National College of Ireland

Configuration Manual for Enhancing Cybersecurity through AI-Driven Threat Detection Systems

MSc Research Project
Cyber Security

Harini Srinivasulu
Student ID: 23187921

School of Computing
National College of Ireland

Supervisor: Eugene Mclaughlin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Harini Srinivasulu
Student ID:	23187921
Programme:	MSc in Cyber Security **Year:** 2024
Module:	Enhancing Cyber Security through AI- Driven Threat Detection Systems
Lecturer:	Eugene Mclaughlin
Submission Due Date:	12-12-2024
Project Title:	Enhancing Cyber Security through AI- Driven Threat Detection Systems
Word Count:	706 **Page Count:** 10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Harini

Date: 12-12-2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual for Enhancing Cybersecurity through AI-Driven Threat Detection Systems

Harini Srinivasulu
Student ID: 23187921

1. Introduction

This manual provides a comprehensive guide for implementing a machine learning-based threat detection system using the Network Intrusion Dataset. The goal of this research is to leverage various machine learning models to classify network traffic as either benign or malicious, specifically focusing on Distributed Denial of Service (DDoS) attacks.

2. System Hardware Requirements

- Processor (CPU):
  - Intel Core i7 or AMD Ryzen 7 (8 cores, 16 threads minimum)
  - Preferred: Intel Core i9 or AMD Ryzen 9 for better performance.
- Memory (RAM):
  - Minimum: 16 GB
  - Recommended: 32 GB or more for faster model training, especially for deep learning tasks.
- Storage:
  - Minimum: 500 GB SSD for storing datasets, model outputs, and logs.
  - Recommended: 1 TB SSD or higher for extensive datasets and efficient I/O performance.
- GPU (for deep learning models):
  - Minimum: NVIDIA GTX 1060 / AMD equivalent for basic model training.
  - Recommended: NVIDIA RTX 3060 or higher (e.g., RTX 3080/3090) for faster neural network training.
- Operating System: Windows 10/11, macOS, or Linux (Ubuntu 20.04 or newer preferred).

3. Software Requirements:

- Python 3.7 or newer
- pandas (for data handling), numpy (for numerical operations).
- matplotlib, seaborn (for data visualization and plotting).
- scikit-learn (for model building, classification, evaluation).
- tensorflow and keras (for deep learning model development).
- sklearn.metrics (for performance metrics such as accuracy, confusion matrix).
- warnings (for ignoring warnings), LabelEncoder, SimpleImputer, MinMaxScaler, StandardScaler, SelectKBest.
- Jupyter Notebook, Visual Studio Code, or PyCharm (for better debugging and development experience).
- Anaconda or Miniconda (for environment management and package installation).

4. Dataset Details

Dataset Source:

The dataset used in this project is from Kaggle, specifically the "Network Intrusion Dataset". It contains network traffic data for different types of attacks (e.g., DDoS, PortScan, and more) labeled as either BENIGN or DDoS. The dataset is useful for training machine learning models for intrusion detection and cybersecurity analysis.

Dataset Link: Network Intrusion Dataset on Kaggle

Dataset Description:

- The dataset consists of various network traffic features like flow duration, packet length, packet count, flags, and other network attributes, making it suitable for detecting different attack types.
- The data is collected in different scenarios and reflects real-time network traffic patterns, which includes both normal (BENIGN) and malicious (DDoS) behavior.

File Format: CSV file: Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv
Features: Includes features such as flow duration, packet length, flow bytes/s, total forward packets, total backward packets, active time, idle time, and TCP flags.

