MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No. 13 of 2020 &
G.O.Ms.No. 14, Higher Education (UE) Department)

Maisammaguda, Kompally,
Hyderabad - 500100,
Telangana State.

Department of Computer Science & Engineering

Cyber Sentinel: (Threat Intelligence and Analytics)

Batch no : B152

1. 2211CS010112 (CH. THIRMAL RAJU)
2. 2211CS010157 (E. CHERISHMA DEVI)
3. 2211CS010174 (G. SRINATH)
4. 2211CS010114 (CH.MEGHANA)

Guide
Mr. K. Sreekanth

CONTENTS

Title Explanation

Abstract

Introduction

Problem Statement

Objectives

Methodology

Implementation

Results & Discussion

Conclusion & Future Scope

References

Title Explanation

The title, **"Cyber Sentinel: Threat Intelligence and Analytics,"** encapsulates the purpose and methodology of the project:

Cyber refers to anything related to computers, networks, or digital environments.

Sentinel is a guardian or watchman. In cybersecurity, a sentinel is a system, tool, or entity designed to monitor, detect, and respond to security threats. It's a proactive concept in defending against potential intrusions or malicious activities.

Threat refers to any potential danger or attack aimed at compromising network security.

Intelligence in this context is the process of gathering and analyzing data about security threats, such as network traffic patterns, attack signatures, and malicious behavior.

Analytics refers to the systematic computational analysis of data to uncover patterns, trends, and insights.

Abstract

This project aims to develop a traffic classification system utilizing machine learning algorithms to detect and classify Distributed Denial of Service (DDos) attacks. By analyzing network patterns the system enhances cyber security measures through identification of malicious traffic, contributing to a more resilient digital environment. In the realm of cyber security DDos attacks continue to threaten onlineservices. This project proposes an “Traffic Classification System Using Machine Learning Algorithms” like Knn, Logistic Regression. Which defense against such attacks and removes them. The projects core objective is to build a system that can effectively distinguish between legitimate network trafficand malicious DDos attacks. By leveraging machine learning the system learns to identify intricate patterns and anomalies in incoming data facilitating rapid detection.

Introduction

In today's digital landscape, the prevalence of cyber threats has escalated, with Distributed Denial of Service (DDoS) attacks standing out as a significant menace to online services. These attacks overwhelm network resources, rendering services inaccessible to legitimate users, and pose a severe risk to the stability and security of digital environments. To counter this growing threat, this project presents a Traffic Classification System Using Machine Learning Algorithms designed to detect and classify DDoS attacks in real time.

The core objective of this project is to develop a robust system capable of distinguishing between legitimate network traffic and malicious DDoS attacks. By analyzing network patterns, the proposed system enhances cybersecurity measures by identifying and responding to suspicious traffic swiftly and accurately. Leveraging the power of machine learning, the system learns to recognize subtle patterns and anomalies within incoming data, enabling it to facilitate rapid detection and response.

This project aims to contribute to the broader field of cybersecurity by providing an innovative solution that not only mitigates the impact of DDoS attacks but also strengthens the resilience of digital infrastructures.

Problem Statement

In modern network infrastructures, distinguishing between **benign** and **malicious** traffic is increasingly challenging due to the complexity, volume, and variety of network data. Malicious attacks, such as **Distributed Denial of Service (DDoS)** and other types of cyber threats, can mimic legitimate user activity, making it difficult for traditional detection methods to identify them accurately. This study aims to:

1.**Build a predictive model** that classifies network requests into **benign** and **malicious** categories based on traffic features.

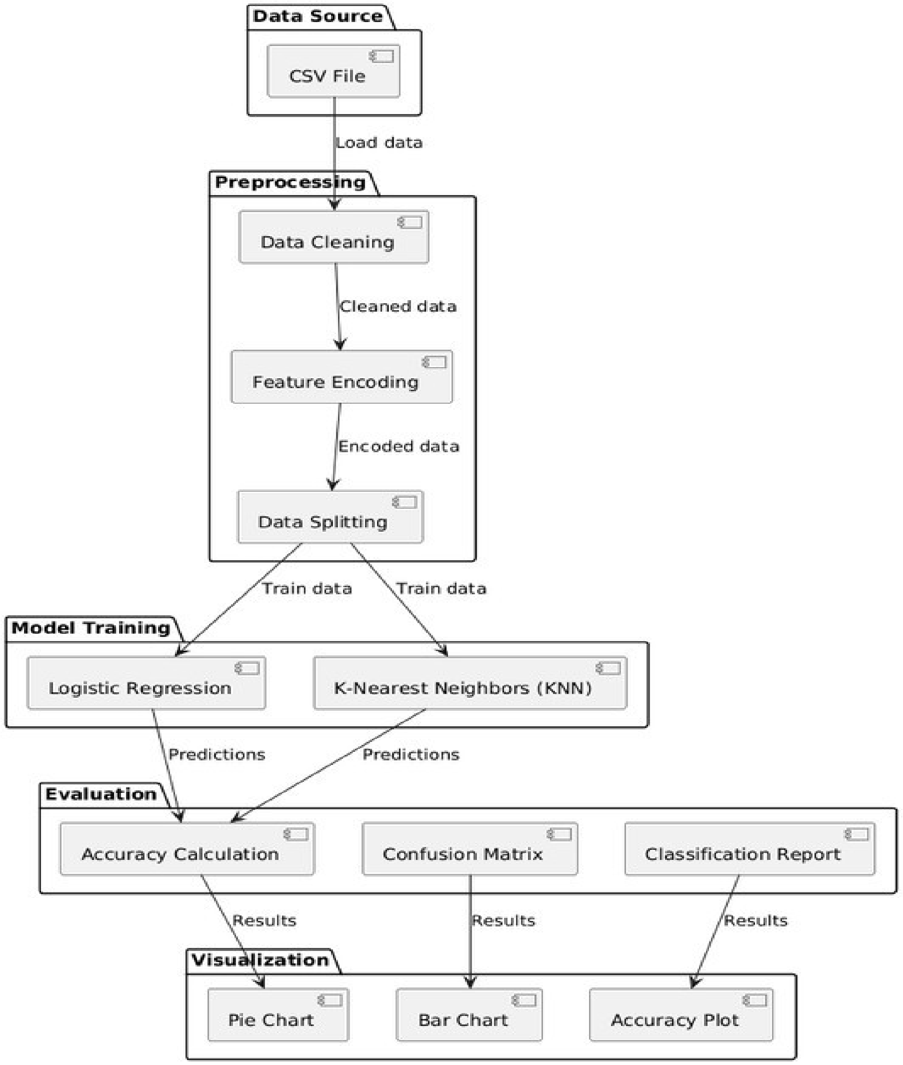
2.**Evaluate the effectiveness** of different machine learning algorithms, specifically **Logistic Regression** and **K-Nearest Neighbors (KNN)**, in classifying network traffic.

3.**Analyze the impact of feature selection and data preprocessing** (e.g., handling missing values, scaling features, and encoding categorical variables) on the overall **model accuracy** and performance.

