Journal of Technology and System Information Vol: 2, No 4, 2025, Page: 1-22

Detection of DDOS Attacks in Software-Based Systems in Cyberspace Using Machine Learning

Zeynep Dolmaz, Ilkay Cinar*

Selçuk University, Faculty of Technology, Department of Computer Engineering

DOI:
https://doi.org/10.47134/jtsi.v2i4.5033
*Correspondence: Ilkay Cinar
Email: ilkay.cinar@selcuk.edu.tr

Received: 03-08-2025
Accepted: 14-09-2025
Published: 28-10-2025

Abstract: *Distributed Denial of Service (DDoS) attacks pose a severe threat to the reliability and stability of modern network infrastructures, often resulting in significant service disruptions and financial losses. Therefore, developing efficient and accurate detection methods has become essential for maintaining service continuity in large-scale networks. This study addresses this critical problem by proposing an integrated approach that combines feature engineering techniques with machine learning algorithms to enhance DDoS attack detection. In the first stage, ANOVA and Chi-Square tests were applied to the dataset to identify statistically significant features, and attributes such as* dt, switch, dur, bytecount, *and* pktcount*—which contributed minimally to classification performance or contained redundant information—were excluded. The optimized feature set was then evaluated using Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Logistic Regression (LR) algorithms. Experimental results showed that feature selection improved SVM accuracy from 74.88% to 95.05%, increased Decision Tree accuracy to 99.94%, slightly reduced KNN performance while maintaining stability, and decreased LR accuracy from 77.15% to 74.87%. Overall, the findings demonstrate that the proposed approach effectively enhances detection accuracy while reducing computational cost, offering a practical and efficient solution for DDoS attack detection.*

Keywords: *Machine Learning, SDN, DDoS Detection, Feature Engineering*

Introduction

Distributed Denial of Service attacks are a significant cyber threat in which numerous compromised devices simultaneously generate high volumes of traffic, exhausting the resources of the target system and rendering services unavailable to legitimate users. These attacks can exploit bandwidth saturation, protocol vulnerabilities, or the application layer, causing service disruption, financial loss, and security risks in critical systems (Aslan, 2022).

While traditional network infrastructures are vulnerable to DDoS attacks, Software-Defined Networks (SDNs) offer advantages through centralized control and dynamic management, enabling rapid detection and effective mitigation. The SDN architecture, in which the control plane is separated and network devices are managed by a centralized unit, enhances flexibility and manageability, allowing faster detection of network events and more efficient execution of mitigation processes (Kreutz et al., 2014). However, SDNs are also exposed to security threats, and the central controller itself may become a target (Liu et al., 2022).

With the rapid advancement of technology, large-scale cyber-attacks worldwide have become increasingly sophisticated and impactful, affecting governments, major corporations, and individuals economically, politically, and socially (Catak & Mustacoglu, 2019). Notable examples include the 2010 Stuxnet attack on Iran's nuclear facilities, which disrupted centrifuge operations and caused significant financial damage (Holat, 2021). The effects of Stuxnet were not only technical but also political. The attack illustrated how a state could employ cyber operations to target another state's nuclear program, marking one of the first concrete examples of state-sponsored cyber warfare and raising global awareness of the threats posed by such attacks (Çetinkaya & Terzi, 2024).

In 2014, following Russia's annexation of Kırım and the subsequent separatist movements in Donbas, Ukraine became the center of both physical and digital conflicts. During this period, APT (Advanced Persistent Threat) groups allegedly linked to or supported by Russia launched numerous attacks targeting Ukrainian public institutions, critical infrastructure, and media outlets (Aslan, 2022). Immediately before and during the Russia–Ukraine conflict, the websites of Ukrainian ministries, banks, and public institutions were subjected to DDoS attacks. These attacks aimed to disrupt citizens' access to critical information and generate public panic. Additionally, certain government websites were hacked to disseminate propaganda (Paltacı, 2022).

Traditional countermeasures, including signature-based detection, traffic volume analysis, and threshold-based alerts, are often inadequate against zero-day attacks and low-volume persistent threats (Giuzio et al., 2019).

Signature-based systems detect activities that match known attack patterns stored in a database. While they can rapidly identify known threats, they are ineffective against previously unknown or unsigned attacks (Holm, 2014). Zero-day attacks exploit previously unknown vulnerabilities in software or systems and cannot be detected by signature-based solutions, posing significant risks to critical infrastructure and large networks and emphasizing the need for adaptive, behavior-based security measures (Karaman et al., 2020).

To address these challenges, this study proposes an integrated framework combining feature engineering with machine learning algorithms for DDoS detection. The dataset used in this study differs structurally from commonly used datasets in the literature, and the optimized feature set was evaluated using Decision Tree, SVM, K-Nearest Neighbors, and Logistic Regression algorithms. This approach aims to improve detection accuracy while maintaining computational efficiency.

Recent literature demonstrates that machine learning and deep learning methods outperform conventional approaches in DDoS detection by capturing discriminative patterns in large-scale traffic data. For example, Çatak and Mustacoğlu (2019) integrated autoencoder-based feature extraction with a deep neural network (DNN) classifier on NSL-KDD, reporting 97% accuracy (Catak & Mustacoglu, 2019). In a software-defined networking SDN simulation, Deepa et al. (2019) employed an ensemble of KNN, Naïve Bayes, SVM, and SOM, achieving 98.12% accuracy (Deepa et al., 2019). Using CIC-DDoS2019, Aytaç et al. (2020) compared multiple ML algorithms and found KNN, Logistic Regression, and Bernoulli Naïve Bayes each exceeded 99% accuracy (Aytaç et al., 2020).

Evaluating CIC-IDS2017 in an SDN setting, Nadeem et al. (2022) reported 99.97% accuracy with Random Forest (Nadeem et al., 2022). On CIC-DDoS2019, Yağmur (2023) tested ESA and KNN, obtaining 92.43% and 96.30% accuracy, respectively (Yağmur, 2023). For CSE-CIC-IDS2018, Topbaş (2024) combined KNN and a 1D-CNN with hybrid sampling, yielding accuracies above 97% (Topbaş, 2024). Most recently, Wang et al. (2024) introduced DDoS-MSCT, a multi-scale CNN–Transformer architecture, reaching 99.94% accuracy on CIC-DDoS2019 and CIC-IDS2017, underscoring the advantage of modern DL hybrids (Wang et al., 2024). Collectively, these studies highlight that representation learning (e.g., autoencoders), ensemble strategies, imbalance-aware sampling, and CNN–Transformer integration substantially elevate DDoS detection performance across diverse datasets and network contexts.

Table 1: Comparison of studies on DDoS detection in the literature

Year	Author(s)	Dataset	Accuracy (%)
2019	Çatak & Mustacoğlu	NSL-KDD	97.00
2019	Deepa et al.	SDN-Simülasyon	98.12
2020	Aytaç et al.	CIC-DDoS2019	>99.00
2022	Nadeem et al.	CIC-IDS2017 (SDN)	99.97 (RF)
2023	Yağmur	CIC-DDoS2019	92.43 (ESA), 96.30 (KEYK)
2024	Topbaş	CSE-CIC-IDS2018	>97.00
2024	Wang et al.	CIC-DDoS2019, CIC-IDS2017	99.94

As shown in Table 1, a comparative summary of recent studies conducted for the detection of DDoS attacks is presented in terms of publication year, datasets used, and achieved accuracy rates. The reviewed studies predominantly employ CIC-based datasets, with reported accuracy rates ranging from 97% to 99.97%. Notably, in recent years, the adoption of deep learning and hybrid models has led to significant improvements in accuracy. However, these widely used benchmark datasets in the literature have limitations in terms of generalizability and real-time applicability. This study aims to make a substantial contribution to the literature by utilizing a unique and real-time dataset.

