offs between accuracy, complexity, and interpretability, and how they fit with their unique goals and objectives.

Chapter 6 : Conclusions & Recommendations

6.1 Conclusions

The objective of this study was to create and assess machine learning models that can identify network intrusions. This was done by using a varied dataset that includes information on network traffic and activity. By following a methodical procedure that includes data preparation, feature engineering, model training, and assessment, we have effectively built many classification models, such as Random Forest, KNN, XGBoost, and Logistic Regression.

The assessment of these models demonstrated significant levels of accuracy, precision, and recall, especially when using the Random Forest and XGBoost methods. The results emphasize the effectiveness of ensemble learning methods and gradient boosting algorithms in distinguishing between harmless and harmful network traffic, thereby improving cybersecurity defenses.

Professionals in the field of cybersecurity may use the knowledge acquired from this study to improve their defensive strategies by integrating machine learning-powered intrusion detection systems into their current infrastructure. The interpretability of certain models, such as Logistic Regression, provides valuable understanding of the underlying elements that contribute to network intrusions. This understanding allows firms to build focused countermeasures and tactics to mitigate risks.

6.2 Recommendations and Future Work

Despite the encouraging outcomes of our research, it is crucial to acknowledge several limitations that may limit the generalizability of our conclusions. These include the possibility of overfitting with complicated models, biases present in the dataset, and the dynamic nature of cyber threats that might change over time. In order to overcome these obstacles, future research should make use of more representative and varied datasets, sophisticated ensemble methodologies, and real-time monitoring to stay up to date with evolving threat environments.

References/Bibliography

1. Alzahrani, R., & Alzahrani, A. (2021). Security analysis of DDOS attacks using machine learning algorithms in networks traffic. *Electronics*, *10*(23), 2919. https://doi.org/10.3390/electronics10232919

2. Bandara, K., Abeysingle, T. Hijaz, A., Darshana, D., Aneez, H., Kaluarachchi, S., Sulochana, K. & Dhammearatchi, D. (2016). *Preventing ddos attack using data mining algorithms*. Retrieved from https://www.ijsrp.org/research-paper-1016/ijsrp-p5857.pdf

3. Behal, S., Kumar, K., & Sachdeva, M. (2021). *D-FAC: A novel /-Divergence based distributed DDoS defense system.* Retrieved from https://www.sciencedirect.com/science/article/pii/S1319157817304111

4. Chen, Z., Jiang, F., Cheng, Y., Gu, X., Liu, W.& Peng, J. (2018). *XGBoost classifier for ddos attack detection and analysis in sdn-based cloud.* Retrieved from https://ieeexplore.ieee.org/document/8367124?denied=

5. Devi, K., Preetha, G. & Selvaram, G. (2014). *An impact analysis: real time ddos attack detection and mitigation using machine learning.* Retrieved from https://ieeexplore.ieee.org/document/6996133?denied=

6. Filho, F., Silveira,F. Junior,A. Solar, G. & Silveira, L. (2019). *Smart detection: an online approach for dos/ddos attack detection using machine learning.* Retrieved from https://downloads.hindawi.com/journals/scn/2019/1574749.pdf?_gl=1*u1as3h*_ga*NTIwNjgxOTczLjE2OTUxODgwNTU.*_ga_NF5QFMJT5V*MTY5OTgxNTMyMy4zLjAuMTY5OTgxNTMyMy42MC4wLjA.&_ga=2.20649848.1597351220.1699797596-520681973.1695188055

7. Gu, Y., Guo.Z & Wang, Y.(2019). *Semi supervised k-means ddos detection method using hybrid feature selection algorithm.* Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8717648

8. Johnson, J. & George, S. (2022). *Review on ddos detection using machine learning.* Retrieved from https://www.ijert.org/research/review-on-ddos-detection-using-machine-learning-IJERTCONV10IS04056.pdf

9. J. Hou, P. Fu, Z. Cao and A. Xu, "Machine Learning Based DDos Detection Through NetFlow Analysis," MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 2018, pp. 1-6, doi: 10.1109/MILCOM.2018.8599738

10. Kadam, N., Sekhar, K. (2021). *Machine Learning Approach of Hybrid KSVN Algorithm to Detect DDoS Attack in VANET.* Retrieved from

https://thesai.org/Downloads/Volume12No7/Paper_82-Machine_Learning_Approach_of_Hybrid_KSVN_Algorithm.pdf

11. Li, Q., Meng, L., Zhang, Y., Yan, J. (2019). DDoS Attacks Detection Using Machine Learning Algorithms. In: Zhai, G., Zhou, J., An, P., Yang, X. (eds) Digital TV and Multimedia Communication. IFTC 2018. Communications in Computer and Information Science, vol 1009. Springer, Singapore. https://doi.org/10.1007/978-981-13-8138-6_17

12. Lee, K., Kim, J., Kwon, K. H., Han, Y., & Kim, S. (2008). DDoS attack detection method using cluster analysis. Expert systems with applications, 34(3), 1659-1665.

