

A Mini Project report on

Cyber Threat Detection for DOS Attacks

Submitted in partial fulfillment of the requirement for the award of the degree
of Bachelor of Technology in Computer Science and Technology

By

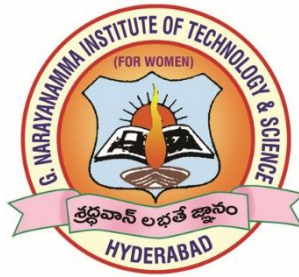
1.G. Nandhini	22251A3610
2. S. Praneetha	22251A3629

Under the guidance of

Mrs. K. Jayasree

Assistant Professor, IT.

Project Batch Number-22CST12



G. Narayanamma Institute of Technology and Science
(AUTONOMOUS) (For Women)

Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad

Accredited by NBA & NAAC

Shaikpet, Hyderabad – 500104, TS.

May 2025



Declaration By Candidate

We hereby affirm, to the best of our knowledge and belief, based on observations, reviews, testing of the project and upon reports submitted by us, this mini project entitled **Cyber Threat Detection for DOS Attacks** is substantially complete and operable. The execution of this project adheres strictly to the guidelines issued by the department.

Name of the candidate Signature

G. Nandhini

S. Praneetha



CERTIFICATE

This is to certify that mini project report entitled **Cyber Threat Detection for DOS Attacks** bonafide work done

By

Name of the candidate

Registration Number

1. G. Nandhini

22251A3610

2. S. Praneetha

22251A3629

under the guidance Mrs. K. Jayasree, Asst. Professor during January 2025 to May 2025, in partial fulfilment for the award of degree in B. Tech in Computer Science and Technology, from G. Narayanamma Institute of Technology & Science (for women).

Mrs. K. Jayasree
Asst. Professor, IT
Internal Guide

Dr.S. Ramcharan
Professor & HOD, IT

ACKNOWLEDGEMENT

The successful completion of our mini project would not be possible without the timely help and guidance rendered by many people. We would like to take this opportunity to thank all of them from the bottom of our heart.

We express our sincere thanks to **Dr. K. Ramesh Reddy**, Principal, G. Narayanamma Institute of Technology and Science for providing us with all the resources and equipment without which this mini project would not be possible.

We express our gratitude to **Dr. S. Ramacharan**, Professor & Head of the Department of Information Technology, G. Narayanamma Institute of Technology and Science, for his support which helped us to carry out the mini project work.

We profoundly indebted to our project coordinators **Dr. V. Sesha Bhargavi**, Asst. Professor, Dept. of IT and **Mr. B. Vijay Kumar**, Asst. Professor, Dept. of IT, for their prompt guidance, support, and oversight, which kept us inspired the entire time. Finally, we extend our heartfelt thanks to our project internal guide, **Mrs. K. Jayasree**, Assistant Professor Dept. of IT, whose guidance steered us in the right direction during our project.

Name of the candidate	Signature
-----------------------	-----------

G. Nandhini	
-------------	--

S. Praneetha	
--------------	--

ABSTRACT

Denial-of-Service (DoS) attacks pose a significant cybersecurity threat by disrupting network services, leading to financial losses and operational downtime. This project leverages machine learning-based classification models to detect DoS attacks, improving upon traditional rule-based and signature-based detection methods. The study compares Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree classifiers on the SDN Dataset to evaluate their accuracy, precision, and recall.

Incorporating insights from the IEEE paper (Online Network DoS/DDoS Detection: Sampling, Change Point Detection, and Machine Learning Methods), our project focuses on model-based classification rather than statistical sampling and change point detection. Unlike the referenced study, which emphasizes real-time anomaly detection techniques, our approach provides a comparative analysis of multiple ML classifiers to determine the most effective model for DoS detection.

The experimental results demonstrate that the Decision Tree classifier achieves the highest accuracy (99.98%), though overfitting concerns are noted. Future enhancements include real-time implementation via web-based applications, optimized feature selection, and deep learning integration. This project contributes to advancing intelligent cybersecurity solutions by utilizing machine learning to enhance DoS threat detection in modern networks.

TABLE OF CONTENTS

	Page No
1. Introduction and Problem Analysis	1-2
1.1 Domain Description	2
1.2 Problem Statement	2
1.3 Scope of the Project	3
1.4 Objectives	3
1.5 Organization of Project Report	3-4
2. Literature Survey	
2.1 Existing Approaches / System	5-6
2.2 Drawbacks in Existing System	6-7
2.3 Summary Table of Literature Review	8-9
2.4 Motivation for Proposed System	9-10
3. Requirement Specification and Design	
3.1 Overall Description of the Project (Use Case Diagrams)	11-12
3.2 Functional & Non-functional Requirements Specification	12-13
3.3 S/w and H/w Requirements	13-14
4. Implementation	
4.1 Methodology (with Flowchart)	15-16
4.2 System Architecture	16-17
4.3 Modules Description with Algorithms / Pseudo-code	18-19
5. Testing	
5.1 Test Cases	20
5.2 Result Analysis / Performance Analysis	21-23
5.3 Conclusion	24
5.4 Future Scope	24
Appendix	
I. Screenshots representing the flow of your project work	25-30

II.	Code (covering only major functionality of the project)	31-44
III.	Bibliography / References	44

LIST OF TABLES

Table 2.1	Summary Table of Literature Review	8-9
Table 5.1	Test Cases	20

LIST OF FIGURES

Fig 1.	Dos Attack Detection	1
Fig 2.	Use Case Diagram	11
Fig 3.	Flow of the Project	15
Fig 4.	System Architecture	17
Fig 5.	Result1	21
Fig 6.	Result2	22
Fig 6.	Result3	23
Fig 8.	Flow1	25
Fig 9.	Flow2	25
Fig 10.	Flow3	26
Fig 11.	Flow4	26
Fig 12.	Flow5	27
Fig 13.	Flow6	27
Fig 14.	Flow7	28
Fig 15.	Flow8	28
Fig 16.	Flow9	29
Fig 17.	Flow10	30
Fig 18.	Flow11	30

1. INTRODUCTION

Denial-of-Service (DoS) attacks are a serious cybersecurity issue that aim to crash or overload online services by flooding them with traffic. This causes major problems for businesses, such as service downtime, financial losses, and damaged reputations. Traditional detection systems that rely on fixed rules or known attack patterns often fail to detect new or evolving threats.

This project uses Machine Learning (ML) to improve the detection of DoS attacks. It compares three popular ML models—Logistic Regression, K-Nearest Neighbours (KNN), and Decision Tree—to see which performs best in detecting malicious network activity. These models are trained using the SDN dataset, which includes real-world traffic data and network parameters.



Fig 1. Dos attack detection

Many recent research papers show that ML can be very effective in cybersecurity, especially in detecting DoS and DDoS attacks. However, some methods rely heavily on deep learning, which needs powerful hardware and large datasets. This project focuses on simpler models that are easier to understand, faster to run, and require fewer resources.

The system is designed to be accurate, easy to use, and scalable. It works on standard computers and uses open-source tools like Python, scikit-learn, and pandas. Among the tested models, the Decision Tree classifier gives the highest accuracy though it may need tuning to avoid overfitting.

With the rapid growth of internet-connected devices and the increasing reliance on cloud services, the threat landscape has expanded significantly. Cybercriminals now have more opportunities to launch DoS attacks against critical infrastructure, e-commerce platforms, financial systems, and public services. As the scale and sophistication of these attacks continue to evolve, traditional signature-based detection methods are proving insufficient. This has created a growing demand for intelligent, adaptive systems that can detect and respond to new attack patterns in real-time. By integrating machine learning into network security strategies, organizations can improve threat detection capabilities, reduce response times, and enhance overall resilience against disruptive attacks.

1.1 Domain and Description of the project

Cybersecurity specifically focusing on Network Security and Intrusion Detection. It involves the use of Machine Learning techniques to detect and prevent Denial-of-Service (DoS) attacks by analysing network traffic patterns and identifying malicious activity in real-time or near real-time.

1.2 Problem Statement

Denial-of-Service attacks flood networks with excessive traffic, disrupting services and making traditional signature-based detection methods ineffective, especially against evolving threats. While Machine Learning offers improved detection.

This project aims to solve these challenges by evaluating three ML classifiers—Logistic Regression, K-Nearest Neighbors, and Decision Tree using the SDN dataset. The objective is to develop an accurate, f1 score, recall, and precision DoS detection system with fast response time, low false positives, and the flexibility to adapt to new attack patterns.

1.3 Scope of the Project

This project focuses on detecting DoS attacks using Machine Learning classifiers (Logistic Regression, KNN, and Decision Tree). It analyzes network traffic data from the SDN dataset to classify normal and malicious traffic. The system aims to improve detection accuracy, precision, f1 score recall, and enhance graph adaptability to evolving attacks. Future enhancements include real-time monitoring, deep learning integration, and IDS implementation

1.4 Objectives

- Develop a ML-based DDoS detection system using Logistic Regression, KNN, and Decision Tree classifiers trained on the SDN dataset to distinguish attack scenarios.
- Compare ML algorithms by evaluating parameters to identify the best classifier for DoS detection while addressing high false positives and low detection rates.

1.5 Organization of Project Report

The project begins by introducing the concept of Denial-of-Service (DoS) attacks and explaining why they are a major threat to network systems. It highlights the weaknesses of traditional detection methods, which often fail to identify new or evolving attack patterns. To overcome these issues, the project explores how machine learning can be used for more accurate and efficient detection. A detailed background study is conducted through a literature survey, reviewing recent IEEE research papers that use various ML and deep learning models for DoS detection. These papers help identify gaps in current approaches, such as high computational requirements and low interpretability, especially in real-time systems.

Next, the project clearly defines the problem it aims to solve and explains the need for a lightweight and scalable detection system. It proposes a solution using three ML algorithms—Logistic Regression, K-Nearest Neighbours, and Decision Tree to analyze network traffic data from the SDN dataset. The project ensuring that the system is not only technically sound but also user-friendly, reliable, and adaptable for future improvements. The implementation phase involves collecting and preprocessing the SDN dataset, training the ML models, and evaluating their performance based on

accuracy, precision, and f1score recall. Among the models, the Decision Tree performs best, achieving up to accuracy, though care must be taken to prevent overfitting. The results are analyzed and compared to show which model is most suitable for practical deployment. Finally, the project concludes by discussing how this ML-based system can help detect DoS attacks more effectively and suggests future enhancements like real-time monitoring, integration with deep learning, and improved interfaces for easier use in cybersecurity environments.

The project is organized into several well-defined sections to ensure clarity and logical progression. It begins with an Introduction, which outlines the motivation behind the work and the growing threat of DoS attacks. This is followed by a Literature Review, where relevant research studies are analyzed to establish the current state of the field and identify existing challenges. The Problem Definition and Objectives section clearly states the issues with traditional methods and explains the need for a lightweight ML-based solution. The Methodology chapter details the dataset used (SDN dataset), the preprocessing steps, feature selection, and the implementation of the three chosen ML algorithms—Logistic Regression, K-Nearest Neighbours (KNN), and Decision Tree. The Results and Evaluation section compares the models based on key metrics such as accuracy, precision, recall, and F1-score, and discusses the implications of the findings. Finally, the Conclusion and Future Work section summarizes the contributions of the project, reflects on its limitations, and proposes directions for future enhancements such as real-time integration and adoption of hybrid ML/deep learning models for improved robustness.