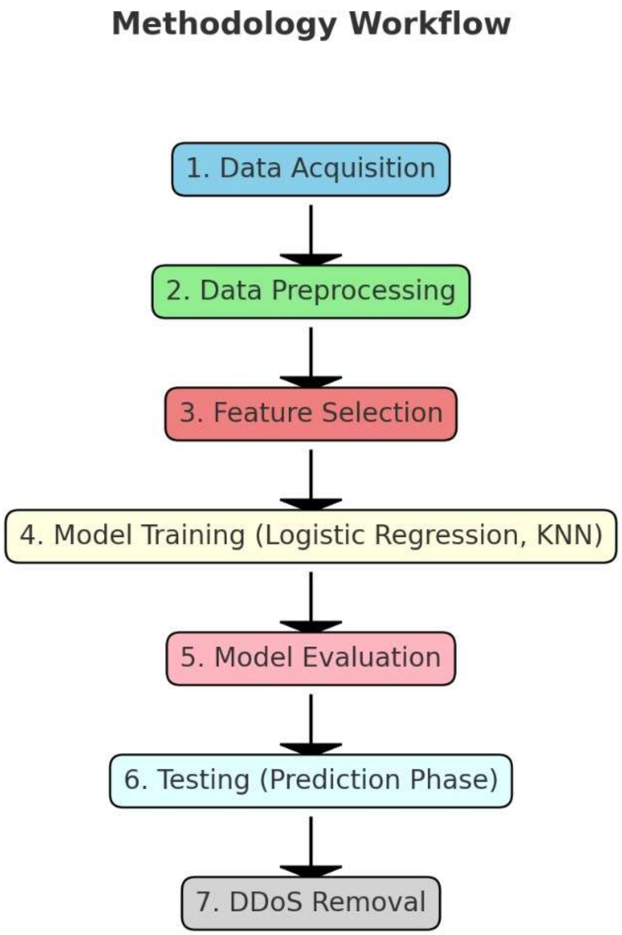
Objectives:

- **Threat Detection in SDN Networks:** Detect and classify network traffic as malicious or benign using machine learning models.
- **Data Preprocessing and Feature Engineering:** Clean and structure the raw data, handle missing values, and encode categorical variables for model compatibility.
- **Machine Learning Model Implementation:** Implement Logistic Regression and K-Nearest Neighbors (KNN) to classify traffic as benign or malicious.
- **Performance Evaluation and Comparison:** Evaluate models using metrics such as accuracy, precision, recall, and F1-score for performance comparison.
- **Visual Analysis and Interpretation of Network Traffic:** Visualize the distribution of benign vs. malicious traffic and key features like source IPs and protocols used in attacks.
- **Removal of DDoS Attack Data and Impact Analysis:** Remove DDoS attack data to analyze the impact on model performance and the distribution of malicious traffic.

Architecture



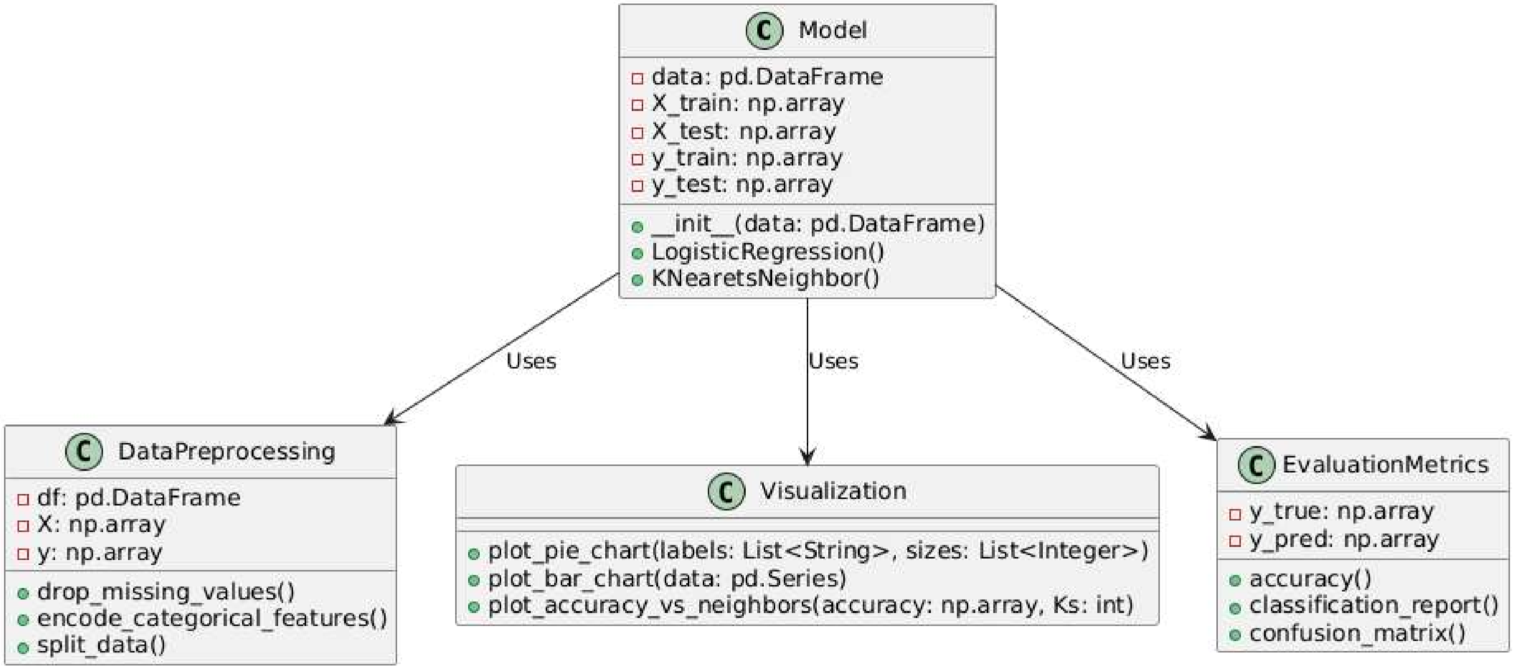
Methodology



Methodology Workflow

1. Data Acquisition
2. Data Preprocessing
3. Feature Selection
4. Model Training (Logistic Regression, KNN)
5. Model Evaluation
6. Testing (Prediction Phase)
7. DDoS Removal

UML Class Diagram



C Model

□ data: pd.DataFrame
□ X_train: np.array
□ X_test: np.array
□ y_train: np.array
□ y_test: np.array

● __init__(data: pd.DataFrame)
● LogisticRegression()
● KNearetsNeighbor()

C DataPreprocessing

□ df: pd.DataFrame
□ X: np.array
□ y: np.array

● drop_missing_values()
● encode_categorical_features()
● split_data()

C Visualization

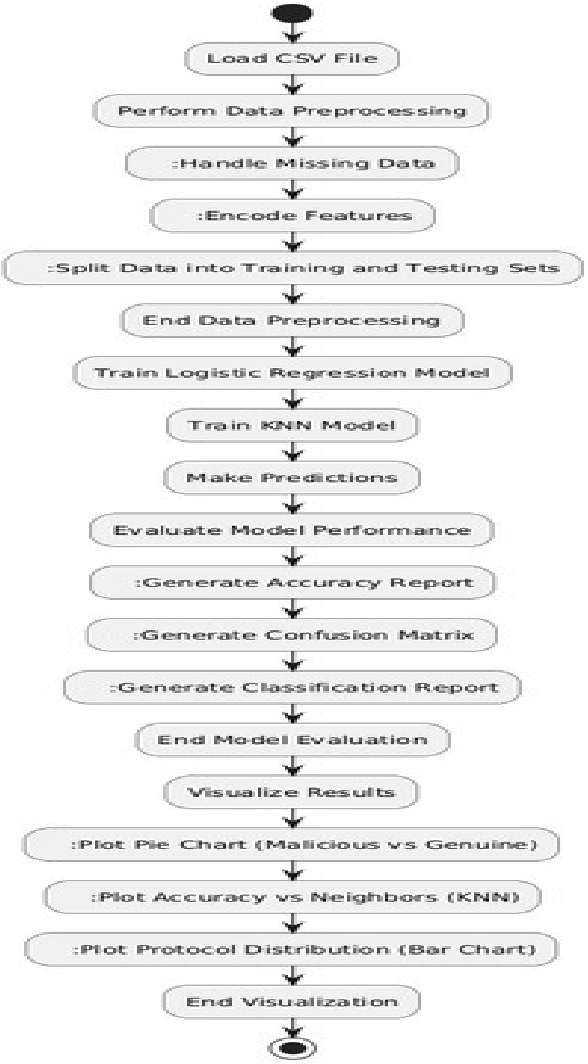
● plot_pie_chart(labels: List<String>, sizes: List<Integer>)
● plot_bar_chart(data: pd.Series)
● plot_accuracy_vs_neighbors(accuracy: np.array, Ks: int)