The proposed study delivers more balanced, efficient, and reliable results compared to the unidirectional approaches commonly found in the literature. The use of a novel dataset enables the evaluation of attacks under more realistic conditions, while features enriched through feature engineering enhance the performance of the machine learning models. Thus, the study presents a more effective, faster, and practically applicable solution for the detection of DDoS attacks.

Methodology

This section provides a detailed explanation of the methods followed in the dataset preparation, feature engineering, and modeling process. Initially, missing values in the dataset were addressed to prevent potential information loss that could adversely affect model accuracy. During the data preprocessing stage, gaps observed in numerical features were imputed using the arithmetic mean of the respective columns. In particular, missing values in the rx_kbps and tot_kbps features were filled using their mean values. The mean

imputation method preserves the overall distribution of the dataset, prevents excessive distortion of variance, and contributes to maintaining data integrity. Moreover, this approach ensures the preservation of sample size, allowing for more reliable and statistically significant results during model training.

In this study, a model was proposed by integrating feature engineering and machine learning techniques for anomaly detection on network traffic data. A systematic approach was followed, particularly in data preprocessing, the numerical encoding of categorical variables, and feature selection steps (Söğüt & Erdem, 2023).

The dataset was divided into two subsets, with 70% allocated for training and 30% for testing, to ensure that the model sees a sufficient number of examples during the learning process while also enabling an objective evaluation of its generalization capability on unseen data. This split provides adequate data for training while allowing for statistically meaningful assessment on the test set. The flowchart illustrating the methodology of this study is presented in Figure 1.

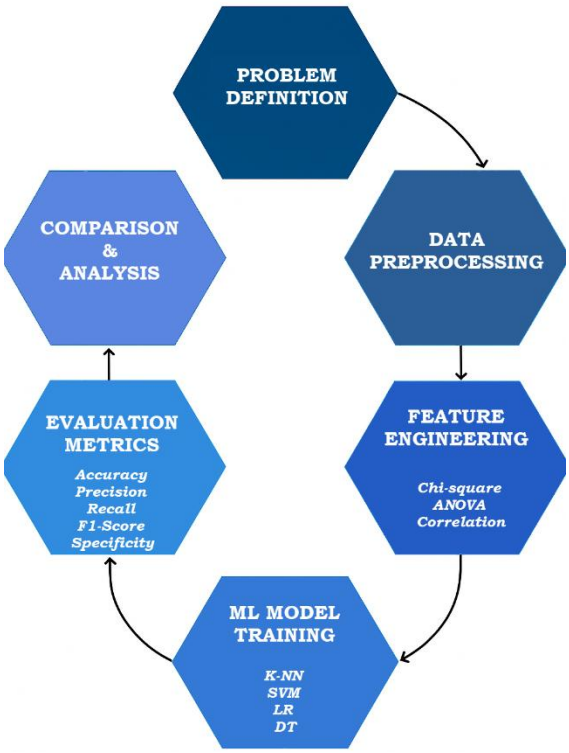

Figure 1. Flow diagram of study

When evaluated collectively, the Chi-square test, ANOVA, and correlation matrix analyses indicated that the features byteperflow, packetins, port_no, rx_kbps, and tx_kbps did not exhibit a significant relationship with the target variable. Furthermore, the dt feature, due to its timestamp nature and high correlation, was found to introduce redundant information into the model. Consequently, these features were excluded from the modeling process, achieving dimensionality reduction and enabling the model to attain a more streamlined, stable, and highly generalizable structure.

Following these feature engineering decisions, the remaining significant features were used for training machine learning algorithms. This approach ensured that models

were built using only attributes exhibiting statistically strong relationships with the target variable, thereby reducing computational cost and enhancing classification performance.

The training duration of each model was recorded, and performance metrics including accuracy, precision, recall, and AUC were calculated on the test dataset. Additionally, model performance was visualized using the Confusion Matrix and ROC curves. This methodology not only aims to improve model accuracy and generalizability but also seeks to enhance computational efficiency by eliminating redundant features, thereby increasing processing speed. As a result, the developed models were optimized to provide both high accuracy and efficient runtime performance.

In this study, various supervised machine learning algorithms were applied to detect DDoS attacks within software-defined network traffic. The employed methods included K-Nearest Neighbors Logistic Regression, Decision Trees, and Support Vector Machines. These algorithms were selected due to their distinct mathematical foundations and differing advantages in classification performance. Each model was trained on a preprocessed and feature-optimized dataset, and their performance was subsequently evaluated.

In this study, the Materials and Methods section is structured around five key components to support the systematic execution of the research. First, the operating principles of software-defined networks are explained. In the second stage, the dataset used in the study is introduced, along with the characteristics of the data and the rationale for its selection. The third section presents in detail the feature engineering techniques applied to enhance the model's performance. In the fourth section, the proposed method and the applied methodology are comprehensively described. Finally, in the fifth section, the metrics used to evaluate the performance of the developed model are detailed.

Software-Defined Networks (SDN)

Software-Defined Networks are a modern network architecture designed to enhance flexibility, centralized control, and programmability in network management. SDN separates the control plane from the data plane, allowing decision-making functions to be handled by a centralized controller, while data transmission is managed by network devices. The architecture consists of three layers: the application layer (defines policies), the control layer (manages data flows), and the data layer (contains physical network devices). Inter-layer communication is facilitated via Northbound and Southbound APIs, enabling dynamic management and coordinated operations (Söğüt & Erdem, 2023).