2. LITERATURE SURVEY

2.1 Existing Approaches /System

Traditional signature-based and rule-based DoS detection methods are ineffective against new and evolving attack patterns. Some ML and deep learning approaches proposed in IEEE studies enhance detection but require high computational power, large datasets, and complex implementation. These limitations make them less practical for real-time cybersecurity applications.

Paper 1 – Performance Evaluation of Deep Learning Techniques for DoS Attacks in WSNs(2023)

This study tested CNN, RNN, and LSTM models on WSN data to detect DoS attacks like blackhole and flooding. A lightweight deep learning model achieved 96.7% accuracy, balancing performance with efficiency for resource-limited environments. Key features like packet loss rate and inter-arrival time proved crucial. While it focused on WSNs, the idea of lightweight, accurate models supports our project's approach for efficient deployment on platforms like Google Colab.

Paper 2 – Extended Evaluation of ML Techniques for DoS Detection in WSNs (2023)

This paper compared models like REPTree, Random Forest, and K-Means. REPTree stood out with a 95.69% F1-score and fast performance (~0.93 seconds). It emphasized the importance of parameter tuning and highlighted how unsupervised models, while flexible, fall short in accuracy. Though based on WSNs, the strong results from Decision Tree variants align with our project's use of similar models for scalable and quick detection.

Paper3 – DoS Detection with Lightweight ML in WSNs (2024)

Using a Decision Tree with Gini-based feature selection, this study reached a 99.86% true positive rate and 0.05% false positive rate—extremely strong results for WSN scenarios. It focused on selecting the most relevant features to cut down training time without losing accuracy. This approach mirrors our project's goals: keep the model simple, fast, and effective—especially on platforms with limited processing power.

Paper 4 – *Deep Learning for DoS Detection in SDNs (2023)*

This is one of the few studies focusing specifically on Software-Defined Networks (SDNs). It used RNN, LSTM, and GRU models on the InSDN dataset to protect the SDN controller. GRU performed best, with 97.8% accuracy and low inference time (0.12s). The paper emphasized SDN-specific issues like controller overload. While deep learning models are powerful, they are computationally heavier—less practical for real-time detection on limited hardware, reinforcing our choice of simpler ML models.

Paper 5 – *ML-Based DoS Detection in WSNs (2023)*

This paper compared Decision Trees and SVMs, with DT achieving a 99.86% TPR and near-instant training (0.8s). Important features included packet transmission rate and latency. While it used a WSN setting, the focus on efficiency and lightweight models makes it very relevant to our goal of using Decision Trees for fast, accurate DoS detection in an SDN environment on Colab.

2.2 Drawbacks of existing system

- **High Computational Load:** Deep learning models like CNNs, RNNs, and LSTMs require GPUs or high-performance CPUs, which are impractical for environments like IoT, WSNs, or lightweight cloud platforms (e.g., Google Colab with limited free resources). Large Labeled Data Requirement: Supervised ML methods need extensive labeled datasets, which are hard to obtain and maintain.
- **Complex Implementation and Tuning:** Advanced models involve intricate architectures, hyperparameter tuning, and large dependency libraries, making them harder to deploy, maintain, and interpret—especially in fast-paced security environments.

- **Limited Real-Time Applicability:** Long training or inference times restrict the ability to detect and respond to attacks instantly. In networks like SDN where controllers must act quickly, delays can lead to significant damage.
- **Class Imbalance in Datasets :** Most datasets contain significantly more normal traffic than attack traffic, leading to biased models. This imbalance causes under-detection of attacks, especially when rare attacks are present.
- **Vulnerability to Adversarial Attacks:** Many ML-based systems can be fooled by carefully crafted adversarial inputs, causing them to misclassify malicious traffic as benign. Attackers can exploit this vulnerability to bypass detection.
- **Scalability Challenges:** Systems that perform well on small test datasets may fail under real-world loads due to poor scalability. Increased traffic volume, multiple data sources, or distributed systems may overwhelm such models.
- **Limited Adaptability to Evolving Threats:** Static models are not equipped to adapt to newly emerging or mutating attack types without frequent retraining, which is resource-intensive and slow.
- **Insufficient Dataset Standardization:** Existing datasets often lack uniformity in feature representation, making it difficult to compare methods or reproduce results. This limits benchmarking and real-world applicability
- **Large Labeled Data Requirement:** Supervised ML models depend on large, well-labeled datasets. However, collecting labeled DoS attack data is costly, time-consuming, and often unrealistic—especially for evolving or zero-day attacks.

2.3 Summary table of Literature Survey

S.No	Title of the Publication	Journal/Conference Name with Month and Year of Publication	Methodology Used	Key Findings	Limitations/Gaps Identified
1	Detection of DoS Attack in WSNs: A Lightweight ML Approach	IEEE Xplore, 2024	DT with Gini selection	99.86% true positive rate	WSN-specific, not SDN-tailored
2	Extended Evaluation on ML Techniques for DoS Detection in WSNs	Engineering Applications of Artificial Intelligence, 2023	REPTree, RF, K-Means	95.69% F1-score for blackhole attacks	WSN-focused, non-SDN dataset
3	Performance Evaluation of DL Techniques for DoS Detection in WSNs	Journal of Big Data, February 2023	CNN, RNN, LSTM (lightweight)	96.7% accuracy on WSN-DS	WSN-focused, high computational cost
4	Deep Learning Algorithms for Detecting DoS Attacks in SDNs	ScienceDirect, 2023	RNN, LSTM, GRU	97.8% accuracy on InSDN	Resource-intensive, complex
5	Using ML to Detect DoS Attacks in WSNs	IEEE Xplore, 2023	DT, SVM	99.86% true positive (DT)	WSN-focused, not SDN-specific

Table 1. Literature survey

Recent studies on Denial-of-Service (DoS) attack detection using machine learning (ML) and deep learning (DL) techniques have produced encouraging outcomes across different network architectures, particularly Wireless Sensor Networks (WSNs) and Software-Defined Networks (SDNs). In WSNs, research emphasizes the use of lightweight and interpretable models such as Decision Trees, REPTree, and Random

Forest, which offer high detection accuracy—sometimes exceeding 99%—with minimal computational cost, making them suitable for resource-constrained environments. Features like packet inter-arrival time, packet loss rate, and reduced feature sets enhance their real-time detection capabilities. Additionally, unsupervised methods such as K-Means clustering have been explored to address data labeling challenges, though with generally lower performance. However, these approaches are narrowly tailored to WSN-specific characteristics and fail to address the architectural and threat model differences inherent in SDNs, such as centralized control, dynamic flow management, and vulnerability to controller-targeted attacks like flow table saturation or controller bottlenecks. On the other hand, SDN-specific studies deploy deep learning models like RNN, LSTM, and GRU, leveraging their ability to analyze sequential traffic patterns and achieving impressive accuracy (up to 97.8%). These models demonstrate strong potential for capturing complex temporal patterns in network traffic. Nevertheless, they are resource-intensive and pose challenges for real-time or edge deployment due to their computational complexity and memory requirements, particularly in constrained environments such as IoT-integrated SDN or cloud-based simulations. Moreover, many studies are limited by the use of isolated datasets such as WSN-DS or InSDN, which may not fully reflect real-world variability or evolving attack strategies. This contrast between lightweight efficiency and high-performing complexity underscores a critical research gap: the absence of adaptive, SDN-aware ML/DL models that can offer a balanced trade-off between detection accuracy, scalability, computational overhead, and practical deployment, especially in dynamic or hybrid network environments. Future research should focus on creating generalized frameworks that integrate lightweight architectures with domain adaptation, online learning, or federated learning to enhance their robustness and adaptability in diverse, real-time settings.

2.4 Motivation of proposed System

Traditional signature-based and rule-based detection methods struggle to detect new and evolving DoS attacks, often leading to high false positives and low detection rates. Existing supervised learning models require large labelled datasets, while unsupervised methods have low accuracy.

To overcome these limitations, our Machine Learning-based approach uses the SDN dataset with network parameters to evaluate and compare Logistic Regression, KNN, and Decision Tree classifiers. Unlike existing systems, our approach:

- Achieves higher accuracy, precision, and recall, ensuring better attack detection.
- Compares multiple ML models, allowing selection of the best-suited algorithm.
- Works with any dataset by adjusting preprocessing steps, making it adaptable to evolving threats.

This makes our system more reliable, flexible, and efficient than traditional detection methods.

3. REQUIREMENT SPECIFICATION AND DESIGN

3.1 Overall description of the project (Use case Diagrams)

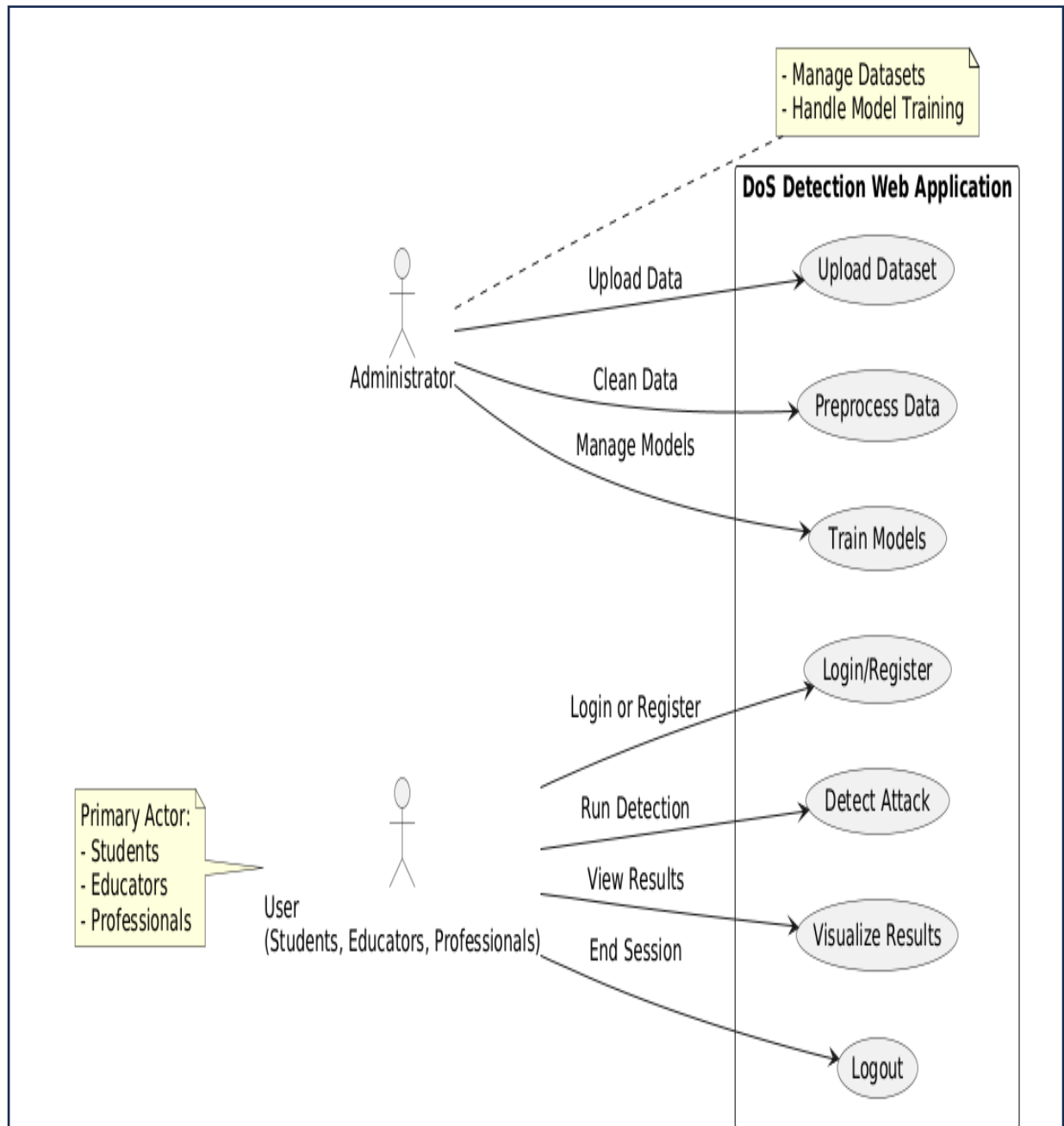


Fig 2. Use case diagram

The image is a Use Case Diagram for a Cyber Threat Detection System, outlining the interactions between users and the system. Here's a breakdown for your report:

1. **User:** Interacts with the system to perform detection and view results.
2. **Administrator:** A role for managing datasets and models.