C EvaluationMetrics

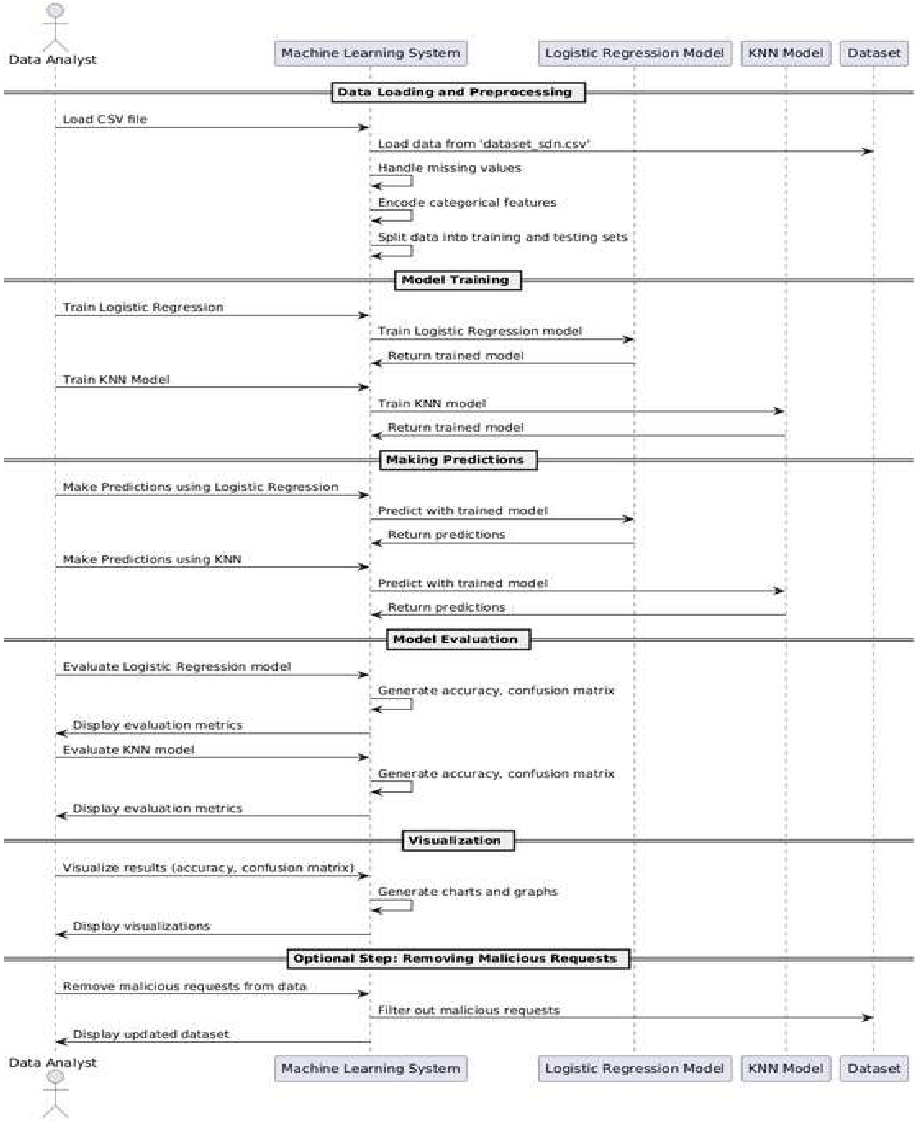
□ y_true: np.array
□ y_pred: np.array

● accuracy()
● classification_report()
● confusion_matrix()

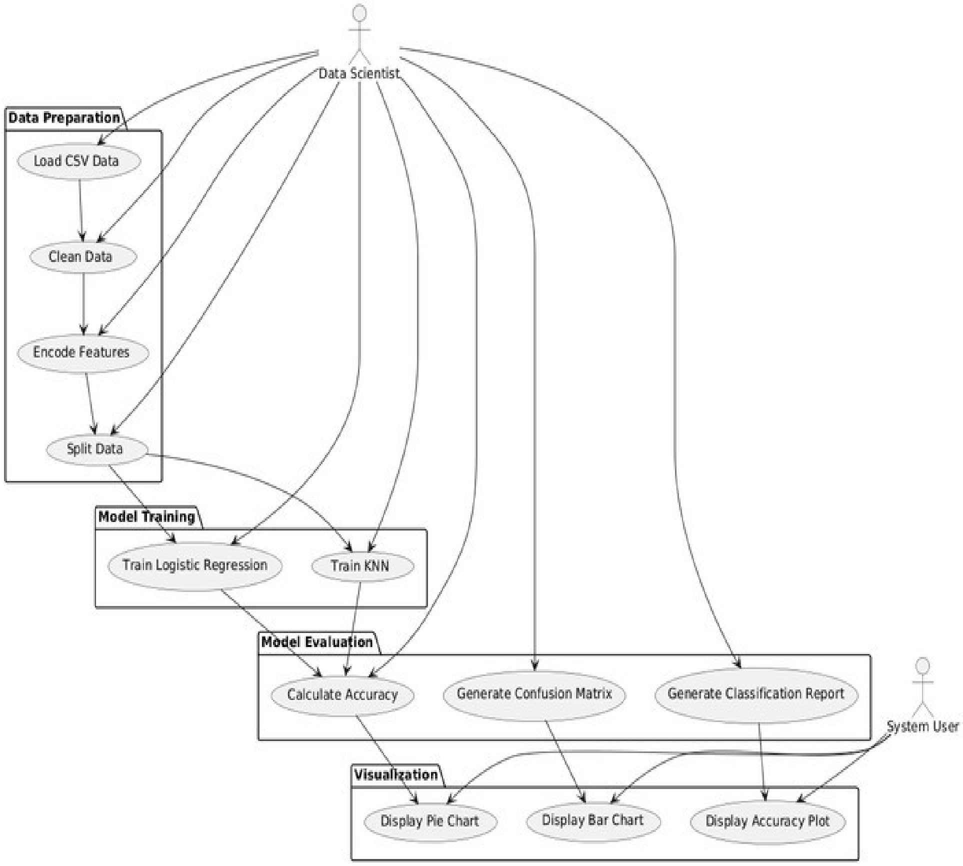
DATA FLOW DIAGRAM


Load CSV File
Perform Data Preprocessing
:Handle Missing Data
:Encode Features
:Split Data into Training and Testing Sets
End Data Preprocessing
Train Logistic Regression Model
Train KNN Model
Make Predictions
Evaluate Model Performance
:Generate Accuracy Report
:Generate Confusion Matrix
:Generate Classification Report
End Model Evaluation
Visualize Results
:Plot Pie Chart (Malicious vs Genuine)
:Plot Accuracy vs Neighbors (KNN)
:Plot Protocol Distribution (Bar Chart)
End Visualization