5. Execution of the Code Implementation

Import Required Libraries

Start by importing the essential libraries required for data manipulation, model building, and visualization. The following libraries are used:

```
#import the necessary libaries
import pandas as pd   # For data manipulation and analysis
import numpy as np    # For numerical operations
import matplotlib.pyplot as plt # For basic data visualizationn
import seaborn as sns  # For enhanced data visualization
from sklearn.preprocessing import LabelEncoder #For label encoding categorical variable
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split #For splitting the data into train and test
from sklearn.neighbors import KNeighborsClassifier #KNN Classifier
from sklearn.ensemble import RandomForestClassifier # Random forest Classifier
from sklearn.linear_model import LogisticRegression #Logistic Regression Classifier
from sklearn.tree import DecisionTreeClassifier #Decision Tree classifier
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix # For model evaluation
from sklearn.metrics import roc_curve, auc #roc_curve plot

import warnings
warnings.filterwarnings('ignore')
```

Load the Dataset

The dataset can be downloaded from Kaggle and loaded into a pandas DataFrame. This will allow you to explore and preprocess the data.

```
#fetch the dataset
dataFrame = pd.read_csv('Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv')
```

```
#columns in the dataset
dataFrame.columns
```

```
Index([' Destination Port', ' Flow Duration', ' Total Fwd Packets',
       ' Total Backward Packets', 'Total Length of Fwd Packets',
       ' Total Length of Bwd Packets', ' Fwd Packet Length Max',
       ' Fwd Packet Length Min', ' Fwd Packet Length Mean',
       ' Fwd Packet Length Std', 'Bwd Packet Length Max',
       ' Bwd Packet Length Min', ' Bwd Packet Length Mean',
       ' Bwd Packet Length Std', 'Flow Bytes/s', ' Flow Packets/s',
       ' Flow IAT Mean', ' Flow IAT Std', ' Flow IAT Max', ' Flow IAT Min',
       'Fwd IAT Total', ' Fwd IAT Mean', ' Fwd IAT Std', ' Fwd IAT Max',
       ' Fwd IAT Min', 'Bwd IAT Total', ' Bwd IAT Mean', ' Bwd IAT Std',
       ' Bwd IAT Max', ' Bwd IAT Min', 'Fwd PSH Flags', ' Bwd PSH Flags',
       ' Fwd URG Flags', ' Bwd URG Flags', ' Fwd Header Length',
       ' Bwd Header Length', 'Fwd Packets/s', ' Bwd Packets/s',
       ' Min Packet Length', ' Max Packet Length', ' Packet Length Mean',
       ' Packet Length Std', ' Packet Length Variance', 'FIN Flag Count',
       ' SYN Flag Count', ' RST Flag Count', ' PSH Flag Count',
       ' ACK Flag Count', ' URG Flag Count', ' CWE Flag Count',
       ' ECE Flag Count', ' Down/Up Ratio', ' Average Packet Size',
       ' Avg Fwd Segment Size', ' Avg Bwd Segment Size',
       ' Fwd Header Length.1', 'Fwd Avg Bytes/Bulk', ' Fwd Avg Packets/Bulk',
       ' Fwd Avg Bulk Rate', ' Bwd Avg Bytes/Bulk', ' Bwd Avg Packets/Bulk',
       'Bwd Avg Bulk Rate', 'Subflow Fwd Packets', ' Subflow Fwd Bytes',
       ' Subflow Bwd Packets', ' Subflow Bwd Bytes', 'Init_Win_bytes_forward',
       ' Init_Win_bytes_backward', ' act_data_pkt_fwd',
       ' min_seg_size_forward', 'Active Mean', ' Active Std', ' Active Max',
       ' Active Min', 'Idle Mean', ' Idle Std', ' Idle Max', ' Idle Min',
       ' Label'],
      dtype='object')
```

Explore the Dataset

Before performing any preprocessing, understand the dataset by inspecting the first and last few rows, dataset info, and basic statistics:

```
# View the first 5 rows of the dataset
dataFrame.head()
```

Python

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	...	min_seg_size_for
0	54865	3	2	0	12	0	6	6	6.0	0.0	...	
1	55054	109	1	1	6	6	6	6	6.0	0.0	...	
2	55055	52	1	1	6	6	6	6	6.0	0.0	...	
3	46236	34	1	1	6	6	6	6	6.0	0.0	...	
4	54863	3	2	0	12	0	6	6	6.0	0.0	...	