Administrator Use Cases:

- Train Machine Learning Models – The administrator trains models for detecting cyber threats.
- Upload Dataset – Responsible for uploading datasets used for model training or threat detection.
- Preprocess Data – Ensures data is cleaned and formatted appropriately before analysis.

User Use Cases:

- Visualize Results Views processed data and detection outputs in a user friendly format
- Detect Attack – Engages the system to identify potential cyber threats.
- Generate Alerts and Reports – Automatically or manually generates alerts and reports based on threat detection outcomes.

Purpose:

To visually represent the roles and responsibilities of system users (Administrator and User) and to detail the key functionalities of the system for cyber threat detection and response.

3.2 Functional and Non-functional Requirement Specification

Functional system Requirements:

1. **Data Collection & Preprocessing:** The system must collect network traffic data from the SDN dataset and preprocess it for feature selection and normalization.
2. **Machine Learning Model Training:** The system should support training using multiple classification algorithms (Logistic Regression, K-Nearest Neighbours, and Decision Tree) to identify DoS attacks.

3. **Attack Detection & Classification:** The trained model must be able to classify incoming network traffic as either legitimate or malicious (DoS attack).
4. **User Interface for Monitoring:** A basic user interface should allow security analysts to view real-time predictions, logs, and reports on detected DoS attacks.
5. **Model Optimization:** The system should allow for future enhancements, such as feature selection improvements and hyperparameter tuning, to reduce overfitting.

Non-Functional Requirements:

1. **Performance Efficiency:** The system must process large volumes of network traffic data efficiently, ensuring real-time or near-real-time detection.
2. **Scalability:** The system should be capable of handling increasing traffic loads and adapting to new attack patterns.
3. **Reliability:** The detection model should consistently provide accurate results with minimal false positives and false negatives.
4. **Security:** The system must ensure data integrity and protect against adversarial attacks that attempt to manipulate detection results.
5. **Usability:** The interface must be user-friendly, allowing security analysts with basic ML knowledge to operate it effectively.
6. **Maintainability:** The system should be easy to update, allowing for new algorithms, datasets, or security measures to be incorporated.
7. **Cost-effectiveness:** The solution should utilize open-source tools and require minimal hardware investment for practical deployment.

3.3 Software and hardware components

Software Requirements

- Windows XP or later
- Google Collab
- Python

Hardware Requirements

- 2GB RAM
- 2.0+Ghz Processor
- 2TB Hard disk

4.IMPLEMENTATION

4.1 Methodology Flow chart

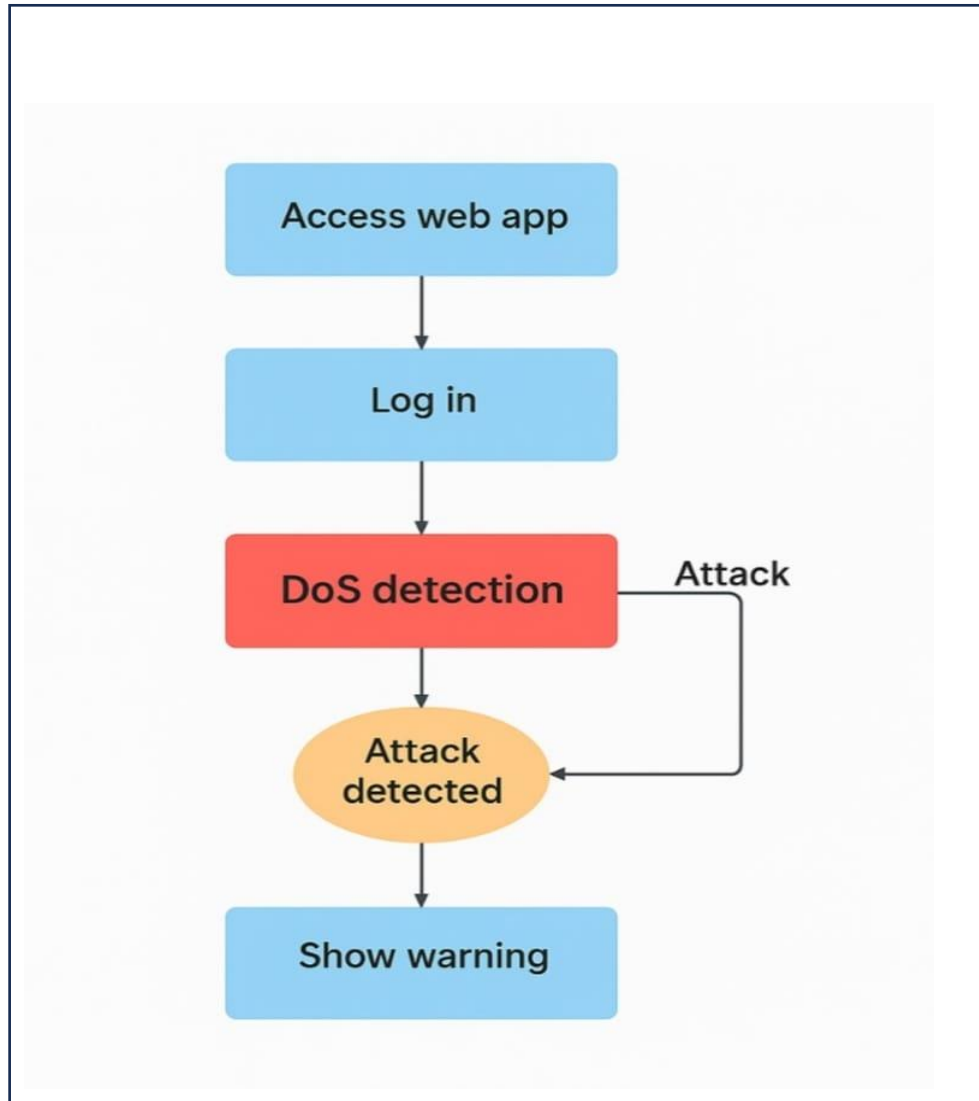


Fig 3. Flow chart

The diagram illustrates a flowchart for detecting and handling a Denial-of-Service (DoS) attack within a web application. The process begins when a user accesses the web application and proceeds to log in. Once logged in, the system initiates a DoS detection mechanism to monitor for any suspicious activities indicative of an attack. If a DoS attack is detected, the system triggers an alert, indicating that an attack has been identified. In response, a warning message is displayed to the user or administrator to notify them of the potential threat. The flowchart also highlights a feedback loop

between the "DoS detection" and "Attack detected" stages, emphasizing continuous monitoring and real-time response to ongoing threats. This continuous vigilance is crucial in managing and mitigating the impact of potential DoS attacks on the application.

1.Logistic Regression Module: Logistic Regression is used for binary classification to predict whether network traffic is normal or an attack based on input features. It uses a sigmoid activation function to output probability scores.

2.K-Nearest Neighbors Module: KNN classifies a data point by finding the 'K' closest points in the feature space and assigning the most common class among them. It is a distance-based learning algorithm.

3.Decision Tree Module: Decision Tree builds a hierarchical tree structure where each internal node represents a decision based on a feature value. It splits the data recursively until it classifies the input traffic. It works well for handling non-linear data patterns.

4.2 System Architecture

The system architecture diagram of the Cyber Threat Detection System represents a layered design that efficiently organizes the flow of data from input to actionable output. It begins with the Input Layer, which collects raw data from an SDN dataset. This data is passed into the Processing Layer, the core of the system, which contains three key modules: the Data Preprocessing Module prepares and cleans the data, the ML Model Training Module builds predictive models using machine learning algorithms, and the Attack Detection Module uses those models to identify malicious activities. Finally, the Output Layer delivers the results by generating Reports for analysis and Alerts to notify about detected threats. This architecture ensures a systematic, modular approach to detecting cyber threats in a software-defined networking environment.