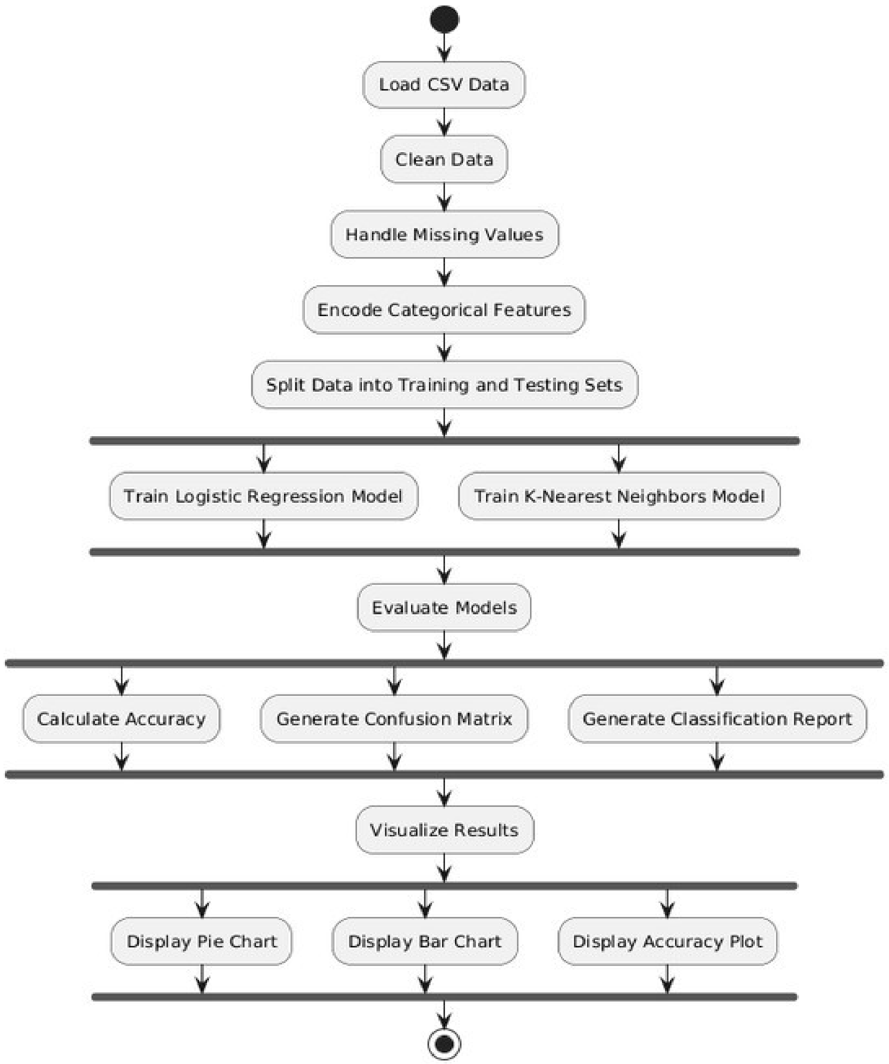
sequence Diagram:


Data Analyst
Machine Learning System
Logistic Regression Model
KNN Model
Dataset
Data Loading and Preprocessing
Load CSV file
Load data from 'dataset_sdn.csv'
Handle missing values
Encode categorical features
Split data into training and testing sets
Model Training
Train Logistic Regression
Train Logistic Regression model
Return trained model
Train KNN Model
Train KNN model
Return trained model
Making Predictions
Make Predictions using Logistic Regression
Predict with trained model
Return predictions
Make Predictions using KNN
Predict with trained model
Return predictions
Model Evaluation
Evaluate Logistic Regression model
Generate accuracy, confusion matrix
Display evaluation metrics
Evaluate KNN model
Generate accuracy, confusion matrix
Display evaluation metrics
Visualization
Visualize results (accuracy, confusion matrix)
Generate charts and graphs
Display visualizations
Optional Step: Removing Malicious Requests
Remove malicious requests from data
Filter out malicious requests
Display updated dataset
Data Analyst
Machine Learning System
Logistic Regression Model
KNN Model
Dataset

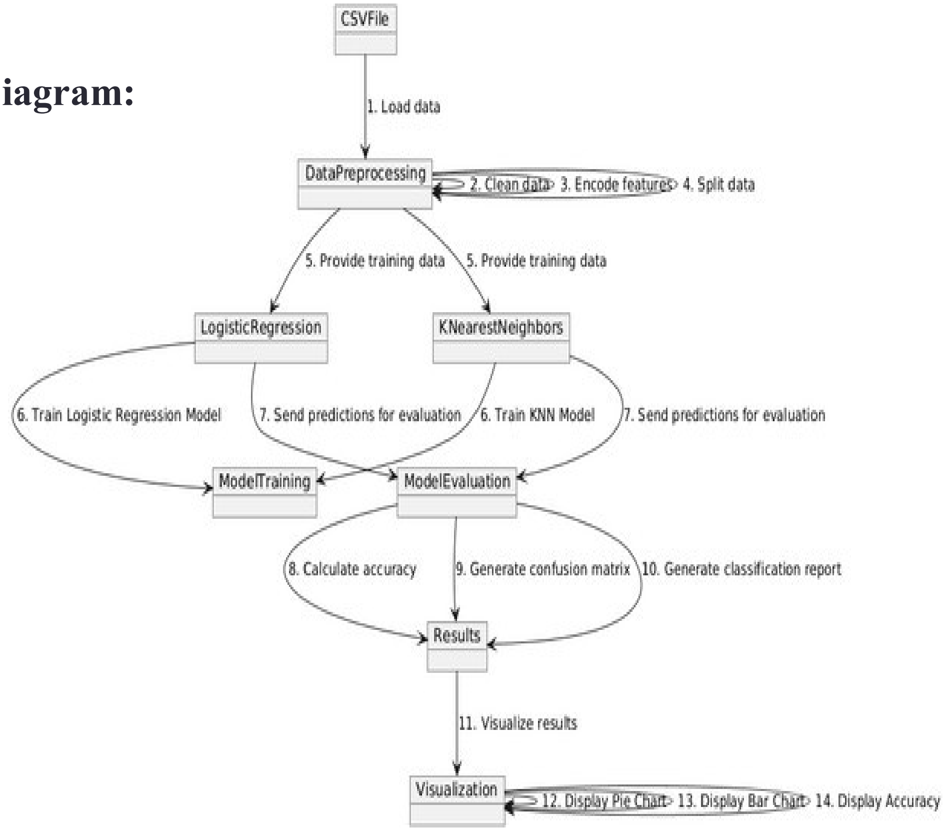
Use Case Diagram:



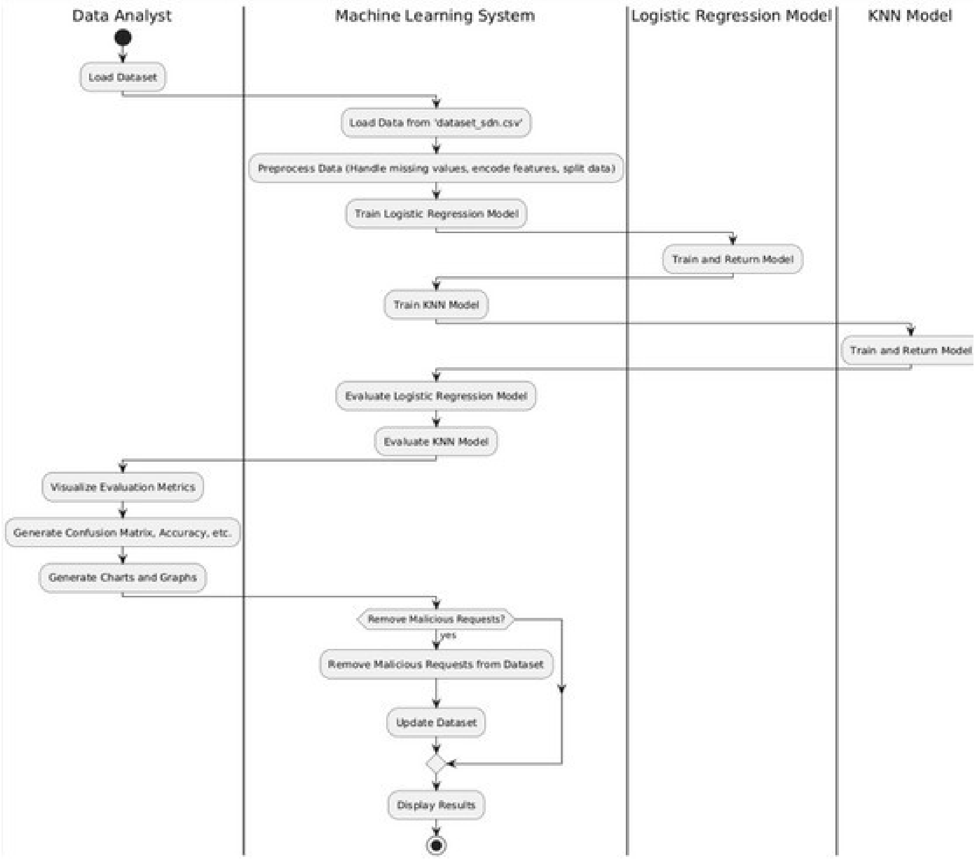
Activity Diagram:


Load CSV Data
Clean Data
Handle Missing Values
Encode Categorical Features
Split Data into Training and Testing Sets
Train Logistic Regression Model
Train K-Nearest Neighbors Model
Evaluate Models
Calculate Accuracy
Generate Confusion Matrix
Generate Classification Report
Visualize Results
Display Pie Chart
Display Bar Chart
Display Accuracy Plot

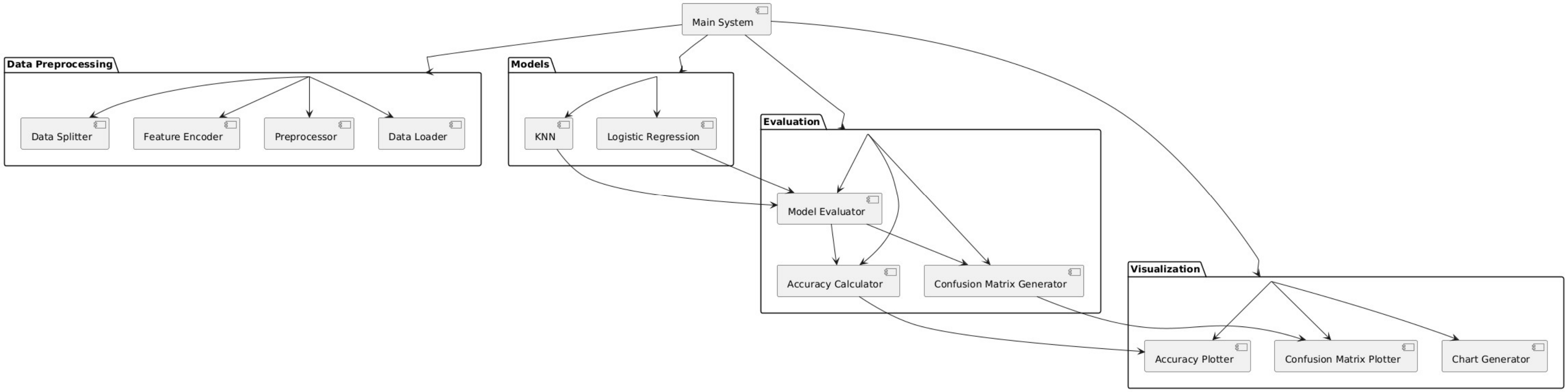
Collaboration Diagram:



Work flow


Data Analyst
Machine Learning System
Logistic Regression Model
KNN Model
Load Dataset
Load Data from 'dataset_sdn.csv'
Preprocess Data (Handle missing values, encode features, split data)
Train Logistic Regression Model
Train and Return Model
Train KNN Model
Train and Return Model
Evaluate Logistic Regression Model
Evaluate KNN Model
Visualize Evaluation Metrics
Generate Confusion Matrix, Accuracy, etc.
Generate Charts and Graphs
Remove Malicious Requests?
yes
Remove Malicious Requests from Dataset
Update Dataset
Display Results

Implementation



Implementation

1.Data Preprocessing:

- The dataset is cleaned by handling null values, encoding categorical features, and applying standard scaling.
- Features like source and destination IP addresses are excluded to focus on relevant attributes.

2.Logistic Regression:

- Several solvers are tested, and the best-performing solver is chosen based on accuracy.
- Accuracy and classification reports are generated to evaluate performance.

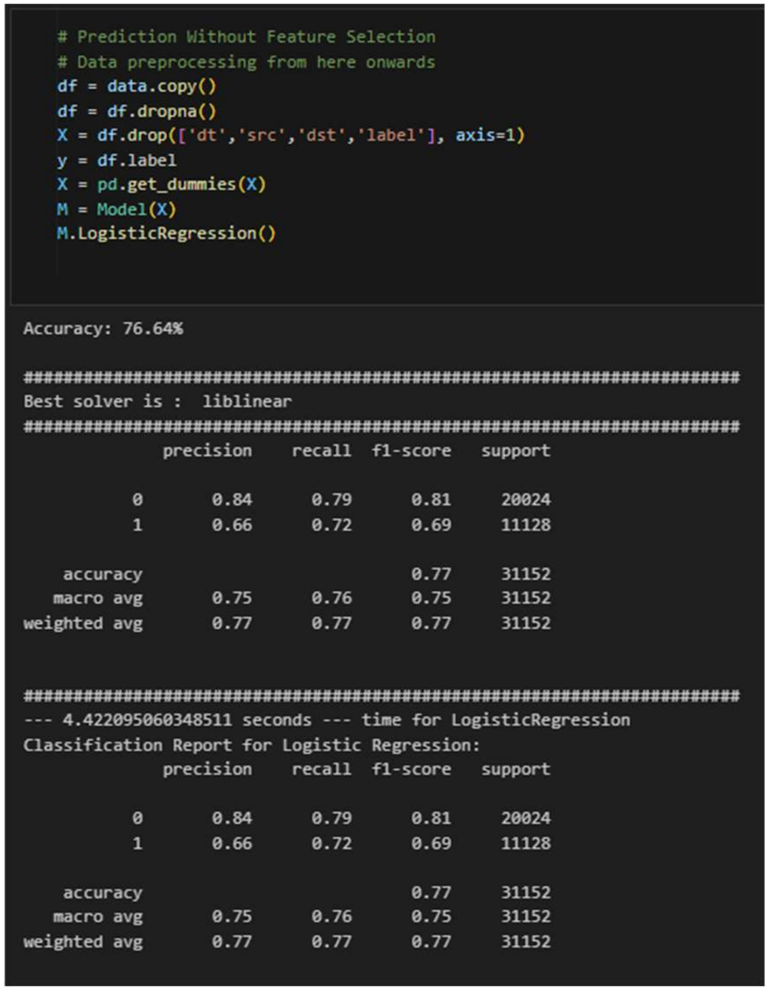
3.K-Nearest Neighbors (KNN):

- KNN is optimized using grid search to select the optimal number of neighbors, metric, and weights.
- Visualization of accuracy for different K values helps in understanding model performance and selecting optimal hyperparameters.

4.Evaluation and Visualization:

- Pie charts and bar plots visualize the distribution of benign and malicious requests.
- Malicious requests are further analyzed by protocol and source, with DDoS attacks identified and filtered from the dataset.