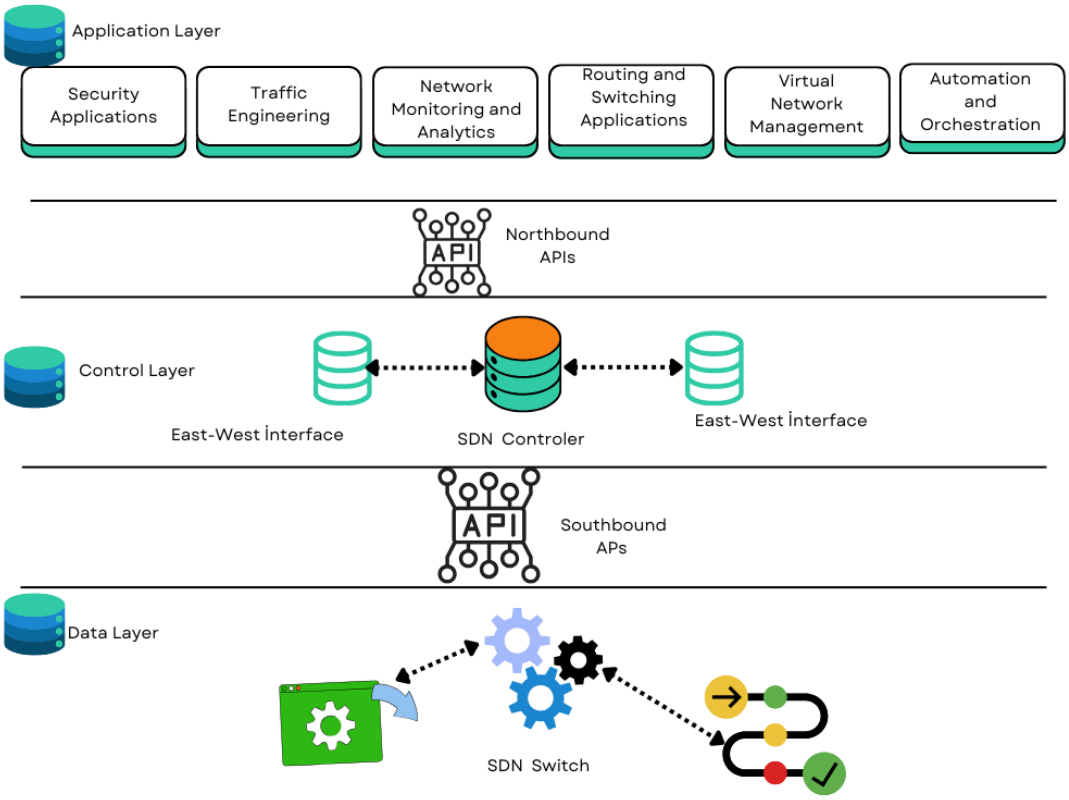

Figure 2. SDN Architecture

SDN systems are favored in academia and industry for enabling flexible, centralized, and automated network management. Unlike traditional hardware-dependent networks, SDN allows centralized software control, dynamic resource configuration, enhanced security, and rapid integration of new technologies, while supporting low-cost simulations and testing. Figure 2 illustrates the operation of the network architecture in software-defined systems (Ayodele & Buttigieg, 2024). However, the centralized controller introduces security risks, particularly from DDoS attacks, which can compromise all network traffic and disrupt services (Mahamat & Çeken, 2019).Machine learning–based anomaly detection can identify such threats early by analyzing statistical deviations in network traffic (Erhan & Anarım, 2020).

Dataset

Commonly used datasets for DDoS attack detection include CIC-DDoS2019, NSL-KDD, UNSW-NB15, CSE-CIC-IDS2018, and BoT-IoT. CIC-DDoS2019 is widely adopted due to its inclusion of multiple attack types (e.g., UDP, TCP, HTTP) and real traffic scenarios, though its complex structure requires extensive preprocessing (Moustafa & Slay, 2015). . NSL-KDD is outdated and provides limited protocol information, making it insufficient for modern attacks. UNSW-NB15 offers a more balanced distribution but lacks coverage for certain traffic types (Sharafaldin et al., 2018).

In this study, the "SDN-Dataset," which includes SDN-specific attack types and reflects contemporary network scenarios, was used. The dataset comprises 23 meaningful features combining timing, protocol, and flow information, minimizing unnecessary complexity during model training and improving performance. It contains 98,748 rows and 23 columns, covering 104,345 network traffic flows. Most features are numeric and include source/destination IPs, packet and byte counts, duration and bandwidth measures, as well as protocol and port information. This dataset provides both ease of analysis and a realistic representation of current network traffic (Sharafaldin et al., 2018; Moustafa & Slay, 2015). The dataset features and their descriptions are presented in Table 2.

Table 2: Dataset features and descriptions

No	Feature Name	Description
1	dt	Timestamp containing the date and time when the event was recorded
2	switch	Identifier of the network switch involved in the communication.
3	src	Source IP address of the network traffic.
4	dst	Destination (receiver) IP address of the network traffic.
5	pktcount	Total number of packets transmitted during the communication
6	bytecount	Total number of bytes transmitted during the communication.
7	dur	Duration of the communication in seconds.
8	dur_nsec	Duration of the communication in nanoseconds.
9	tot_dur	Total duration of the communication (seconds + nanoseconds)
10	flows	Total number of flows associated with this communication.
11	packetins	Number of received (incoming) packets.
12	pktperflow	Average number of packets per flow
13	byteperflow	Average number of bytes per flow.
14	pktrate	Packet transmission rate (packets / second).
15	Pairflow	Flow identifier defining the source–destination IP pair.
16	Protocol	Network protocol used in the communication (e.g., TCP, UDP).
17	port_no	Port number associated with the communication (e.g., 80, 443).
18	tx_bytes	Total number of transmitted bytes
19	rx_bytes	Total number of received bytes.
20	tx_kbps	Transmission rate in kilobits/second.
21	rx_kbps	Reception rate in kilobits/second.
22	tot_kbps	Total data rate in kilobits per second (tx + rx).
23	label	Label indicating whether the communication is related to a DDoS attack (e.g., 0: Normal, 1: DDoS).

The dataset contains detailed information related to network traffic. The src and dst columns represent the source and destination IP addresses within the network, while columns such as pktcount, bytecount, tx_bytes, and rx_bytes include the number of packets and bytes transmitted and received. For timing and performance measurements, duration-related variables such as dur, dur_nsec, and tot_dur, as well as bandwidth indicators such as tx_kbps, rx_kbps, and tot_kbps, are provided. The Protocol column specifies the protocol used in the communication (e.g., TCP, UDP), whereas port_no denotes the port number through which the connection is established. In addition, flow- and packet-based traffic

statistics, including flows, packetins, and pktrate, are also available. Table 3 presents a comparison between the most commonly used datasets in the literature and the dataset employed in this study.

Table 3: Dataset features and descriptions

Dataset	Number of Features	Timing	Protocol Information	Flow Information
CIC-DDoS2019	80+	✓	✓	✓
NSL-KDD	41	✗	Limited	✗
UNSW-NB15	49	Restricted	✓	✗
SDN-Dataset	23	✓	✓	✓

According to Table 3, although the CIC-DDoS2019 dataset possesses a highly detailed and rich set of features, its complex structure necessitates preprocessing. The NSL-KDD dataset, on the other hand, offers fewer attributes and limited protocol information; due to its outdated and low-resolution structure, it is insufficient for representing modern network attacks. While UNSW-NB15 provides a more balanced distribution, it still faces limitations stemming from the underrepresentation of certain traffic types.

An analysis of the source (src) and destination (dst) IP address variables indicates that network traffic is often concentrated around specific IPs, reflecting structured and potentially predictable patterns. Such concentrations, particularly from spoofed or targeted sources in DDoS attacks, serve as critical indicators of abnormal activity and enhance the accuracy of attack classification, while also facilitating efficient network monitoring, management, and security analysis.

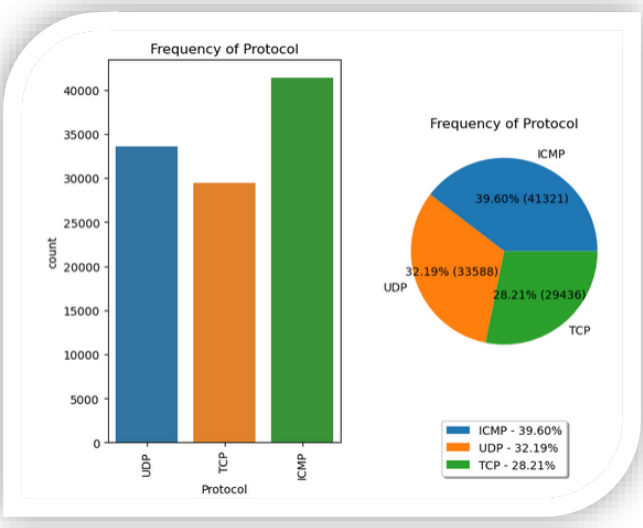

Figure 3. Dataset features (Protocol and IP Address) distribution

The distribution shown in Figure 3 illustrates that the three primary protocols in network traffic are represented in a balanced and diverse manner. ICMP (39.6%), UDP (32.2%), and TCP (28.2%) are all present in significant proportions, indicating that the dataset is rich in representing various types of communication. This diversity provides a

notable advantage for analyzing overall network behavior as well as for developing protocol-based security or performance models.