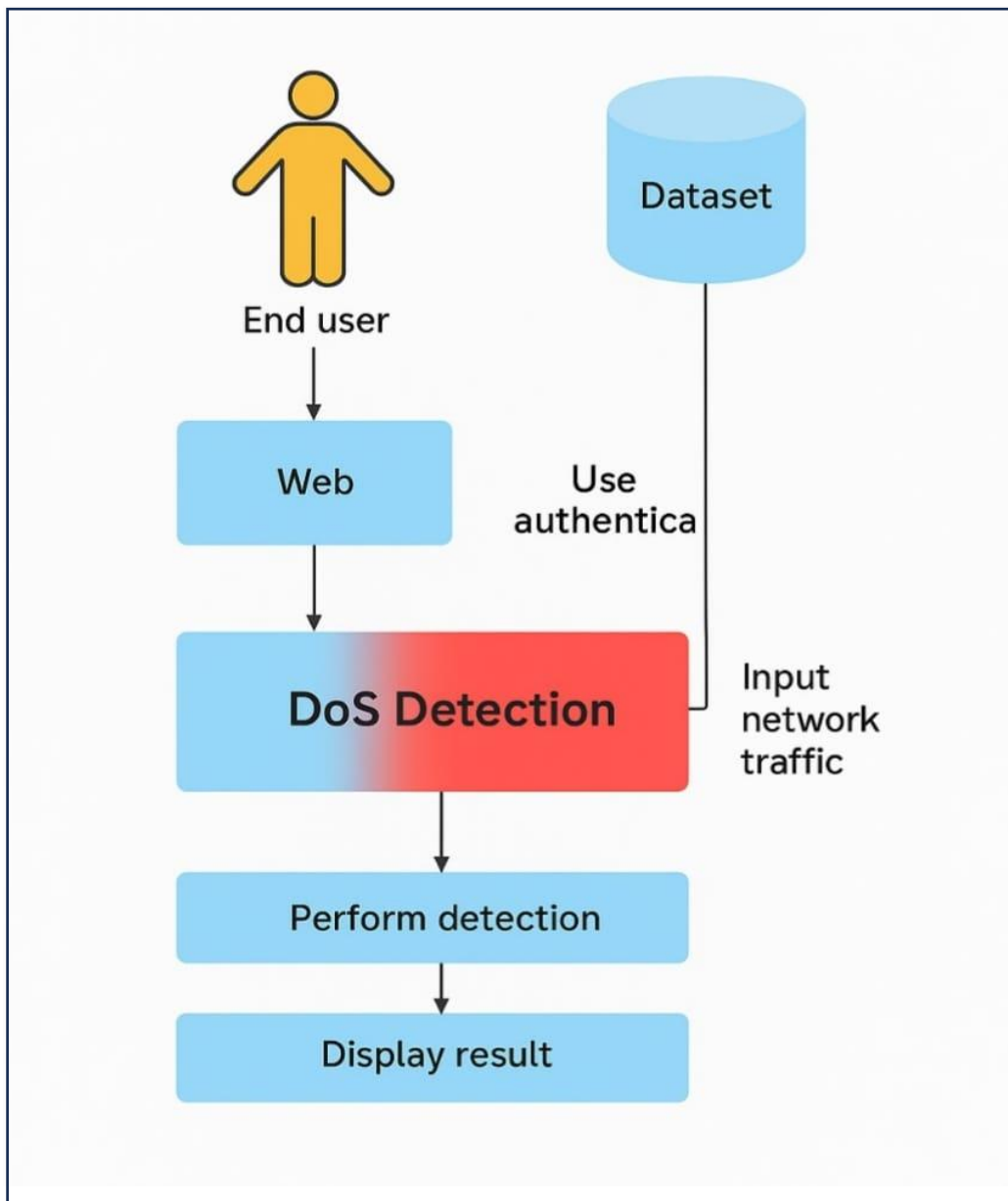


Fig 4. System architecture

The diagram represents a system architecture for detecting Denial-of-Service (DoS) attacks. It begins with an end user accessing the web through a user interface. The system then utilizes authentication mechanisms to verify the user before proceeding to the DoS Detection stage. The detection process uses input from network traffic data and a dataset that likely contains known attack patterns or historical data for comparison.

The detection process itself is conducted by analysing the network traffic to identify abnormal patterns or potential DoS attacks. Once the detection process is complete, the system proceeds to display the result, providing feedback on whether an attack has been

identified. This architecture highlights the integration of data-driven detection with real-time traffic monitoring to safeguard web applications from DoS threats.

4.3 Modules Description

1.Data Preprocessing Module: The Data Preprocessing Module plays a vital role in preparing the SDN dataset for machine learning by ensuring the raw input is clean, consistent, and suitable for training. This involves cleaning the data by removing duplicates, handling missing values, and filtering out noise or irrelevant features. The data is then scaled or normalized so that all features operate on a similar range, which is crucial for many machine learning algorithms to perform optimally. Additionally, categorical features are encoded into numerical form using techniques such as label encoding or one-hot encoding. Another important step is feature selection or extraction, where the most relevant features for detecting threats are identified to improve model accuracy and efficiency. Finally, the data is split into training and testing sets to allow for proper evaluation of model performance and to prevent overfitting.

2.Evaluation Module: The Evaluation Module is responsible for assessing the effectiveness of the trained machine learning model using various performance metrics. This module ensures that the model is not just trained but also capable of accurately identifying cyber threats in unseen data. Key evaluation metrics include accuracy, which measures the overall correctness of the model; precision, which indicates the proportion of true positives among all predicted positives; and recall, which shows the model's ability to detect all actual threats. The F1-score combines precision and recall into a single measure, especially useful when dealing with imbalanced datasets. Additionally, a confusion matrix is often generated to provide a visual summary of correct and incorrect predictions across different classes. This comprehensive evaluation helps in refining the model and ensuring its reliability in real-world deployment.

Pseudo code

Start

Load SDN dataset from URL

Cyber Threat Detection for DOS Attack

Clean missing data

Convert categorical features to numeric

Drop unused columns

Split data into features (X) and labels (y)

Scale features

Split X and y into training and testing sets

Initialize ML models: Logistic Regression, SGD, Decision Tree, KNN

For each model:

Train model on training set

Predict on test set

Calculate accuracy and performance metrics

Select the best performing model

Display:

- Accuracy scores
- Confusion matrix
- Classification report
- Top DoS source IPs
- Feature correlation

Provide a login interface

If login is valid:

Show detection interface

Else:

Show error or registration option

End

5.TESTS

5.1Test Cases

Test Case ID	Test Case Name/Objective	Prerequisite/Pre-condition	Test Description/Testing Process	Expected Result	Actual Result	Pass/Fail/Not executed/suspended	Action/Notes
TC01	Valid Login	Users dictionary initialized with admin:password123	Enter username admin, password password123. Click Login.	Redirect to dashboard.	Redirected to dashboard.	Pass	None
TC02	Invalid Login	Users dictionary initialized	Enter username admin, password wrongpass. Click Login.	Error: "Invalid username or password".	Error displayed.	Pass	Validates error handling.
TC03	New User Registration	Users dictionary initialized	Enter new username testuser, password test123. Click Create Account.	Success: "Account created! Please log in."	Success message shown.	Pass	None
TC04	Duplicate Registration	Users dictionary contains testuser	Enter username testuser, password test123. Click Create Account.	Error: "Username already exists".	Error displayed.	Pass	Ensures unique usernames.
TC05	Detection Functionality	Logged in, dataset URL accessible	Click Run Detection button.	Display dataset stats, IPs, correlations, metrics, and bar plot.	All outputs displayed.	Pass	None
TC06	Dataset Load Failure	Dataset URL inaccessible	Modify URL to invalid link. Click Run Detection.	Error: "Error loading dataset".	Error displayed.	Pass	Simulates network failure.
TC07	Empty Input on Login	Users dictionary initialized	Leave username and password fields empty. Click Login.	Error: "Invalid username or password".	Error displayed.	Fail	Expected error message shown, but system should prompt for non-empty input explicitly.
TC08	Logout Functionality	Logged in	Click Logout button.	Redirect to login page.	Redirected to login page.	Pass	None

Table 2. Test cases

RESULTS AND CONCLUSION

6.1 Result Analysis / Performance Analysis

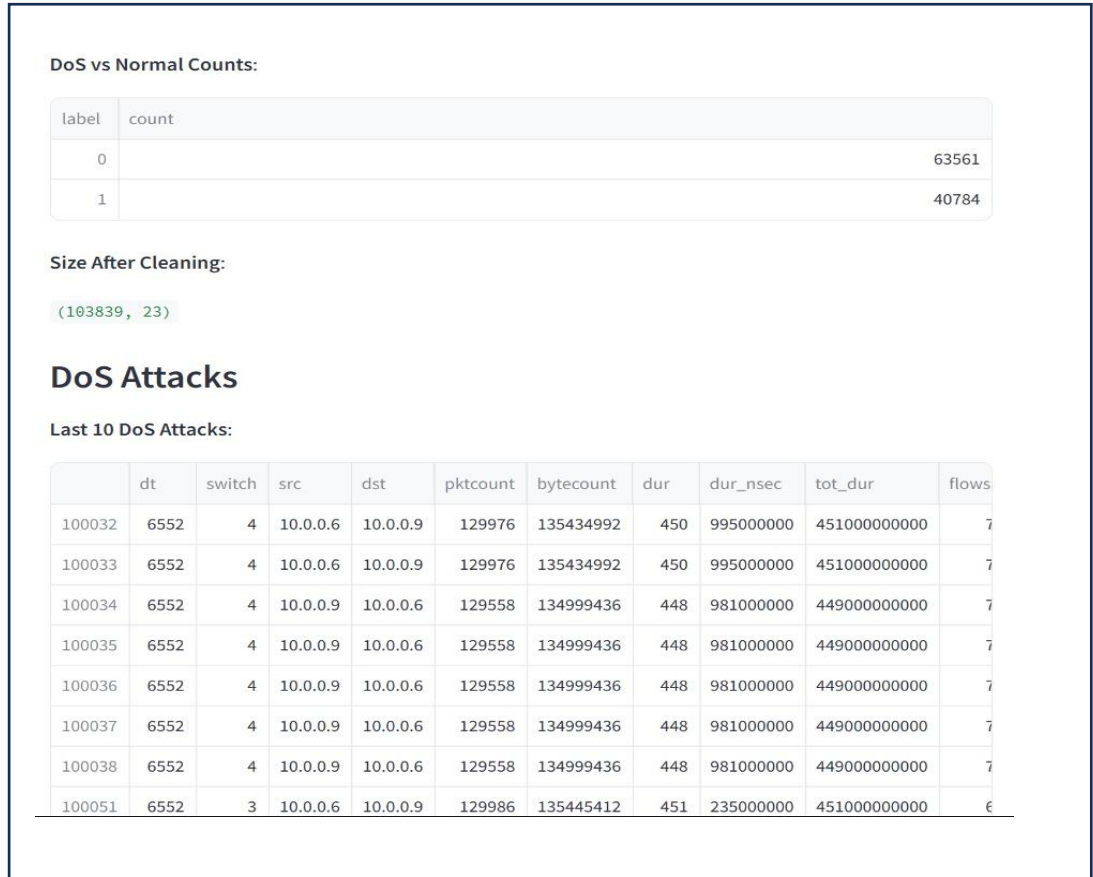


Fig 5. Result1

1. Data Analysis:

- The first image shows the distribution of DoS and normal traffic after data cleaning.
- There are 63,561 normal instances (label 0) and 40,784 DoS instances (label 1).
- The cleaned dataset contains 103,839 rows and 23 features, indicating extensive preprocessing to remove noise and irrelevant data.

2. Recent DoS Attack Records:

- The table highlights the last 10 detected DoS attacks, displaying features such as source, destination, packet count, byte count, duration, and flow metrics.

- It provides a snapshot of ongoing or recent attacks, showing uniformity in IP addresses and data transfer metrics, which can indicate a repetitive attack pattern.

Feature Correlations (>0.60)

	dt	switch	pktcount	bytecount	dur	dur_nsec	tot_dur	flows	packetins	pktperflow
switch	None	1	None	None	None	None	None	None	None	None
pktcount	None	None	1	0.6758	None	None	None	None	None	None
bytecount	None	None	0.6758	1	None	None	None	None	None	None
dur	None	None	None	None	1	None	1	None	None	None
dur_nsec	None	None	None	None	None	1	None	None	None	None
tot_dur	None	None	None	None	1	None	1	None	None	None
flows	None	None	None	None	None	None	None	1	None	None
packetins	None	None	None	None	None	None	None	None	1	None
pktperflow	None	None	None	None	None	None	None	None	None	1
byteperflow	None	None	None	None	None	None	None	None	None	0.8

Fig 6. Result2

3. Feature Correlation:

- The second image presents a correlation matrix for features with a correlation coefficient above 0.60.
- Key correlations:
 - pktcount and bytecount (0.6758) - High correlation, suggesting that an increase in packet count often leads to a similar increase in byte count.
 - dur and tot_dur (1) - Duration and total duration are perfectly correlated.
 - pktperflow and byteperflow (0.8) - Indicates that flows with higher packet counts generally also have higher byte counts.

- These correlations help in feature selection, optimizing the detection model by focusing on strongly related attributes.