5 rows × 79 columns

Data Cleaning and Preprocessing

- Handle missing values: Replace NaN with zeros or use imputation techniques.
- Handle infinities: Replace any infinite values with NaN, then impute or drop.

```
# Replace infinities with NaN
dataFrame.replace([np.inf, -np.inf], np.nan, inplace=True)
# Replace NaNs with 0
dataFrame.fillna(0, inplace=True)
```

- Remove duplicates: Drop duplicate rows to prevent skewing the analysis.

```
print('Number of duplicates are : ', dataFrame.duplicated().sum())
```

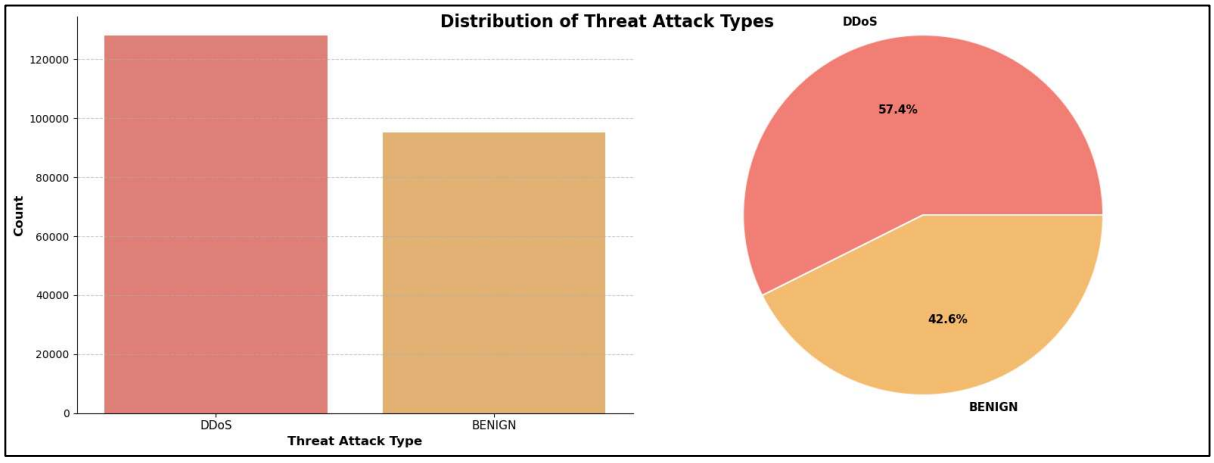
```
Number of duplicates are :  2633
```

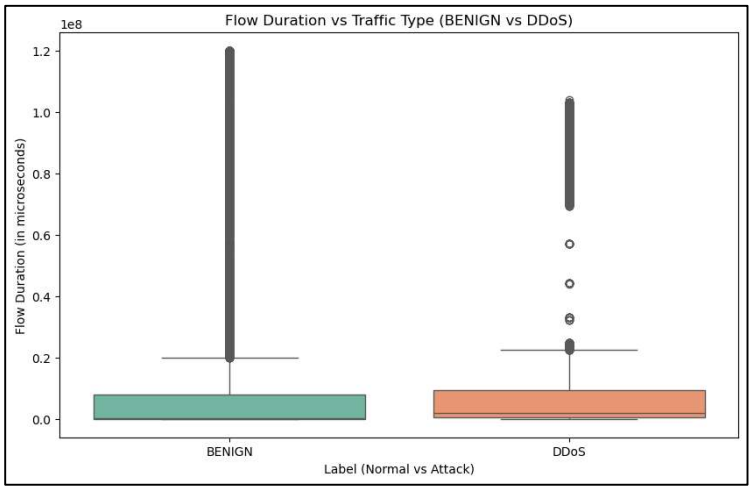
```
# Remove duplicates from the DataFrame
dataFrame = dataFrame.drop_duplicates()

# Check the number of duplicates after removal
print('Number of duplicates are : ', dataFrame.duplicated().sum())
```