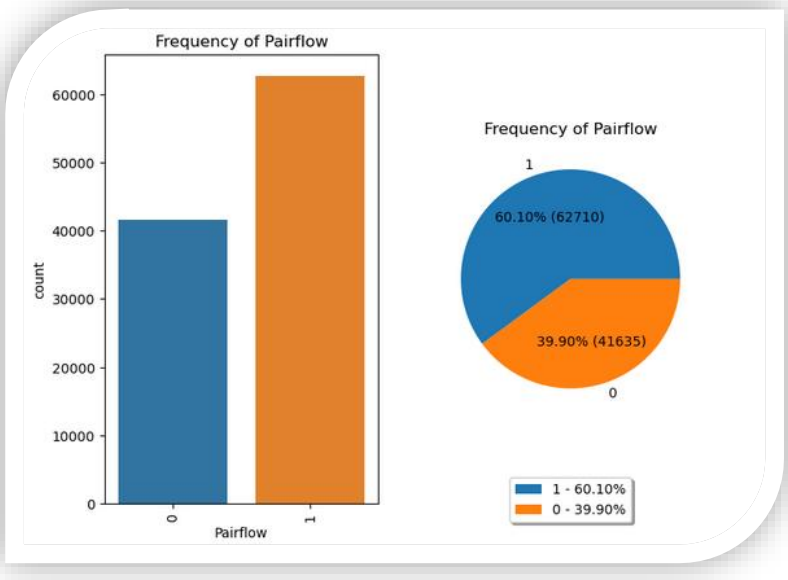

Figure 4. Dataset features (Pairflow)

The Pairflow variable in the dataset represents a flow identifier that corresponds to pairs of source and destination IP addresses. As illustrated in Figure 4, the analysis reveals that 60.1% (62,710 out of 104,345) of the total flows consist of matched or active IP pairs. This finding indicates that a significant portion of the network traffic occurs through well-defined and traceable connections. Such a structure suggests that the network system is highly manageable and controllable in terms of security monitoring, analytical assessment, and performance optimization, providing a solid foundation for accurate anomaly detection and efficient traffic management.

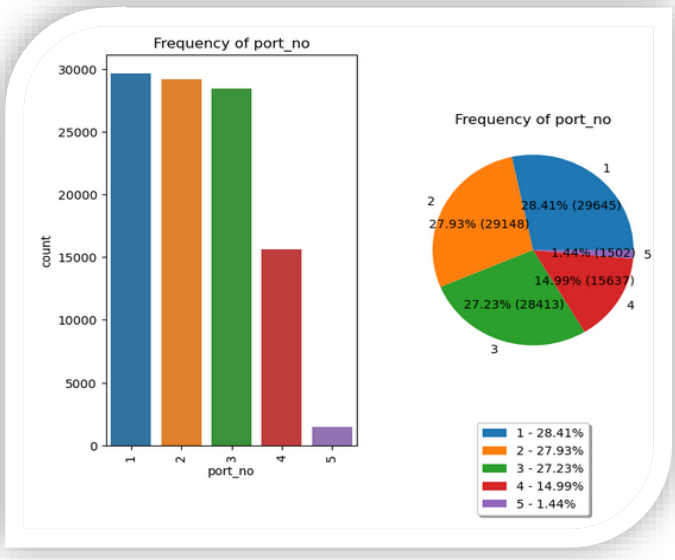

Figure 5. Dataset features (Port_no)

As shown in Figure 5, the network traffic is predominantly concentrated on the first three ports, with 83.6% of the total traffic occurring through ports 1, 2, and 3.

Feature Engineering

Chi-square (χ^2) test

The Chi-square (χ^2) test is a non-parametric statistical method used to assess the significance of relationships between categorical variables. In this study, it was applied during feature selection to evaluate the association between independent variables (features) and the dependent variable (class label: normal or DDoS traffic) (Farhana et al., 2023). Features with significant relationships ($p < 0.05$) were retained due to their higher discriminative power, while non-significant features were removed to reduce dimensionality and computational complexity, enhancing classification performance (Wang & Zhou, 2020).Table 4 lists features with the weakest association to the target variable.

Table 4: Features with the lowest significance according to the Chi-square test

Feature	Chi-kare Score	p-value
Byteperflow	0.0564	0.812
Packetins	0.1536	0.695
Port_no	0.5163	0.472
rx_kbps	1.1811	0.277
tx_kbps	1.1820	0.277

According to Table 4, these five features (byteperflow, packetins, port_no, rx_kbps, and tx_kbps) do not exhibit a statistically significant relationship with the dependent variable, as their p-values are above 0.05. These features do not contribute meaningfully to the classification performance of the model; on the contrary, they may introduce unnecessary complexity. Therefore, their removal is beneficial for dimensionality reduction and is expected to improve the generalization performance of the model (Wang & Zhou, 2020).

Analysis of variance (ANOVA) test

The ANOVA test is a robust statistical method used to examine the variance differences of independent variables with respect to the target variable.

Table 5: Features with the lowest level of significance according to the ANOVA test

Features	**P-value**
packetins	0.393
Port_no	0.126

Based on the results of the ANOVA test, the p-values of the variables port_no and packetins were determined as 0.126 and 0.393, respectively. Since these values exceed the statistical significance threshold of 0.05, it can be inferred that these features do not have a meaningful effect on the target variable. Therefore, these two variables were excluded from the modeling process to reduce the number of features and improve the overall model performance (Huang & Chen, 2008).

Correlation matrix

A correlation matrix shows the strength and direction of linear relationships between dataset features, with values ranging from -1 to +1 (Pearson correlation). Values near +1 or -1 indicate strong positive or negative relationships, while values near 0 suggest no linear relationship. Correlation matrices are used for feature selection and detecting multicollinearity, as highly correlated features can introduce redundancy and reduce model performance (Garcia-Ramirez et al., 2022). Figure 6 presents the correlation matrix of the dataset used in this study.