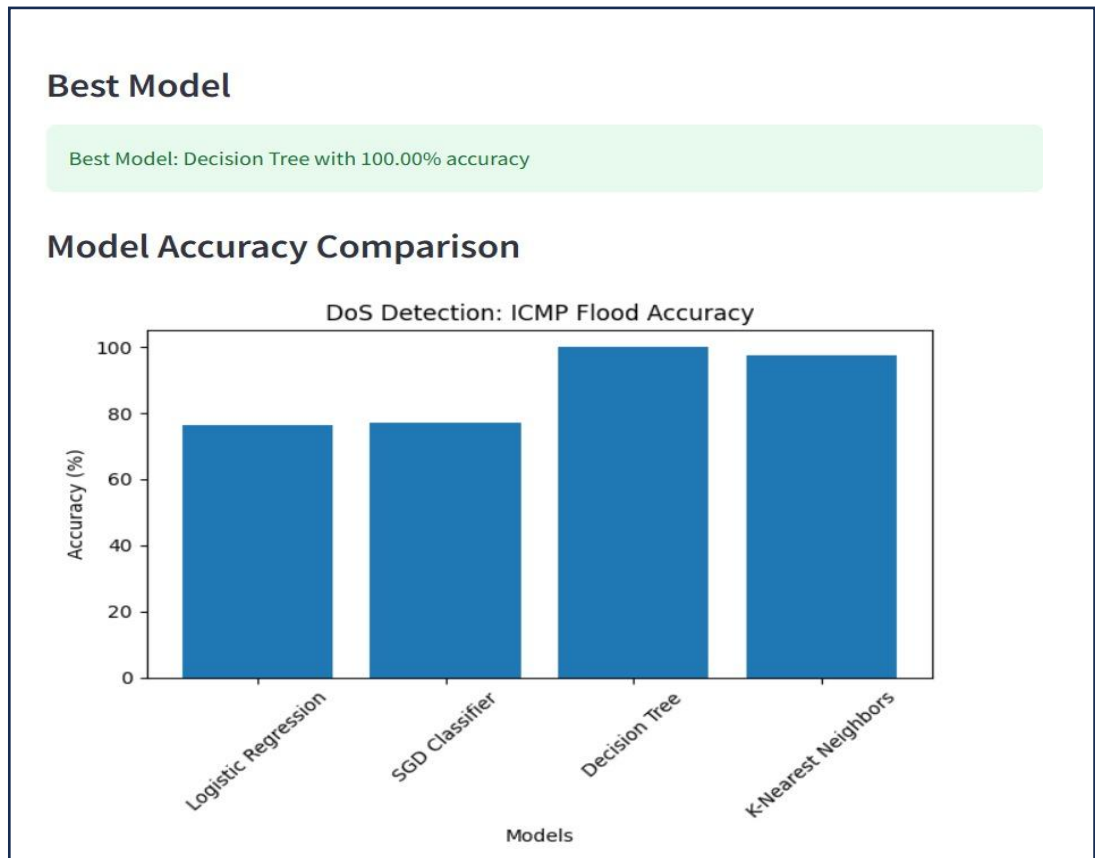


Fig 7. Result3

4. Model Accuracy Comparison:

- The third image highlights a model comparison chart for DoS detection accuracy, specifically focusing on ICMP flood attacks.
- The Decision Tree model achieved the highest accuracy (100%), followed closely by K-Nearest Neighbors (KNN).
- Other models like Logistic Regression and SGD Classifier show lower accuracy, suggesting that tree-based or instance-based methods are more effective for this dataset.
- The Decision Tree model is chosen as the best model due to its perfect accuracy, making it ideal for precise DoS attack detection.

6.2 Conclusion


This project demonstrates that Machine Learning model specifically Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree classifier can effectively detect Denial-of-Service (DoS) attacks by analyzing network traffic data from the SDN dataset. By focusing on lightweight and interpretable models, the system achieves high detection accuracy while remaining fast, resource-efficient, and scalable for real-world use. Among the tested models, the Decision Tree classifier outperforms others, although care must be taken to prevent overfitting. The project successfully highlights the limitations of traditional rule-based detection systems and shows how ML techniques can adapt to evolving attack patterns. Additionally, it sets a strong foundation for future enhancements such as real-time monitoring, deep learning integration, deployment as a full IDS, and the development of user-friendly dashboards, ultimately contributing to more robust and practical cybersecurity solutions.

6.3 Future Scope

In the future, the project can be enhanced by integrating real-time monitoring and exploring deep learning techniques like CNNs and RNNs for improved detection. Expanding capabilities to detect Distributed Denial-of-Service (DDoS) attacks and applying advanced feature engineering can further boost accuracy. Optimization through ensemble methods, scalability for large networks, and cross-dataset validation will improve reliability. Incorporating Explainable AI (XAI) will enhance model transparency, while deploying the system as a full Intrusion Detection System (IDS) will offer automated responses. Finally, creating user-friendly interfaces and dashboards will make the system accessible to both technical and non-technical users.

APPENDIX

i. Screenshots flow of project work



Login to DoS Detection :

About This Project

This project detects ICMP flood Denial-of-Service (DoS) attacks using machine learning. It analyzes network traffic with four models (Logistic Regression, SGD Classifier, Decision Tree, KNN) and displays results like DoS IPs, correlations, and model accuracy

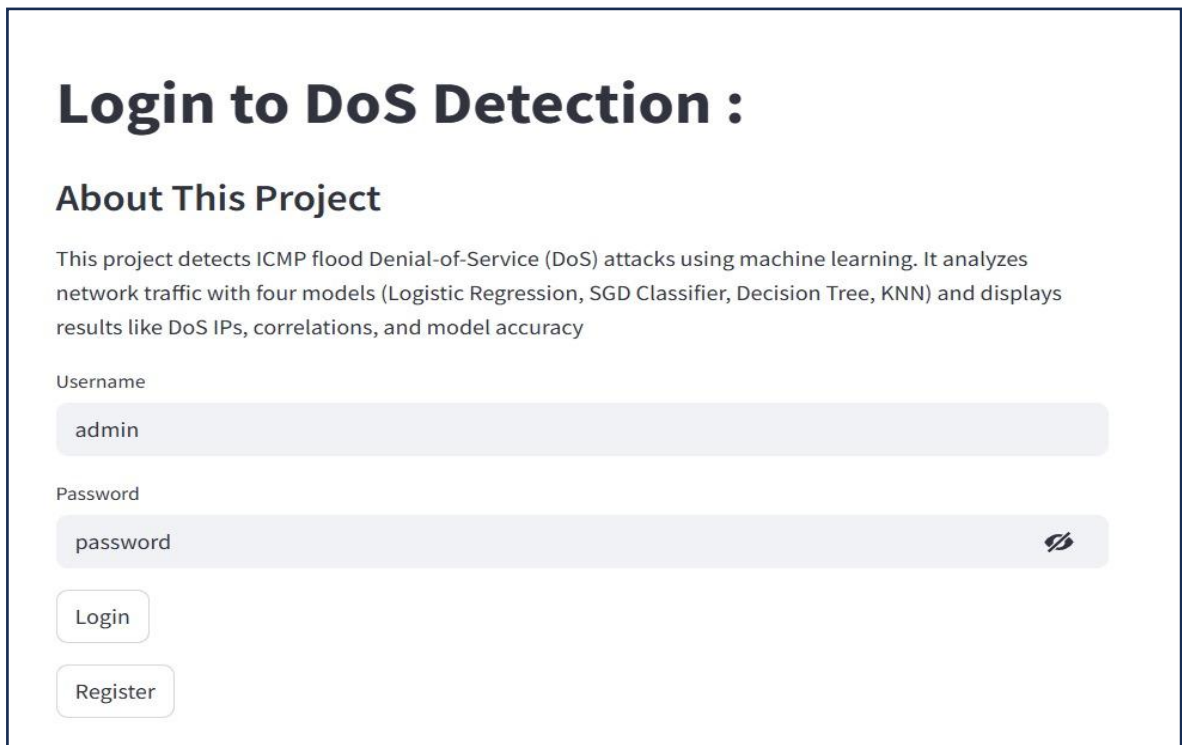
Username

Password

Login

Register

Fig 8. Flow1



Login to DoS Detection :

About This Project

This project detects ICMP flood Denial-of-Service (DoS) attacks using machine learning. It analyzes network traffic with four models (Logistic Regression, SGD Classifier, Decision Tree, KNN) and displays results like DoS IPs, correlations, and model accuracy