13. M. Tayyab, B. Belaton and M. Anbar, "ICMPv6-Based DoS and DDoS Attacks Detection Using Machine Learning Techniques, Open Challenges, and Blockchain Applicability: A Review," in IEEE Access, vol. 8, pp. 170529-170547, 2020, doi: 10.1109/ACCESS.2020.3022963.

14. Mihoub, A., Ben Fredj, O., Cheikhrouhou, O., Derhab, A., & Krichen, M. (2022). Denial of service attack detection and mitigation for the Internet of Things using looking-back-enabled machine learning techniques. Computers and Electrical Engineering, 98, 107716. https://doi.org/10.1016/j.compeleceng.2022.107716

15. Nalayinil, C., & Katravan,J. (2022). *Detection of ddos attack using machine learning algorithms.* Retrieved from https://deliverypdf.ssrn.com/delivery.php?ID=716097073084077006113114070098118004042021055002019085109020126093009002126106006018021054116024117056041064104029107101094066055058046000054086098103065082125126126093026076125068083000125124103094068095117008123004103094108001112075006110091067096017&EXT=pdf&INDEX=TRUE

16. Pei, J., Chen, Y. & Ji, W. (2019). *A ddos attack detection method based on machine learning.* Retrieved from https://iopscience.iop.org/article/10.1088/1742-6596/1237/3/032040/pdf

17. Prriyadarshini, M. & Devi,s. (2020). *Detection of ddos attacks using supervised learning technique.* Retrieved from https://iopscience.iop.org/article/10.1088/1742-6596/1716/1/012057/pdf

18. P. Khuphiran, P. Leelaprute, P. Uthayopas, K. Ichikawa and W. Watanakeesuntorn, "Performance Comparison of Machine Learning Models for DDoS Attacks Detection," *2018 22nd International Computer Science and Engineering Conference (ICSEC),* Chiang Mai, Thailand, 2018, pp. 1-4, doi: 10.1109/ICSEC.2018.8712757.

19. Patel, M., Amritha, P. P., Sudheer, V. B., & Sethumadhavan, M. (2024). DDoS Attack Detection Model using Machine Learning Algorithm in Next Generation Firewall. Procedia Computer Science, 233, 175-183. https://doi.org/10.1016/j.procs.2024.03.207

20. Qin, X., Xu, T. & Wang, C. (2015). *Ddos attack detection using flow entropy and clustering technique.* Retrieved from https://ieeexplore.ieee.org/document/7397119?denied=

21. Robinson, R. & Thomas, C. (2015).*Ranking of machine learning algorithms based on the performance in classifying ddos attacks.* Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7488411

22. Sanjeetha, R., Raj, A., Saivenu, K., Ahmed, M., Sathvik, B. & Kanavalli, A. (2021). *Detection and mitigation of botnet based DDoS attacks using catboost machine learning algorithm in SDN environment.* Retrieved from https://www.proquest.com/openview/22db9871cc1d369ff954187a5a853de8/1?pq-origsite=gscholar&cbl=2037694

23. Suresh, M. & Anitha, R. (2011). *Evaluating machine learning algorithms for detecting ddos attacks.* Retrieved from https://link.springer.com/chapter/10.1007/978-3-642-22540-6_42

24. Sambangi, S., & Gondi, L. (2020). A machine learning approach for DDoS (Distributed Denial of Service) attack detection using multiple linear regression. Proceedings, 63(1), 51. https://doi.org/10.3390/proceedings2020063051

25. S. Wankhede and D. Kshirsagar, "DoS Attack Detection Using Machine Learning and Neural Network," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697702.

26. Saghezchi, F. B., Mantas, G., Violas, M. A., de Oliveira Duarte, A. M., & Rodriguez, J. (2022). Machine Learning for DDoS Attack Detection in Industry 4.0 CPPSs. Electronics, 11(4), 602. https://doi.org/10.3390/electronics11040602

27. Wang, M., Lu, Y., & Qin, J. (2020). A dynamic MLP-based DDoS attack detection method using feature selection and feedback. Computers & Security, 88, 101645.

28. Yoachimik, O., Desgats, J., Forster, A. (2023). *Cloudflare mitigates record breaking 71 milion request per second ddos attack.* Retrieved from

https://blog.cloudflare.com/cloudflare-mitigates-record-breaking-71-million-request-per-second-ddos-attack/?

29. Ye, J., Cheng, X., Zhu, J., Feng, L., & Song, L. (2018). A DDoS attack detection method based on SVM in software defined network. Security and Communication Networks, 2018.

30. Zekri, M., El Kafhali, S., Aboutabit, N., & Saadi, Y. (2017, October). DDoS attack detection using machine learning techniques in cloud computing environments. Paper presented at the IEEE Conference on Cloud Computing, DOI: 10.1109/CloudTech.2017.8284731.