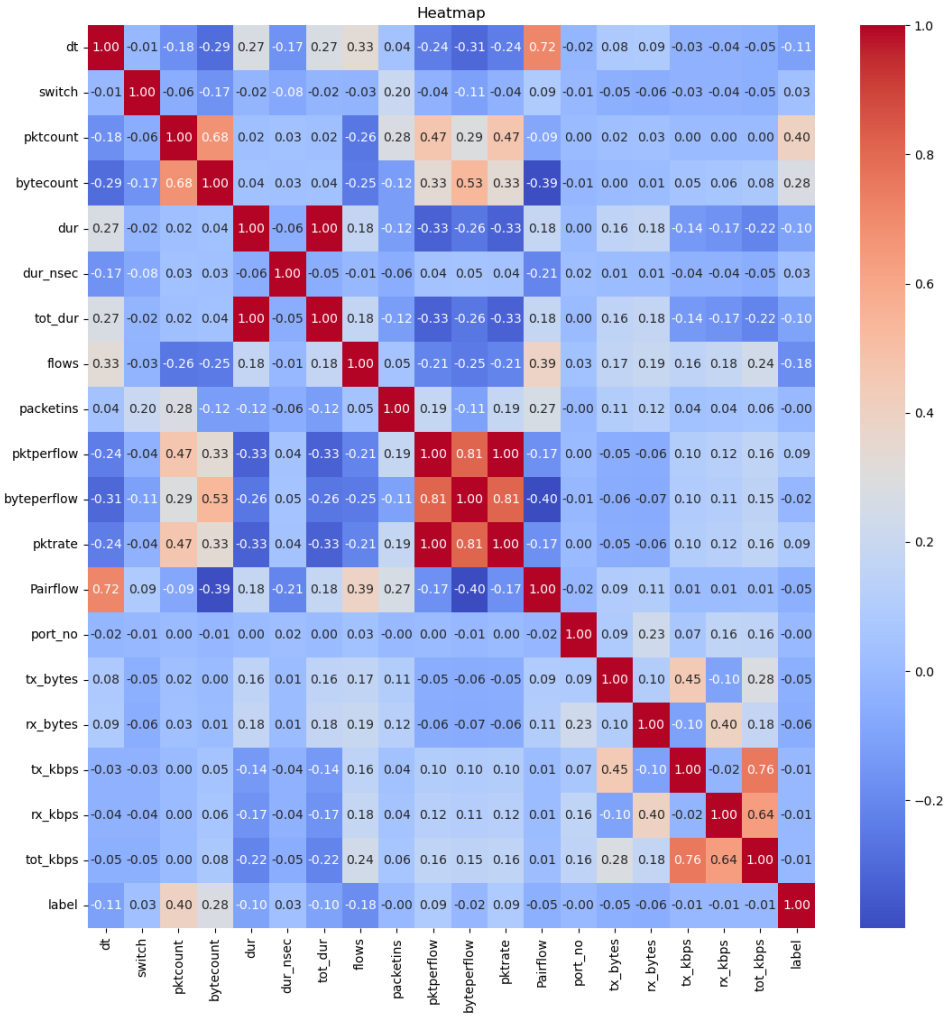

Figure 6. Correlation matrix of the features used in this study

The dt feature represents only the timestamp of events and does not directly reflect network behavior, limiting its contribution to classification. Its high correlation (0.72) with Pairflow may cause multicollinearity and time-dependent biases, reducing model generalization. Combined analyses using Chi-square, ANOVA, and correlation matrices indicated that dt, along with byteperflow, packetins, port_no, rx_kbps, and tx_kbps, lacked significant association with the target variable. Consequently, these features were removed, reducing dimensionality and improving model stability and generalization.

K-Nearest Neighbors (K-NN)

K-Nearest Neighbors is a widely used, non-parametric, distance-based classification technique among supervised learning algorithms. First introduced in 1951 (Eşidir, 2025), this method is particularly effective for classification and regression problems. The fundamental principle of the algorithm is that a sample to be classified is labelled according to the majority class of its k nearest neighbors in the training dataset (Cover & Hart, 1967). The training set used in the model is defined as X= {x1, x2, …, xn} where each xi ∈Rn represents a vector in an n-dimensional feature space. The corresponding target labels are defined as Y= {y1, y2, …, yn}. During the prediction process, the sample x to be classified is compared with all data points in the training set using a similarity metric. Typically, the Euclidean distance is employed for this purpose (Mahamat & Çeken, 2019). The distance between two data points a=(a1, a2) and b=(b1, b2) is given in Equation 1 (Sharafaldin et al., 2018):

$$d(a,b) = \sqrt{\{(a_1 - b_1)^2 + (a_2 - b_2)^2\}} \quad (1)$$

Based on the distance values obtained (Equation 1), the k nearest neighbors to the test sample are selected, and the class that appears most frequently among these neighbors is assigned to the test sample. This approach has proven particularly effective in areas such as pattern recognition, network traffic analysis, and anomaly detection.

Logistic Regression (LR)

Logistic regression is a parametric model widely used for binary classification problems, aiming to estimate the probability that a dependent variable belongs to a particular class. The model is based on transforming the output of a linear regression function through a logistic (sigmoid) function. This transformation normalizes the predicted values between 0 and 1, allowing them to be interpreted as probabilities. Thus, logistic regression directly estimates the likelihood that an observation belongs to a specific class. In network traffic analysis, logistic regression is frequently employed to determine whether incoming packets are normal or associated with a DDoS attack. The model is trained using features such as packet rate and source IP density, and new incoming data are evaluated based on this trained model (Moustafa & Slay, 2015). The fundamental function of the model is expressed in Equation (2).

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \quad (2)$$

The variables used in this model are defined as follows:

- $h_\theta(x)$: Represents the probability that the observation belongs to class 1.
- θ: Denotes the weight vector of the model, representing the learned coefficients for each feature.
- x: Refers to the input feature vector.
- θ^{Tx}: Represents the linear combination of the weights and the input features.

The log-loss (cost) function used to optimize the model is given in Equation (3).

$$J(\theta) = -(1/m)\Sigma[i = 1..m]\left[y^{(i)} log\left(h\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - h\theta\left(x^{(i)}\right)\right)\right] \quad (3)$$

The meanings of the variables are as follows:

- J(θ): The loss function of the model. A smaller value of this function indicates better model performance.
- m: The number of samples in the training dataset.
- $y^{(i)}$: The actual class label of the i-th observation (0 or 1).
- $h\theta(x^{(i)})$: The predicted probability that the i-th observation belongs to class 1.

Decision Trees (DT)

Decision trees are supervised learning algorithms used for both classification and regression problems. They operate by recursively partitioning the data based on specific features and making decisions at each node until a class label is reached at the leaf nodes. Starting from the root node, the model splits the data at each node according to an attribute, ultimately leading to a decision outcome. The primary objective throughout the decision process is to minimize uncertainty (impurity) within the dataset. For this purpose, entropy and information gain measures are employed. Entropy quantifies the impurity of the dataset and is calculated as shown in Equation 4.

$$Entropy(S) = -\Sigma p_k log2(p_k) \quad (4)$$

Meaning of the variables:

- Entropy(S): A measure of the disorder (impurity) within the dataset.
- n: The number of classes.
- p_k: The probability that an instance in the dataset belongs to class k.

The information gain for a given attribute is defined as follows:

$$InformationGain(S, A) = Entropy(S) - \Sigma(|S_v|/|S|) * Entropy(S_v) \quad (5)$$

Meaning of the variables:

- Information Gain (S, A): The amount of information gained as a result of splitting the dataset based on attribute A.

- S: The entire dataset.
- S_v: The subset of S containing instances with the value v for attribute A.
- $|S_v|$: The number of instances in the subset.
- $|S|$: The total number of instances in the dataset.

Decision trees possess the ability to distinguish between normal and abnormal (attack) network traffic based on features such as packet count, port number, and connection duration. Owing to their interpretable structure, the decision-making processes of the model can be easily traced and explained. This characteristic enhances the reliability of intrusion detection and enables the rapid identification of DDoS attacks (Farhana et al., 2023).