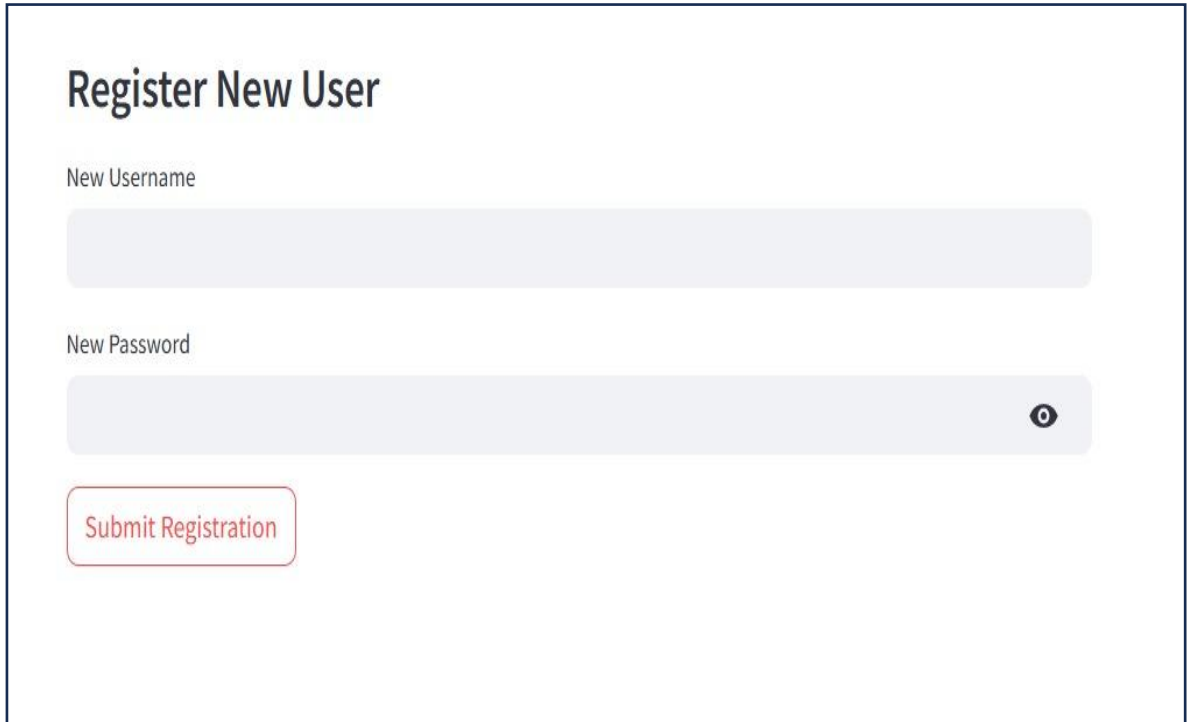
Username

Password

Login

Register

Fig 9. Flow2



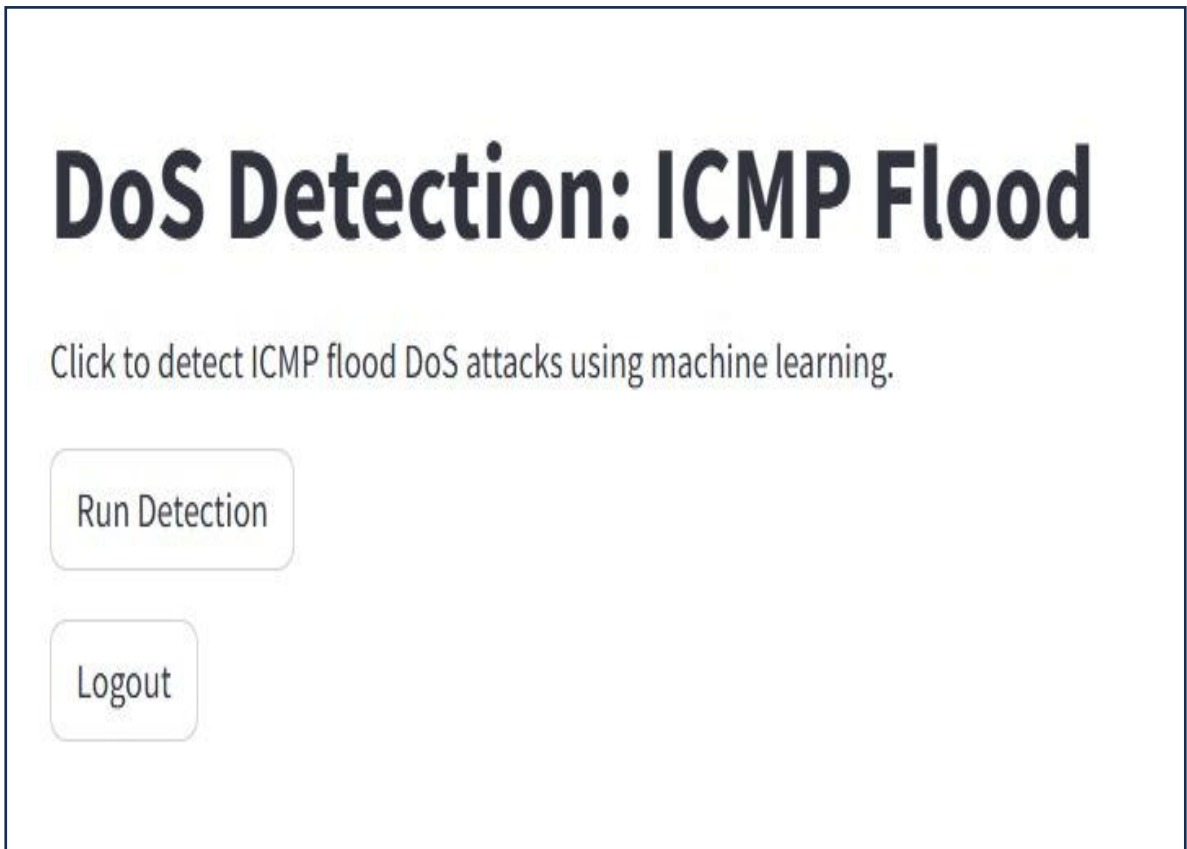
Register New User

New Username

New Password

[Submit Registration](#)

Fig 10. Flow3



DoS Detection: ICMP Flood

Click to detect ICMP flood DoS attacks using machine learning.

[Run Detection](#)

[Logout](#)

Fig 11. Flow4

DoS Detection: ICMP Flood

Click to detect ICMP flood DoS attacks using machine learning.

Run Detection

Dataset Information

Data Size:

(104345, 23)

Missing Data:

	0
dt	0
switch	0
src	0
dst	0
pktpcount	0
bytecount	0
dur	0
dur_nsec	0
tot_dur	0

Fig 12. Flow5

DoS vs Normal Counts:

label	count
0	63561
1	40784

Size After Cleaning:

(103839, 23)

DoS Attacks

Last 10 DoS Attacks:

	dt	switch	src	dst	pktpcount	bytecount	dur	dur_nsec	tot_dur	flows
100032	6552	4	10.0.0.6	10.0.0.9	129976	135434992	450	995000000	451000000000	7
100033	6552	4	10.0.0.6	10.0.0.9	129976	135434992	450	995000000	451000000000	7
100034	6552	4	10.0.0.9	10.0.0.6	129558	134999436	448	981000000	449000000000	7
100035	6552	4	10.0.0.9	10.0.0.6	129558	134999436	448	981000000	449000000000	7
100036	6552	4	10.0.0.9	10.0.0.6	129558	134999436	448	981000000	449000000000	7
100037	6552	4	10.0.0.9	10.0.0.6	129558	134999436	448	981000000	449000000000	7
100038	6552	4	10.0.0.9	10.0.0.6	129558	134999436	448	981000000	449000000000	7
100051	6552	3	10.0.0.6	10.0.0.9	129986	135445412	451	235000000	451000000000	6