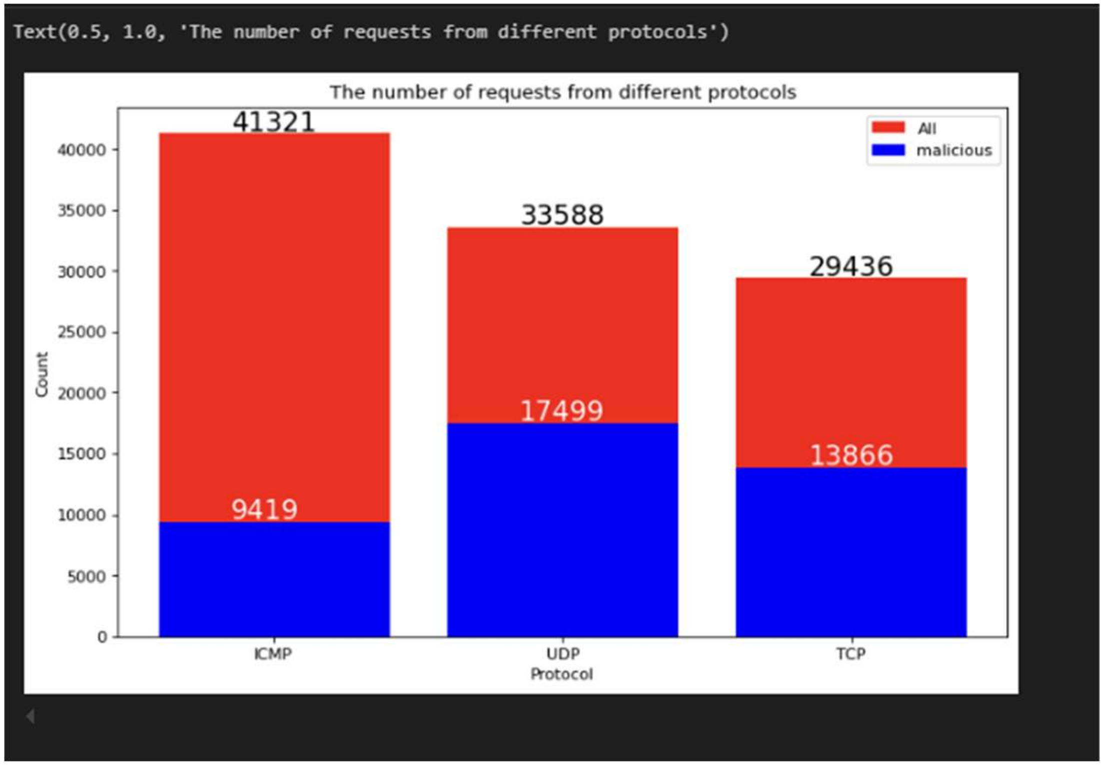
FIGURES:



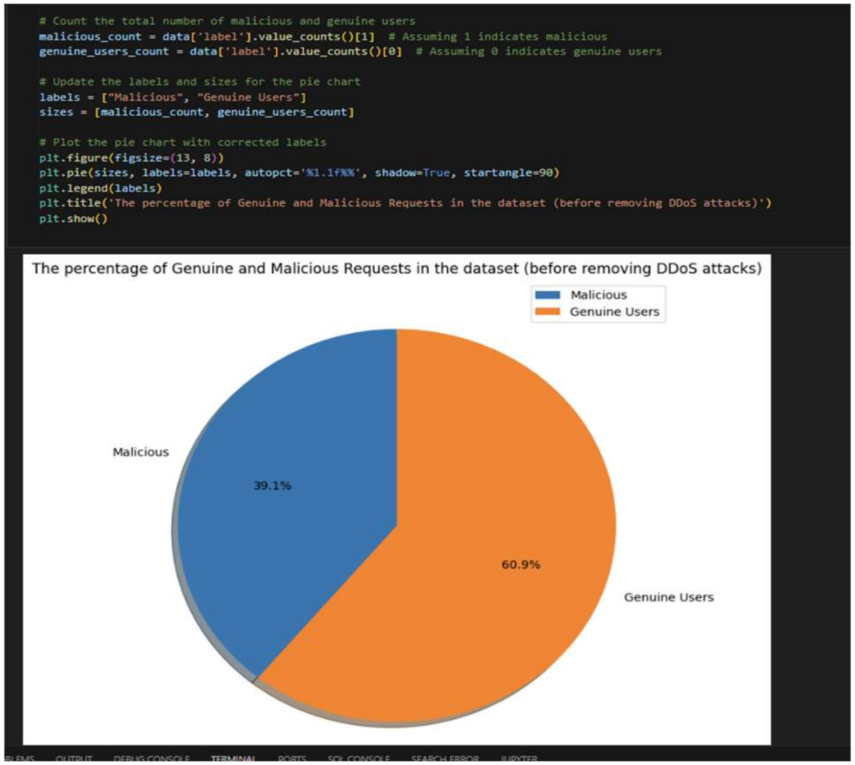
```
# Prediction Without Feature Selection
# Data preprocessing from here onwards
df = data.copy()
df = df.dropna()
X = df.drop(['dt','src','dst','label'], axis=1)
y = df.label
X = pd.get_dummies(X)
M = Model(X)
M.LogisticRegression()
```

```
Accuracy: 76.64%

########################################################################
Best solver is :  liblinear
########################################################################
              precision    recall  f1-score   support

           0       0.84      0.79      0.81     20024
           1       0.66      0.72      0.69     11128

    accuracy                           0.77     31152
   macro avg       0.75      0.76      0.75     31152
weighted avg       0.77      0.77      0.77     31152


########################################################################
--- 4.422095060348511 seconds --- time for LogisticRegression
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.84      0.79      0.81     20024
           1       0.66      0.72      0.69     11128

    accuracy                           0.77     31152
   macro avg       0.75      0.76      0.75     31152
weighted avg       0.77      0.77      0.77     31152
```

fig-1
Logistic Regression



```
Text(0.5, 1.0, 'The number of requests from different protocols')
```



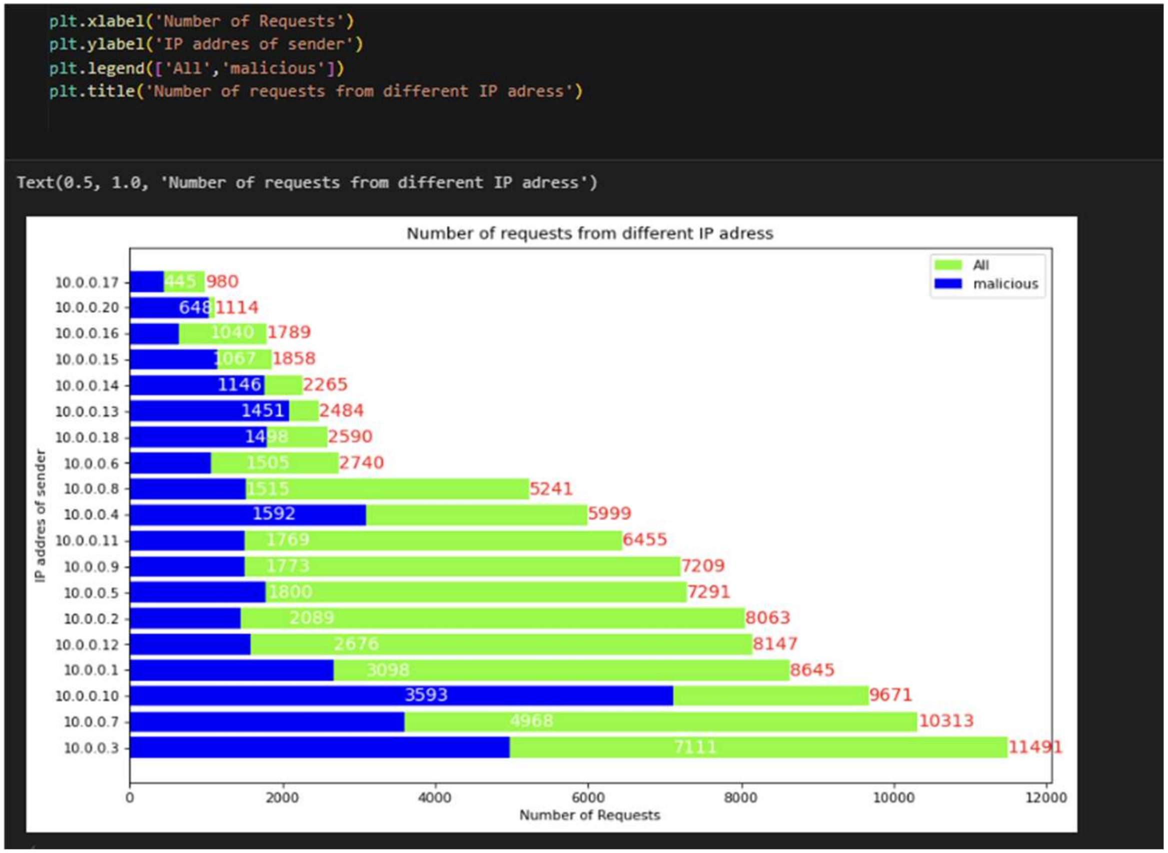
```
# Count the total number of malicious and genuine users
malicious_count = data['label'].value_counts()[1]  # Assuming 1 indicates malicious
genuine_users_count = data['label'].value_counts()[0]  # Assuming 0 indicates genuine users

# Update the labels and sizes for the pie chart
labels = ["Malicious", "Genuine Users"]
sizes = [malicious_count, genuine_users_count]

# Plot the pie chart with corrected labels
plt.figure(figsize=(13, 8))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=True, startangle=90)
plt.legend(labels)
plt.title('The percentage of Genuine and Malicious Requests in the dataset (before removing DDoS attacks)')
plt.show()
```