Support Vector Machines (SVM)

Support Vector Machines are among the most powerful and widely used supervised learning algorithms for classification tasks. The fundamental principle of SVM is to determine the optimal decision boundary (hyperplane) that best separates data points belonging to different classes. This boundary is selected in such a way that it maximizes the margin between classes, thereby enhancing not only the model's fit to the training data but also its generalization capability. When the classes are not linearly separable, kernel functions are employed to project the data into higher-dimensional spaces, enabling the construction of complex, non-linear decision boundaries (Ivanova et al., 2022).

This property makes SVM an effective method for both low-dimensional and high-dimensional datasets. In network traffic analysis, SVM stands out for its high accuracy and generalization capacity in distinguishing between normal and attack traffic, making it a robust tool for detecting DDoS attacks (Çatak & Balaban, 2016).

Confusion Matrix

The confusion matrix presents the correct and incorrect predictions made by the model for each class in a tabular format. This representation provides deeper insight into the model's decision-making process by revealing the conditions under which it performs accurately or fails. In a binary classification setting, the confusion matrix is typically constructed for the classes "attack" and "normal traffic."

The main components of the confusion matrix are defined as follows:

1. True Positive (TP): Instances where the model correctly identifies actual DDoS attack samples as "attack" (Sarhan et al., 2021).
2. True Negative (TN): Instances where the model correctly classifies actual normal traffic as "normal" (Sarhan et al., 2021).
3. False Positive (FP): Instances where normal traffic samples are incorrectly classified by the model as "attack." This condition is considered a false alarm (Sarhan et al., 2021).
4. False Negative (FN): Instances where actual DDoS attack samples are misclassified by the model as "normal" traffic. This indicates that an attack has been missed (Sarhan et al., 2021).

These components form the foundation for understanding how effectively the model detects attacks and manages false alarm rates. The metrics derived from the confusion matrix enable a comprehensive evaluation of model performance. Table 6 presents the detailed descriptions and mathematical formulations of these performance metrics (Hossain, 2025; Zahid & Bharati, 2025).

Table 6: Details of performance metrics

Metric	Description	Formula
Accuracy	The ratio of correct predictions to all predictions.	$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$
Precision	The proportion of instances predicted as attacks that are actually attacks.	$\text{Precision} = \frac{TP}{TP + FP}$
Recall	The proportion of actual attacks that are correctly identified.	$\text{Recall} = \frac{TP}{TP + FN}$
F1-Score	The balanced average of precision and recall.	$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
Specificity	The proportion of actual normal traffic correctly identified.	$\text{Specificity} = \frac{TN}{TN + FP}$
False Positive Rate (FPR)	The proportion of normal traffic incorrectly classified as attacks.	$\text{FPR} = \frac{FP}{FP + TN}$

ROC Curve (Receiver Operating Characteristic Curve)

The ROC curve is a graphical method used to evaluate the performance of a model in binary classification problems across different decision thresholds. This curve illustrates the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR). The TPR indicates the proportion of actual attack traffic correctly detected, while the FPR represents the proportion of normal traffic incorrectly classified as an attack. The ROC curve allows for the analysis of how balanced and effective the model performs across various threshold values (Eşidir, 2025).

For instance, in DDoS attack detection, using a low threshold may enable the model to capture more actual attacks but increases the risk of misclassifying normal traffic. Conversely, a high threshold reduces false positives but may cause some real attacks to be missed. The ROC curve visually represents this trade-off across all threshold values, providing a comprehensive view of the model's discriminative ability (Suvra, 2025).

AUC (Area Under the Curve)

The AUC represents the area under the ROC curve and serves as a numerical performance metric summarizing the model's ability to distinguish between positive (attack) and negative (normal) classes. The AUC value ranges from 0 to 1; a value of 1 indicates perfect classification, while 0.5 corresponds to random guessing. Hence, models

with higher AUC values better delineate the boundary between attack and normal traffic (Özel & Demirsöz, 2021).

In this study, the SVM model's AUC value increased from approximately 0.82 before feature engineering to 0.99 afterward. This improvement indicates a significant enhancement in the model's capability to reduce false positives while maintaining the detection of true positives. Therefore, AUC is a critical metric for objectively and comparably evaluating model performance, particularly in imbalanced datasets (Ivanova et al., 2022).

Result and Discussion

In this section, the experimental findings obtained from the classification processes performed on the original dataset using the proposed hybrid model are presented in detail. The model performances were evaluated using widely accepted performance metrics such as accuracy, precision, recall, and the area under the ROC curve (AUC). Additionally, training and testing times were recorded to analyse computational efficiency.

The primary distinction of this study from similar works lies in the use of an original, real-time network traffic dataset rather than commonly utilized benchmark datasets found in the literature. This dataset reflects actual network behavior, thereby enhancing the applicability and robustness of the developed model in real-world environments. In particular, missing values observed in the rx_kbps and tot_kbps features were addressed using the mean imputation method, ensuring data integrity and statistical balance. The results presented in Table 7 were obtained using the raw form of the dataset.

Table 7: Performance metrics obtained using the raw form of the dataset

Model	Accuracy	Precision	Recall	AUC	Time (s)
DT	0.964797	0.942821	0.968340	0.988992	0.345564
SVM	0.748818	0.704875	0.610810	0.819793	63.139400
KNN	0.976265	0.972748	0.966125	0.995006	0.015990
LR	0.771595	0.727117	0.661991	0.842565	0.456206

The results presented in Table 7 indicate that when using raw data, the KNN and Decision Tree models stand out with higher accuracy and recall values, whereas SVM and Logistic Regression exhibit lower performance levels. In particular, the long processing time of the SVM model constitutes a disadvantage in terms of computational efficiency.

Table 8: Performance metrics obtained after feature selection

Model	Accuracy	Precision	Recall	AUC	Time (s)
DT	0.999361	0.999180	0.999180	0.999328	0.594741
SVM	0.950549	0.923794	0.951526	0.991254	741.868008

Model	Accuracy	Precision	Recall	AUC	Time (s)
KNN	0.973358	0.968719	0.962680	0.995065	0.0157
LR	0.748658	0.680679	0.668061	0.823800	0.208089

An examination of Table 8 reveals that, based on the findings obtained from the Chi-square test, ANOVA, and correlation matrix analysis, the features byteperflow, packetins, port_no, rx_kbps, and tx_kbps do not exhibit a statistically significant relationship with the target variable. Furthermore, the dt feature, due to its timestamp nature and high correlation, was found to cause information redundancy within the model.

By removing these features from the modelling process, dimensionality reduction was achieved, aiming to develop a simpler, more stable, and better-generalizing model structure. Consequently, utilizing only those features that demonstrate a statistically significant relationship with the target variable not only reduces computational cost but also enhances classification performance. The changes in performance metrics between the raw dataset and the feature-selected dataset are presented in Table 9.

Table 9: Changes in performance metrics before and after feature selection using the raw dataset

Model	Change in Accuracy	Change in Precision	Change in Recall	Change in AUC
SVM	%74.88 → %95.05	%70.49 → %92.38	%61.08 → %95.15	%81.98 → %99.13
DT	%96.48 → %99.94	%94.28 → %99.92	%96.83 → %99.92	%98.89 → %99.93
KNN	%97.63 → %97.33	%97.27 → %96.87	%96.61 → %96.26	%99.50 → %99.51
LR	%77.15 → %74.87	%72.71 → %68.07	%66.19 → %66.80	%84.26 → %82.38

An examination of Table 9 reveals that the impact of feature selection varies across models. Notably, a significant performance improvement was observed in the SVM and Decision Tree models. The accuracy of the SVM model increased from 74.88% to 95.05%, with similar substantial enhancements in precision, recall, and AUC values. This finding indicates that SVM is adversely affected by irrelevant or noisy features and yields substantially stronger results when trained on meaningful attributes. Similarly, the Decision Tree model achieved nearly perfect performance values, demonstrating that it became considerably more stable following feature selection.