Fig 13. Flow6

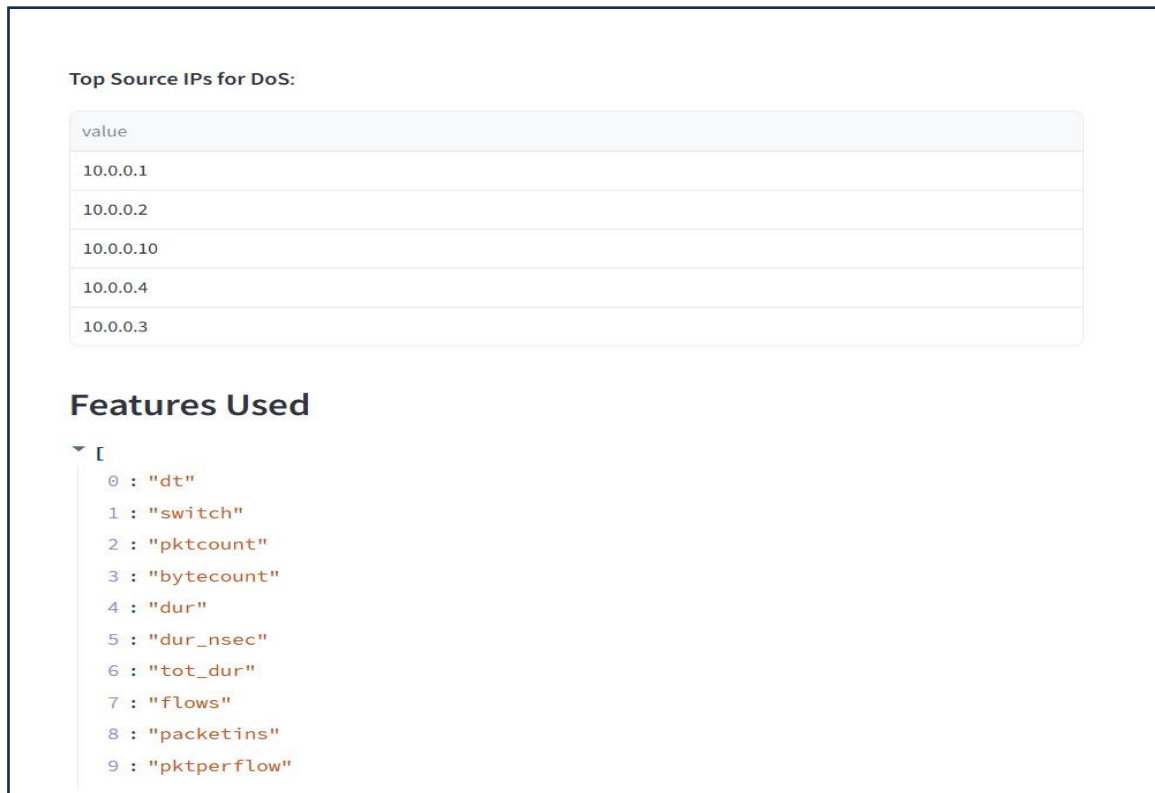


Fig 14. Flow7

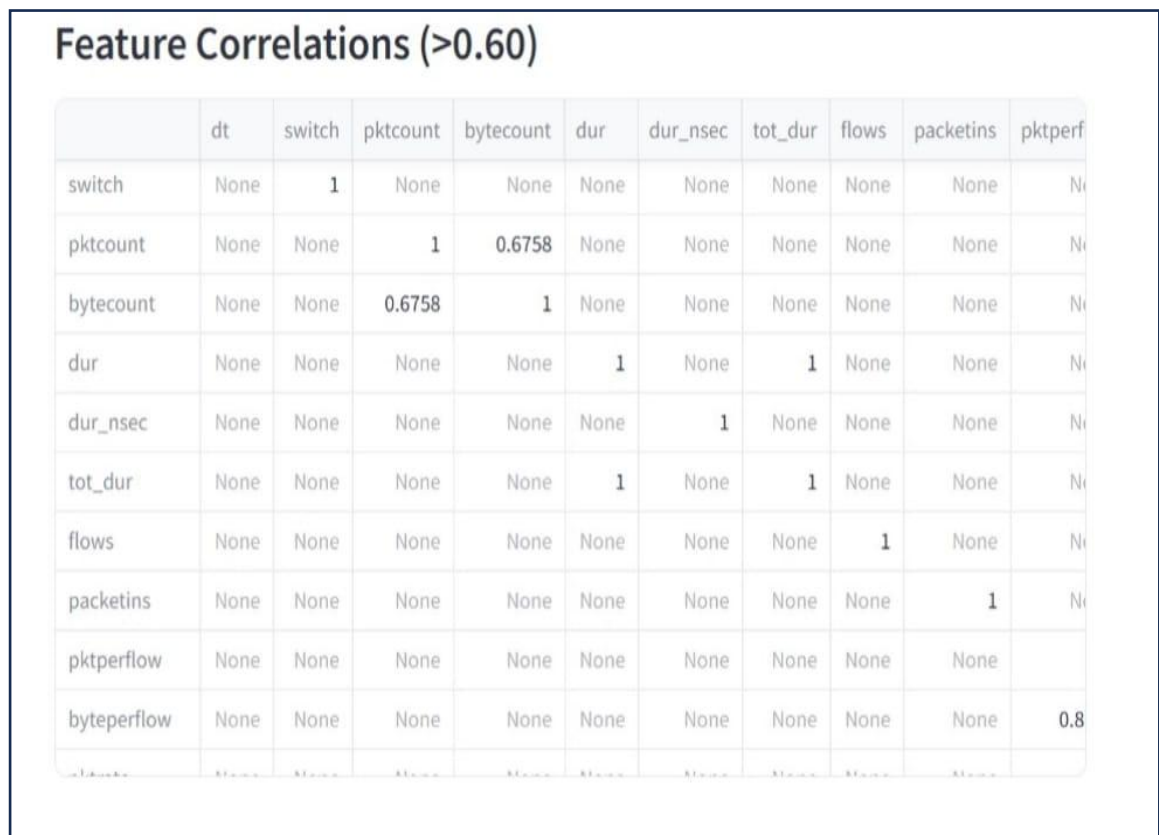


Fig 15. Flow8

Best Model

Best Model: Decision Tree with 100.00% accuracy

Model Accuracy Comparison

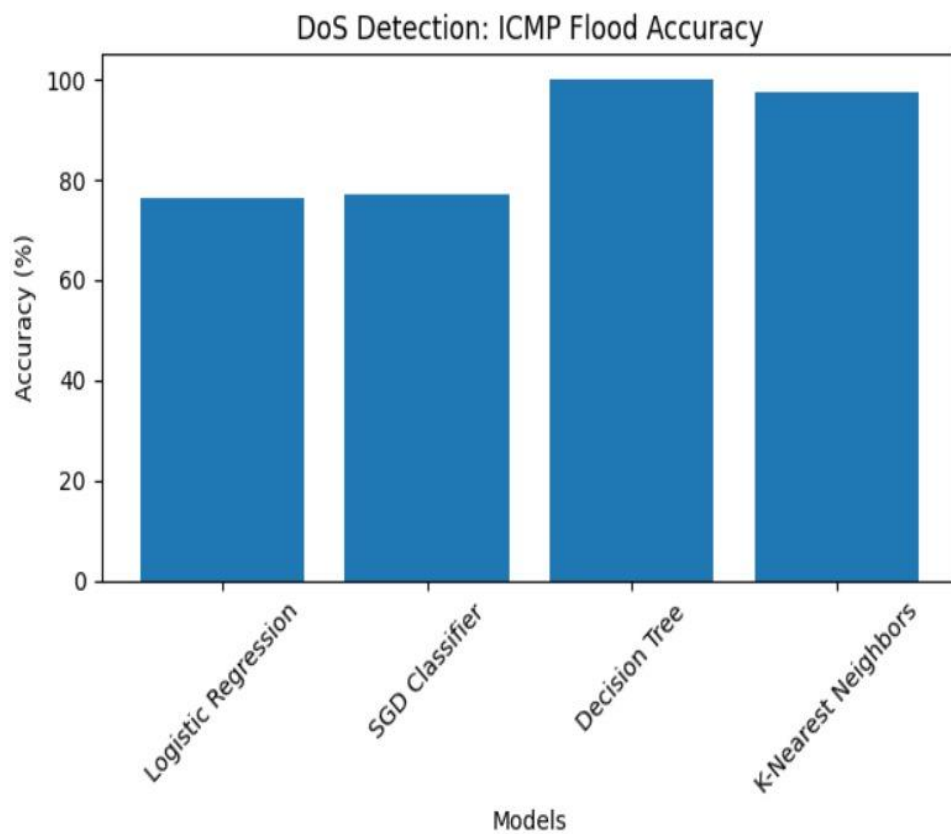


Fig 16. Flow9



Fig 17. Flow10

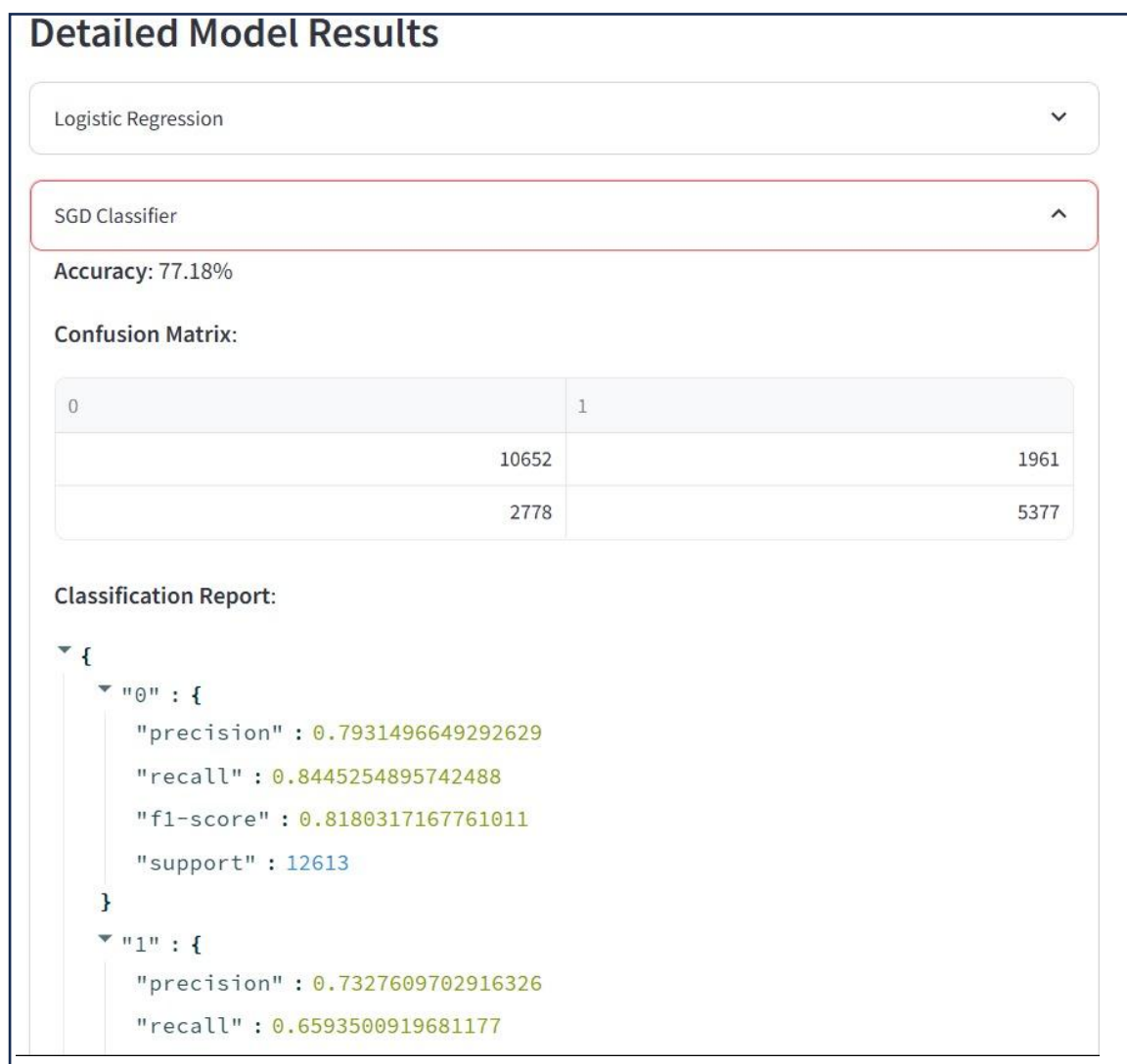


Fig 18. Flow11

ii. Code

Install libraries

```
! pip install streamlit pyngrok -q
```

Import tools

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import streamlit as st
```

```
from pyngrok import ngrok
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.linear_model import SGDClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
import io
```

```
import base64
```

DoS detection function

```
def run_dos_detection():
```

```
    url =  
'https://raw.githubusercontent.com/DrakenWan/DDOS_Detection/master/SDN%20Da  
taset/dataset_sdn.csv'
```

```
    data = pd.read_csv(url)
```

```

data_size = data.shape

missing_data = data.isna().sum()

label_counts = data["label"].value_counts()

dos_rows = data[data['label'] == 1].tail(10)

data = data.dropna()

cleaned_size = data.shape

data = pd.concat([data, pd.get_dummies(data["Protocol"], prefix="Protocol"), axis=1)

dos_ips = data[data['label'] == 1]['src'].unique()[:5]

columns = [i for i in data.columns if data[i].dtype != 'O' and i not in ['src', 'dst']]

features_used = columns

data = data[columns]

label = data["label"]

data = data.drop(["label"], axis=1)

corr_matrix = data.corr()

high_corr = corr_matrix[(corr_matrix > 0.60) == True]

data = data.loc[:, data.nunique() > 1]

scaler = StandardScaler().fit(data)

data_scaled = scaler.transform(data)

train_X, test_X, train_y, test_y = train_test_split(data_scaled, label, test_size=0.2,
random_state=42)

models = {

    "Logistic Regression": LogisticRegression(),

    "SGD Classifier": SGDClassifier(),

    "Decision Tree": DecisionTreeClassifier(),

    "K-Nearest Neighbors": KNeighborsClassifier(n_neighbors=5)

```

```

    }

    results = {}

    scores = []

    for name, model in models.items():

        model.fit(train_X, train_y)

        pred_y = model.predict(test_X)

        score = accuracy_score(test_y, pred_y) * 100

        scores.append(score)

        results[name] = {

            "accuracy": score,

            "confusion_matrix": confusion_matrix(test_y, pred_y),

            "report": classification_report(test_y, pred_y, output_dict=True)

        }

    best_model = list(models.keys())[scores.index(max(scores))]

    best_accuracy = max(scores)

    fig, ax = plt.subplots()

    ax.bar(models.keys(), scores)

    ax.set_xlabel('Models')

    ax.set_ylabel('Accuracy (%)')

    ax.set_title('DoS Detection: ICMP Flood Accuracy')

    plt.xticks(rotation=45)

    plt.tight_layout()

    buf = io.BytesIO()

    fig.savefig(buf, format="png")

    buf.seek(0)

```



```

img_str = base64.b64encode(buf.getvalue()).decode()

return (results, best_model, best_accuracy, img_str, data_size, missing_data,
        label_counts, dos_rows, cleaned_size, features_used, high_corr, dos_ips)

```

Login page function

```

def login_page():

    st.title("Login to DoS Detection App")

    # Project Info

    st.write("### About This Project")

    st.write("This project detects ICMP flood Denial-of-Service (DoS) attacks using
machine learning. It analyzes network traffic with four models (Logistic Regression,
SGD Classifier, Decision Tree, KNN) and displays results like DoS IPs, correlations,
and model accuracy in an interactive web app.")

    # Initialize users dictionary if not in session state

    if 'users' not in st.session_state:

        st.session_state['users'] = {

            "admin": "password123",

            "student": "dosdetect2025",

            "guide": "projectguide"

        }

Login

    username = st.text_input("Username")

    password = st.text_input("Password", type="password")

    if st.button("Login"):

        if username in st.session_state['users'] and st.session_state['users'][username] ==
password:

            st.session_state['logged_in'] = True

            st.success("Logged in successfully!")

```

```

        st.rerun()

    else:

        st.error("Incorrect username or password.")

        st.write("Not registered? [Register here](#register)")

# Registration (shown if login fails or via link)

if 'show_register' not in st.session_state:

    st.session_state['show_register'] = False

if st.session_state['show_register'] or st.button("Register", key="register_link"):

    st.session_state['show_register'] = True

    st.subheader("Register New User")

    new_username = st.text_input("New Username", key="new_username")

    new_password = st.text_input("New Password", type="password",
key="new_password")

    if st.button("Submit Registration"):

        if new_username and new_password and new_username not in
st.session_state['users']:

            st.session_state['users'][new_username] = new_password

            st.success(f"Registered {new_username}! You can now log in.")

            st.session_state['show_register'] = False

            st.rerun()

        else:

            st.error("Username already exists or fields are empty.")

```

Main app function

```

def main_app():

    st.title("DoS Detection: ICMP Flood")

    st.write("Click to detect ICMP flood DoS attacks using machine learning.")

```

```

if st.button("Run Detection"):

    with st.spinner("Running models..."):