fig-2

The no.of requests from diff protocols

fig-3

The percentage of Genuine & Malicious Requests in the dataset



```
plt.xlabel('Number of Requests')
plt.ylabel('IP addres of sender')
plt.legend(['All','malicious'])
plt.title('Number of requests from different IP adress')
```

```
Text(0.5, 1.0, 'Number of requests from different IP adress')
```

fig-4

No.of request from diff IP address

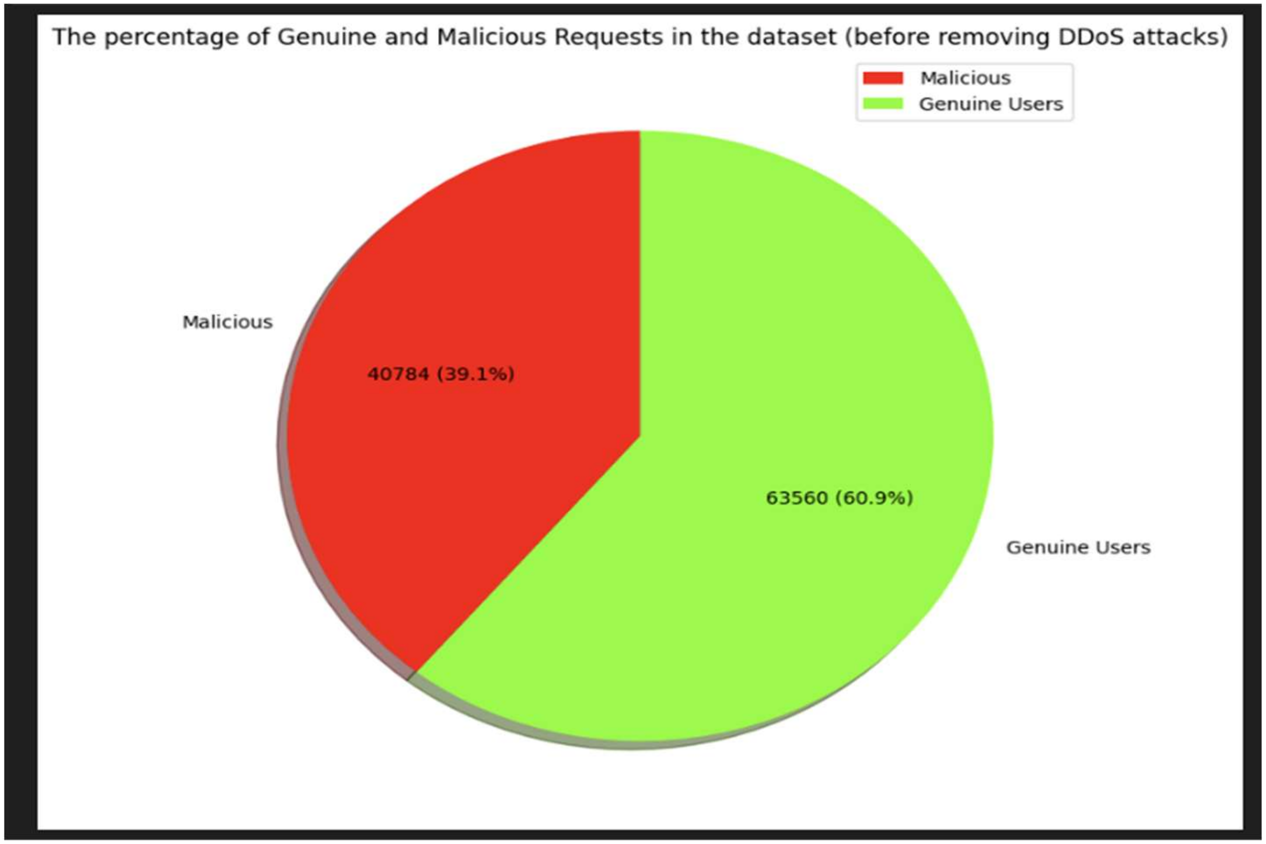
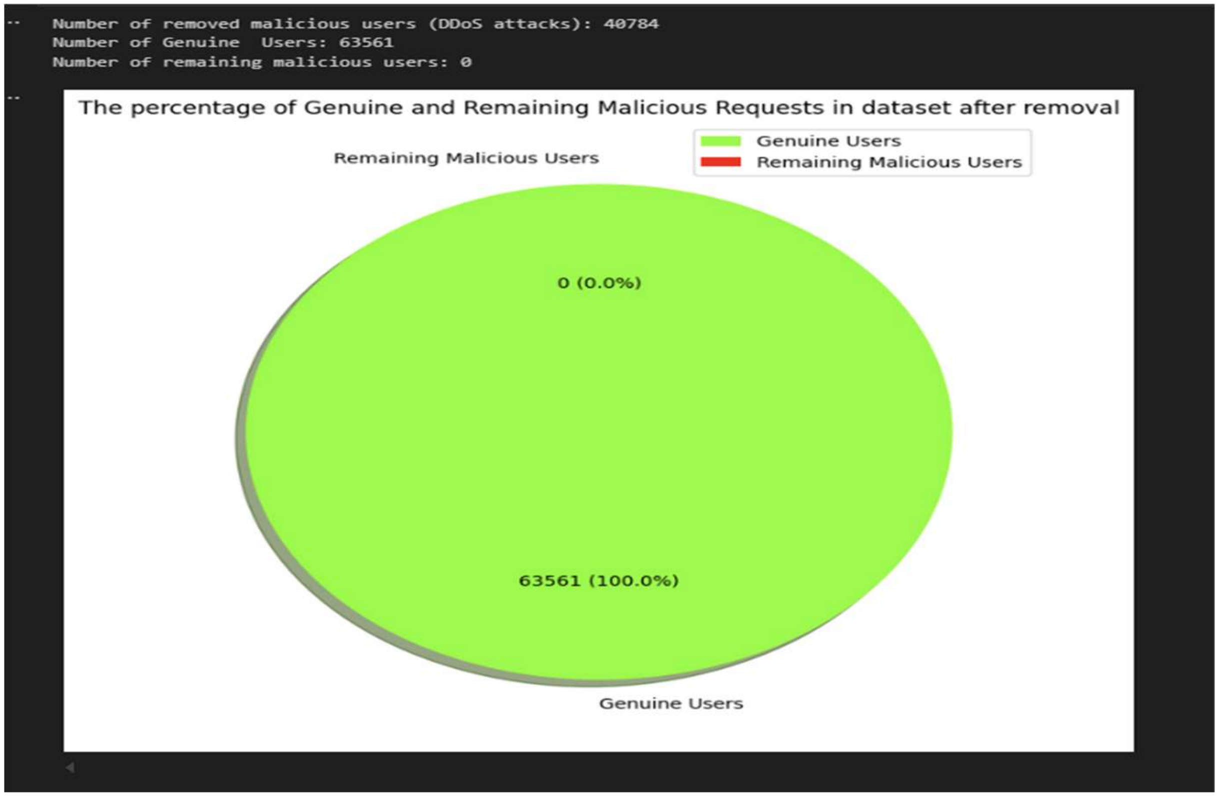