The KNN model, on the other hand, exhibited consistently high performance both before and after feature selection, although slight decreases were observed in the latter case. This can be attributed to the fact that KNN is a distance-based algorithm that can benefit from having a larger number of features rather than being hindered by them. In other words, since KNN is capable of utilizing all features in the raw dataset, reducing the number of features led to minor performance losses.

In contrast, the Logistic Regression model experienced a decline in performance metrics. Its accuracy decreased from 77.15% to 74.87%, and precision dropped from 72.71% to 68.07%. This reduction stems from the linear nature of Logistic Regression; the removal

of certain features made it more difficult for the model to establish a linear separation between classes, thereby negatively affecting its performance.

Overall, feature selection substantially enhanced the performance of complex models such as SVM and Decision Tree, while KNN maintained high accuracy with minor decreases, and Logistic Regression experienced a reduction in performance. These findings clearly demonstrate that the effect of feature engineering varies depending on the type of model.

Figure 7 presents the confusion matrix and ROC curve of the Decision Tree model, which achieved the highest performance after the feature engineering process.

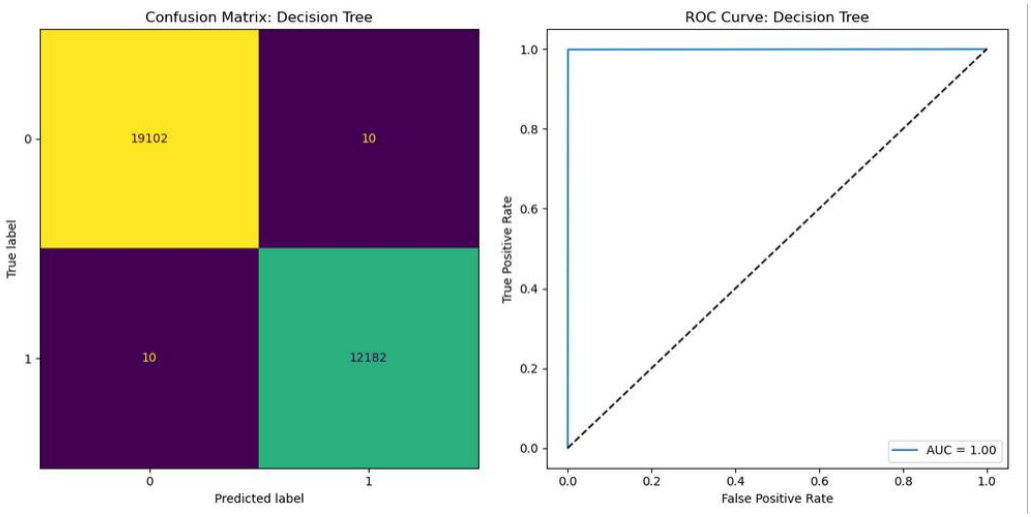

Figure 7. Performance of the Decision Tree after feature engineering

The Decision Tree model demonstrated a significant performance improvement after the application of feature engineering. Initially, the model's accuracy was 96.48%, which increased to 99.94% following feature engineering. This represents an approximate 3.5% improvement, indicating a notable enhancement in the model's overall classification performance.

Parallel improvements were observed in precision and recall. Precision increased from 94.28% to 99.92%, while recall rose from 96.83% to 99.92%. These results indicate that the model not only reduced errors in positive predictions but also substantially improved its ability to correctly identify true positives.

The AUC metric, which provides a more holistic measure of the model's classification power, increased from 98.90% to 99.93%, demonstrating that the model's ability to discriminate between positive and negative classes became nearly perfect.

In terms of computational time, the model required 0.35 seconds before feature engineering and 0.59 seconds afterward. This slight increase reflects the additional computations needed to handle the more complex feature set; however, the runtime remains highly efficient for practical applications.

In summary, feature engineering significantly enhanced the Decision Tree model's classification accuracy and generalization capability, improving its ability to distinguish between positive and negative instances. The modest increase in computation time is a natural consequence of the model's improved performance on a richer and more detailed

feature set. These results clearly demonstrate the effectiveness of feature engineering, particularly for decision tree-based models.

Conclusion

In this study, a novel approach was developed for detecting distributed denial-of-service attacks by integrating machine learning algorithms with feature engineering techniques. The primary goal was to enhance classification performance while maintaining computational efficiency, thereby evaluating the feasibility of applying the proposed method in real-time systems. To achieve this, an original dataset reflecting real-time network traffic was created, and experimental analyses were conducted using various machine learning algorithms.

The findings from these experiments highlight several key outcomes:

- Feature engineering significantly improved model performance. By removing features that did not show a statistically significant relationship with the target variable identified through ANOVA, Chi-square tests, and correlation matrix analysist he model structure became more streamlined and generalizable. In particular, Decision Tree and SVM models demonstrated notable improvements in critical performance metrics such as accuracy, recall, and AUC, emphasizing the crucial impact of proper feature selection on classification success.
- Model performance was assessed not only in terms of accuracy but also computational efficiency. While minor increases in processing time were observed in some models after feature engineering, these increases remained acceptable for practical applications. Algorithms such as KNN and Logistic Regression delivered high accuracy alongside low computational requirements, providing clear advantages for real-time deployment in resource-constrained environments.
- The dataset used in this study differs from commonly used benchmark datasets. Derived from real network traffic, it directly reflects behaviors observed in the field. This ensures that the developed models produce results that are not only theoretically robust but also practically applicable and reliable.

The results demonstrate that the proposed approach can effectively detect DDoS attacks with high accuracy while efficiently utilizing computational resources. Accordingly, the method is suitable for deployment in large-scale network environments and real-time systems.

For future research, the approach should be tested on larger and more complex datasets to further evaluate its robustness. Integrating deep learning architectures may enhance classification performance, while testing the models across diverse network topologies and attack scenarios will help assess their generalizability. Moreover, deploying and validating these models in real-time network environments will be essential to evaluate their adaptability and resilience under dynamic traffic conditions.

Ultimately, this research provides a foundation for developing more dynamic, scalable, and intelligent cybersecurity solutions capable of addressing the rapidly evolving nature of network threats. Researchers are encouraged to extend this framework to create

adaptive detection systems that can operate efficiently in continuously changing network infrastructures.

Finally, testing the models in diverse network environments and on real-time traffic data is critical to fully assess the method's generalizability and resilience. Such efforts will enable the development of more dynamic and scalable solutions to address the rapidly evolving threats in the field of cybersecurity.