        (results, best_model, best_accuracy, img_str, data_size, missing_data,

         label_counts, dos_rows, cleaned_size, features_used, high_corr, dos_ips) =
run_dos_detection()

        st.subheader("Dataset Information")

        st.write("***Data Size***:")

        st.write(data_size)

        st.write("***Missing Data***:")

        st.write(missing_data)

        st.write("***DoS vs Normal Counts***:")

        st.write(label_counts)

        st.write("***Size After Cleaning***:")

        st.write(cleaned_size)

        st.subheader("DoS Attacks")

        st.write("***Last 10 DoS Attacks***:")

        st.write(dos_rows)

        st.write("***Top Source IPs for DoS***:")

        st.write(dos_ips)

        st.subheader("Features Used")

        st.write(features_used)

        st.subheader("Feature Correlations (>0.60)")

        st.write(high_corr)

        st.subheader("Best Model")

        st.success(f"Best Model: {best_model} with {best_accuracy:.2f}% accuracy")

```

```

st.subheader("Model Accuracy Comparison")

st.image(f"data:image/png;base64,{img_str}")

st.subheader("Detailed Model Results")

for name, result in results.items():

    with st.expander(name):

        st.write(f"***Accuracy***: {result['accuracy']:.2f}%")

        st.write("***Confusion Matrix***:")

        st.write(result['confusion_matrix'])

        st.write("***Classification Report***:")

        st.json(result['report'])

if st.button("Logout"):

    st.session_state['logged_in'] = False

    st.rerun()

```

Streamlit app logic

```

if 'logged_in' not in st.session_state:

    st.session_state['logged_in'] = False

if st.session_state['logged_in']:

    main_app()

else:

    login_page()

```

Save Streamlit app

```

with open("app.py", "w") as f:

    f.write("""

import streamlit as st

import pandas as pd

```

Cyber Threat Detection for DOS Attack

```

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.linear_model import SGDClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

import io

import base64

def run_dos_detection():

    url='https://raw.githubusercontent.com/DrakenWan/DDOS_Detection/master/SDN%
20Dataset/dataset_sdn.csv'

    data = pd.read_csv(url)

    data_size = data.shape

    missing_data = data.isna().sum()

    label_counts = data["label"].value_counts()

    dos_rows = data[data['label'] == 1].tail(10)

    data = data.dropna()

    cleaned_size = data.shape

    data = pd.concat([data, pd.get_dummies(data["Protocol"], prefix="Protocol")],
axis=1)

    dos_ips = data[data['label'] == 1]['src'].unique()[:5]

    columns = [i for i in data.columns if data[i].dtype != 'O' and i not in ['src', 'dst']]

    features_used = columns

```

```

data = data[columns]

label = data["label"]

data = data.drop(["label"], axis=1)

corr_matrix = data.corr()

high_corr = corr_matrix[(corr_matrix > 0.60) == True]

data = data.loc[:, data.nunique() > 1]

scaler = StandardScaler().fit(data)

data_scaled = scaler.transform(data)

train_X, test_X, train_y, test_y = train_test_split(data_scaled, label, test_size=0.2,
random_state=42)

models = {

    "Logistic Regression": LogisticRegression(),

    "SGD Classifier": SGDClassifier(),

    "Decision Tree": DecisionTreeClassifier(),

    "K-Nearest Neighbors": KNeighborsClassifier(n_neighbors=5)

}

results = {}

scores = []

for name, model in models.items():

    model.fit(train_X, train_y)

    pred_y = model.predict(test_X)

    score = accuracy_score(test_y, pred_y) * 100

    scores.append(score)

    results[name] = {

        "accuracy": score,

```

```

        "confusion_matrix": confusion_matrix(test_y, pred_y),

        "report": classification_report(test_y, pred_y, output_dict=True)

    }

    best_model = list(models.keys())[scores.index(max(scores))]

    best_accuracy = max(scores)

    fig, ax = plt.subplots()

    ax.bar(models.keys(), scores)

    ax.set_xlabel('Models')

    ax.set_ylabel('Accuracy (%)')

    ax.set_title('DoS Detection: ICMP Flood Accuracy')

    plt.xticks(rotation=45)

    plt.tight_layout()

    buf = io.BytesIO()

    fig.savefig(buf, format="png")

    buf.seek(0)

    img_str = base64.b64encode(buf.getvalue()).decode()

    return (results, best_model, best_accuracy, img_str, data_size, missing_data,

            label_counts, dos_rows, cleaned_size, features_used, high_corr, dos_ips)

def login_page():

    st.title("Login to DoS Detection :")

    st.write("About This Project")

    st.write("This project detects ICMP flood Denial-of-Service (DoS) attacks using machine learning. It analyzes network traffic with four models (Logistic Regression, SGD Classifier, Decision Tree, KNN) and displays results like DoS IPs, correlations, and model accuracy")

```

Initialize users dictionary if not in session state

```

if 'users' not in st.session_state:

    st.session_state['users'] = {

        "admin": "password123",

        "student": "dosdetect2025",

        "guide": "projectguide"

    }

# Login

username = st.text_input("Username")

password = st.text_input("Password", type="password")

if st.button("Login"):

    if username in st.session_state['users'] and st.session_state['users'][username] == password:

        st.session_state['logged_in'] = True

        st.success("Logged in successfully!")

        st.rerun()

    else:

        st.error("Incorrect username or password.")

        st.write("Not registered? [Register here](#register)")

# Registration (shown if login fails or via link)

if 'show_register' not in st.session_state:

    st.session_state['show_register'] = False

if st.session_state['show_register'] or st.button("Register", key="register_link"):

    st.session_state['show_register'] = True

    st.subheader("Register New User")

    new_username = st.text_input("New Username", key="new_username")

```



```

        new_password = st.text_input("New Password", type="password",
key="new_password")

    if st.button("Submit Registration"):

        if new_username and new_password and new_username not in
st.session_state['users']:

            st.session_state['users'][new_username] = new_password

            st.success(f"Registered {new_username}! You can now log in.")

            st.session_state['show_register'] = False

            st.rerun()

        else:

            st.error("Username already exists or fields are empty.")

def main_app():

    st.title("DoS Detection: ICMP Flood")

    st.write("Click to detect ICMP flood DoS attacks using machine learning.")

    if st.button("Run Detection"):

        with st.spinner("Running models..."):

            (results, best_model, best_accuracy, img_str, data_size, missing_data,

            label_counts, dos_rows, cleaned_size, features_used, high_corr, dos_ips) =
run_dos_detection()

            st.subheader("Dataset Information")

            st.write("**Data Size**:")

            st.write(data_size)

            st.write("**Missing Data**:")

            st.write(missing_data)

            st.write("**DoS vs Normal Counts**:")

            st.write(label_counts)

```

```

st.write("**Size After Cleaning**:")

st.write(cleaned_size)

st.subheader("DoS Attacks")

st.write("**Last 10 DoS Attacks**:")

st.write(dos_rows)

st.write("**Top Source IPs for DoS**:")

st.write(dos_ips)

st.subheader("Features Used")

st.write(features_used)

st.subheader("Feature Correlations (>0.60)")

st.write(high_corr)

st.subheader("Best Model")

st.success(f"Best Model: {best_model} with {best_accuracy:.2f}% accuracy")

st.subheader("Model Accuracy Comparison")

st.image(f"data:image/png;base64,{img_str}")

st.subheader("Detailed Model Results")

for name, result in results.items():

    with st.expander(name):

        st.write(f"**Accuracy**: {result['accuracy']:.2f}%")

        st.write("**Confusion Matrix**:")

        st.write(result['confusion_matrix'])

        st.write("**Classification Report**:")

        st.json(result['report'])

if st.button("Logout"):

    st.session_state['logged_in'] = False

```

```

        st.rerun()

if 'logged_in' not in st.session_state:

    st.session_state['logged_in'] = False

if st.session_state['logged_in']:

    main_app()

else:

    login_page()

""")

```

Run Streamlit with ngrok

```

!ngrokauthtoken 2w0JnOUhFWK1JsByYu2LOY52v0y_2gzcWxbDttR64qbG9H1qT
public_url = ngrok.connect(8501)

print(f"Access your DoS detection app at: {public_url}")

!streamlit run app.py --server.port 8501

```

iii. References

- [1] Abiramasundari, S., & Ramaswamy, V. (2025). Distributed denial-of-service (DDoS) attack detection using supervised machine learning algorithms. *Scientific Reports*, 15, 13098. <https://doi.org/10.1038/s41598-024-84879-y>
- [2] Joo, S., Park, S., Shim, H., Oh, Y., & Lee, I. (2025). Machine learning-based detection and selective mitigation of denial-of-service attacks in wireless sensor networks. *Computers, Materials & Continua*, 82(2), 2475–2494. <https://doi.org/10.32604/cmc.2025.058963>
- [3] Becerra-Suarez, F. L., Fernández-Roman, I., & Forero, M. G. (2024). Improvement of Distributed Denial of Service Attack Detection through Machine Learning and Data Processing. *Mathematics*, 12(9), 1294. <https://doi.org/10.3390/math12091294>

- [4] Al-zubidi, A., Farhan, A., & Towfek, S. (2024). Predicting DoS and DDoS attacks in network security scenarios using a hybrid deep learning model. *Journal of Intelligent Systems*, 33(1), 20230195.
<https://doi.org/10.1515/jisys-2023-0195>
- [5] Das, S., Ashrafuzzaman, M., Sheldon, F. T., & Shiva, S. (2024). Ensembling Supervised and Unsupervised Machine Learning Algorithms for Detecting Distributed Denial of Service Attacks. *Algorithms*, 17(3), 99.
<https://doi.org/10.3390/a17030099>
- [6] Alashhab, Z. R., Alcaraz Calero, J. M., & Al-Dubai, A. (2024). A Machine Learning Approach for Detecting and Mitigating DDoS Attacks in Distributed SDN. *IEEE Access*, 12, 2024.
- [7] Kumar, P., Gupta, G. P., & Tripathi, R. (2023). DDoS Attack Detection in IoT Using Machine Learning with Optimized Feature Selection. *Sensors*, 23(5), 2589.
<https://doi.org/10.3390/s23052589>
- [8] Streamlit Documentation (n.d.). Retrieved from <https://streamlit.io>.
- [9] scikit-learn Documentation (n.d.). Retrieved from <https://scikit-learn.org>.
- [10] ngrok Documentation (n.d.). Retrieved from <https://ngrok.com>.
- [11] Wang, Y., Zhang, H., & Li, X. (2024). DDoS-MSCT: A DDoS Attack Detection Method Based on Multiscale Convolution and Transformer. *IET Information Security*, 18(2), 123–134 <https://doi.org/10.1049/2024/1056705>
- [12] Santos-Neto, M. J., Bordim, J. L., & Alchieri, E. A. P. (2024). DDoS Attack Detection in SDN: Enhancing Entropy-Based Detection with Machine Learning. *Concurrency and Computation: Practice and Experience*, 36(11), e8021. <https://doi.org/10.1002/cpe.8021>
- [13] Elsayed, M. S., Le-Khac, N.-A., & Jurcut, A. D. (2020). DDoSNet: A Deep-Learning Model for Detecting Network Attacks. *arXiv preprint arXiv:2006.13981*. <https://doi.org/10.48550/arXiv.2006.13981>
- [14] Rudro, R. A. M., Sohan, M. F. A. A., Chaity, S. K., & Reya, R. I. (2023). Enhancing DDoS Attack Detection Using Machine Learning: A Framework with Feature Selection and Comparative Analysis of Algorithms. <https://doi.org/10.61841/turcomat.v14i03.14086>

[15] Xu, Z. (2025). Deep Learning Based DDoS Attack Detection. ITM Web of Conferences,70,03005. <https://doi.org/10.1051/itmconf/20257003005>