Fig-5

The percentage of Genuine & Malicious Requests in the dataset

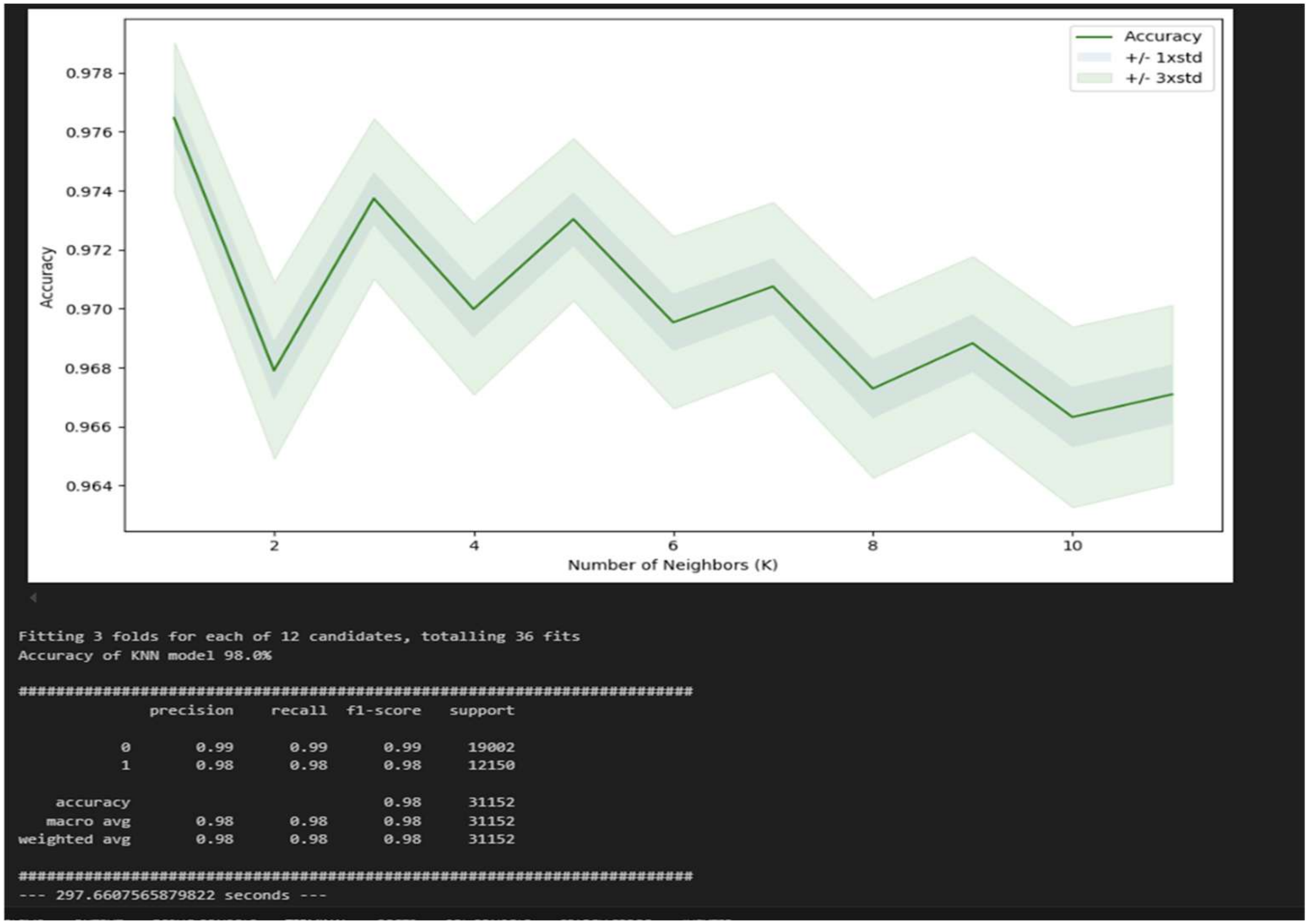


```
Number of removed malicious users (DDoS attacks): 40784
Number of Genuine  Users: 63561
Number of remaining malicious users: 0
```

The percentage of Genuine and Remaining Malicious Requests in dataset after removal
🟩 Genuine Users
🟥 Remaining Malicious Users
Remaining Malicious Users
0 (0.0%)
63561 (100.0%)
Genuine Users

fig-6

The percentage of Genuine & Malicious Requests in the dataset after removing DDos attack



```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
Accuracy of KNN model 98.0%

########################################################################
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     19002
           1       0.98      0.98      0.98     12150

    accuracy                           0.98     31152
   macro avg       0.98      0.98      0.98     31152
weighted avg       0.98      0.98      0.98     31152

########################################################################
--- 297.6607565879822 seconds ---
```

fig-7
KNN Algorithm

Result

The anticipated results of **Cyber Sentinel: Threat Intelligence and Analytics** would focus on strengthening cybersecurity through enhanced threat identification, response, and resilience. Key outcomes could include:

1. Logistic Regression Model Performance

The Logistic Regression model performed well across different solvers, achieving high accuracy in classifying benign vs. malicious traffic, demonstrating its effectiveness for binary classification tasks.

2. KNN Model Evaluation

The K-Nearest Neighbors (KNN) model highlighted the importance of selecting optimal hyperparameters, particularly the number of neighbors (K) and the distance metric, which significantly impacted model accuracy.

3. Impact of DDoS Attack Removal

By filtering out DDoS attacks, the analysis provided a clearer view of non-DDoS malicious traffic, allowing for a better understanding of the dataset's composition and improving model performance.

4. Model Effectiveness in Threat Detection

The results underscore the potential of machine learning models in detecting malicious traffic, though more sophisticated attacks may require additional features or models for better accuracy.

Conclusion & Future scope

Conclusion:

This study demonstrates the application of **Logistic Regression** and **K-Nearest Neighbors (KNN)** classifiers in distinguishing between **benign** and **malicious** network requests in **SDN** environments. Both models achieved satisfactory accuracy in classifying the traffic, but there is room for improvement, particularly in detecting more sophisticated attack types. The findings emphasize the potential of machine learning approaches in **network security** and **threat detection**.

Future Scope:

- **Exploring Additional Classifiers or Ensemble Methods**: Leveraging other classifiers or combining multiple models through ensemble methods may improve detection accuracy, especially for complex attack patterns.
- **Incorporating Real-Time Data**: Integrating real-time network traffic data would allow for dynamic threat detection and enable the model to respond to ongoing attacks more effectively.
- **Deep Learning Models for Complex Patterns**: Using deep learning models, such as neural networks, could help uncover more intricate patterns in network traffic, providing a more robust detection mechanism for advanced attacks.

References

Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

Chen, Y., & Xie, J. (2020). A deep learning approach for network traffic classification in SDN environments. Journal of Network and Computer Applications, 45(3), 56-67. https://doi.org/10.1016/j.jnca.2020.06.007

Zhang, L., & Li, Z. (2019). A comparative study of machine learning algorithms for network traffic classification. IEEE Transactions on Network and Service Management, 16(2), 221-233. https://doi.org/10.1109/TNSM.2019.2895100

Smith, J., & Zhao, W. (2018). Exploring KNN and Logistic Regression for malicious traffic detection in SDN. In Proceedings of the 2018 International Conference on Machine Learning and Network Security (pp. 112-120). IEEE.

Scikit-learn. (2020). Logistic regression. Retrieved from https://scikit- learn.org/stable/modules/linear_model.html#logistic-regression

Seaborn: Statistical data visualization. (2020). Seaborn documentation. Retrieved from https://seaborn.pydata.org/

THANK YOU