References

Aslan, Ö. (2022). A Methodology to Detect Distributed Denial of Service Attacks. *International Journal of InformaticsTechnologies*, *15*(2). https://doi.org/10.17671/gazibtd.1002178

Ayodele, B., & Buttigieg, V. (2024). SDN as a defence mechanism: a comprehensive survey. *International Journal of Information Security*, *23*(1), 141-185. https://doi.org/10.1007/s10207-023-00764-1

Aytaç, T., Aydın, M. A., & Zaim, A. H. (2020). Detection DDOS attacks using machine learning methods. *Electrica*, *20*(2), 159-167. https://doi.org/10.5152/electrica.2020.20049

Catak, F. O., & Mustacoglu, A. F. (2019). Distributed denial of service attack detection using autoencoder and deep neural networks. *Journal of Intelligent & Fuzzy Systems*, *37*(3), 3969-3979. https://doi.org/10.3233/JIFS-190159

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*(1), 21-27. https://doi.org/10.1109/TIT.1967.1053964

Çatak, F. Ö., & Balaban, M. E. (2016). A MapReduce-based distributed SVM algorithm for binary classification. *Turkish Journal of Electrical Engineering and Computer Sciences*, *24*(3), 863-873. https://doi.org/10.3906/elk-1302-68

Çetinkaya, Ş., & Terzi, S. (2024). Analysing the Effects of Cyber Security on National Security from a Realist Perspective:" Stuxnet" Example. *Güvenlik Çalışmaları Dergisi*, *26*(1), 38-51. https://doi.org/10.54627/gcd.1443278

Deepa, V., Sudar, K. M., & Deepalakshmi, P. (2019). Design of ensemble learning methods for DDoS detection in SDN environment. 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN),

Erhan, D., & Anarım, E. (2020). Istatistiksel Yöntemler Ile DDoS Saldırı Tespiti DDoS Detection Using Statistical Methods. 2020 28th Signal Processing and Communications Applications Conference (SIU), Gaziantep, Turkey.

Eşidir, K. A. (2025). Makine Öğrenimi Modelleri ile Yetişkin Eğitimi Analizi: Modellerin Karşılaştırmalı Performansı. *Elektronik Sosyal Bilimler Dergisi*, *24*(2), 946-964. https://doi.org/10.17755/esosder.1589887

Farhana, N., Firdaus, A., Darmawan, M. F., & Ab Razak, M. F. (2023). Evaluation of Boruta algorithm in DDoS detection. *Egyptian Informatics Journal*, *24*(1), 27-42. https://doi.org/10.1016/j.eij.2022.10.005

Garcia-Ramirez, I.-A., Calderon-Mora, A., Mendez-Vazquez, A., Ortega-Cisneros, S., & Reyes-Amezcua, I. (2022). A novel framework for fast feature selection based on multi-stage correlation measures. *Machine Learning and Knowledge Extraction*, *4*(1), 131-149. https://doi.org/10.3390/make4010007

Giuzio, A., Mecca, G., Quintarelli, E., Roveri, M., Santoro, D., & Tanca, L. (2019). INDIANA: An interactive system for assisting database exploration. *Information Systems*, *83*, 40-56. https://doi.org/10.1016/j.is.2019.01.003

Holat, O. (2021). Yeni medya ve siber savaş kavramları bağlamında Stuxnet saldırısı örneğinin incelenmesi. *Abant Kültürel Araştırmalar Dergisi*, *6*(11), 105-121.

Holm, H. (2014). Signature based intrusion detection for zero-day attacks:(not) a closed chapter? 2014 47th Hawaii international conference on system sciences, Waikoloa, HI, USA.

Hossain, M. A. (2025). Deep learning-based intrusion detection for IoT networks: a scalable and efficient approach. *EURASIP Journal on Information Security*, *2025*(1), 28. https://doi.org/10.1186/s13635-025-00202-w

Huang, L.-S., & Chen, J. (2008). Analysis of variance, coefficient of determination and F-test for local polynomial regression. *The Annals of Statistics*. https://doi.org/10.1214/07-AOS531

Ivanova, V., Tashev, T., & Draganov, I. (2022). DDoS attacks classification using SVM. *WSEAS Transactions on Information Science and Applications*, *19*, 1-11. https://doi.org/10.37394/23209.2022.19.1

Karaman, M. S., Turan, M., & Aydın, M. A. (2020). Yapay sinir ağı kullanılarak anomali tabanlı saldırı tespit modeli uygulaması. *Avrupa Bilim ve Teknoloji Dergisi*(Ejosat Ek Özel Sayı (HORA)), 10-17. https://doi.org/10.31590/ejosat.1115825

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14-76. https://doi.org/10.1109/JPROC.2014.2371999

Liu, Y., Zhi, T., Shen, M., Wang, L., Li, Y., & Wan, M. (2022). Software-defined DDoS detection with information entropy analysis and optimized deep learning. *Future Generation Computer Systems*, *129*, 99-114. https://doi.org/10.1016/j.future.2021.11.009

Mahamat, S. B., & Çeken, C. (2019). Anomaly detection in software-defined networking using machine learning. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, *7*(1), 748-756.

Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). 2015 military communications and information systems conference (MilCIS),

Nadeem, M. W., Goh, H. G., Ponnusamy, V., & Aun, Y. (2022). Ddos detection in SDN using machine learning techniques. *Computers, Materials & Continua*, *71*(1). https://doi.org/10.32604/cmc.2022.021669

Özel, Z., & Demirsöz, M. (2021). Makine Öğrenmesi Yöntemleri İle COVID-19 Verilerinin İncelenmesi: Türkiye Örneği. *Sağlık Bilimlerinde Yapay Zeka Dergisi*, *1*(2), 1-7.

Paltacı, B. M. (2022). Ukrayna-rusya savaşı bağlamında siber güvenlik ekosistemi 'nde yaşanan gelişmeler ve değerlendirmeler. *Orta Doğu ve Orta Asya-Kafkaslar Araştırma ve Uygulama Merkezi Dergisi*, *2*(2), 1-19.

Sarhan, M., Layeghy, S., & Portmann, M. (2021). Feature analysis for machine learning-based IoT intrusion detection. *arXiv preprint arXiv:2108.12732*. https://doi.org/10.48550/arXiv.2108.12732

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, *1*(2018), 108-116.

Söğüt, E., & Erdem, O. (2023). SDN Tabanlı SCADA Sistemlerinde Makine Öğrenmesi Tabanlı DDoS Saldırı Tespiti. *Gazi Mühendislik Bilimleri Dergisi*, *1*(1).

Suvra, D. K. (2025). An Efficient Real Time DDoS Detection Model Using Machine Learning Algorithms. *arXiv preprint arXiv:2501.14311*. https://doi.org/10.48550/arXiv.2501.14311

Topbaş, Z. (2024). *Bir Boyutlu Evrişimli Sinir Ağları Kullanılarak ağ Saldırı Tespiti* Necmettin Erbakan University (Turkey)].

Wang, B., Jiang, Y., Liao, Y., & Li, Z. (2024). DDoS-MSCT: A DDoS Attack Detection Method Based on Multiscale Convolution and Transformer. *IET Information Security*, *2024*(1), 1056705. https://doi.org/10.1049/2024/1056705

Wang, Y., & Zhou, C. (2020). Feature selection method based on chi-square test and minimum redundancy. International Conference on Intelligent and Interactive Systems and Applications,

Yağmur, E. (2023). *Scada Sistemlerinde Dagitik Hizmet Disi Birakma Saldirilarinin Derin Ögrenme ve Makine Ögrenmesi Yöntemleri ile Tespiti* Konya Teknik Üniversitesi].

Zahid, M., & Bharati, T. S. (2025). Enhancing cybersecurity in IoT systems: a hybrid deep learning approach for real-time attack detection. *Discover Internet of Things*, *5*(1), 73. https://doi.org/10.1007/s43926-025-00156-y