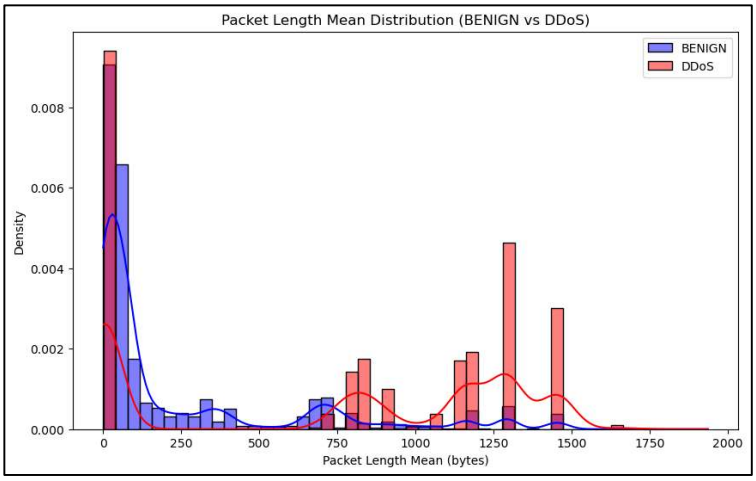
```
Number of duplicates are :  0
```

Exploratory Data Analysis (EDA)

Perform an exploratory analysis to gain insights into the dataset. This includes visualizations of attack types, flow duration, packet length distribution, and more.







Feature Engineering

- Label Encoding: Convert categorical labels into numerical values.

```
#encode the string objects to the categorical values to numerical values
encoder = {}
for i in dataFrame.select_dtypes('object').columns:
    encoder[i] = LabelEncoder()
    dataFrame[i] = encoder[i].fit_transform(dataFrame[i])
```

- Feature Selection: Use SelectKBest to select the top features.

```
# Impute missing values (replace NaNs with the mean)
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(x)

# Determine the number of columns (features) in your DataFrame
num_columns = dataFrame.shape[1]

# Set an appropriate value for k (less than or equal to the number of columns)
k = min(10, num_columns)

# Initialize SelectKBest with the scoring function
k_best = SelectKBest(score_func=f_classif, k=k)

# Fit and transform the imputed data to select the top 10 features
X_new = k_best.fit_transform(X_imputed, y)
```

Split the Data
Split the dataset into training and testing sets (80% training, 20% testing).

```
# Split the data into features (X) and labels (y)
X = dataFrame[selected_feature_names].values
y= dataFrame[' Label'].values
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20,random_state=0)
```

```
print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)
```

```
Training Features Shape: (178489, 10)
Training Labels Shape: (178489,)
Testing Features Shape: (44623, 10)
Testing Labels Shape: (44623,)
```

Model Initialization & Training
Train several machine learning models and evaluate their performance:

- K-Nearest Neighbors Classifier (KNN):

```
#Initialize and train one of the K-Nearest Neighbors Model for Classification of Threat attack
knn_classifier = KNeighborsClassifier(n_neighbors=10000)
knn_classifier.fit(X_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10000)
```

- Decision Tree Classifier:

```
# Initialize and train one of the Decision Tree Model for Classification of Threat attack
dt_classifier = DecisionTreeClassifier(max_depth=2)
dt_classifier.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2)
```

- Logistic Regression Classifier:

```
#Initialize and train one of the (Logistic Regression Model) for Classification of threat attack
logistic_regression_model = LogisticRegression(n_jobs=61)
logistic_regression_model.fit(X_train,y_train)
```

```
LogisticRegression
LogisticRegression(n_jobs=61)
```

- Random Forest Classifier:

```
#Initialize and train one of the (Random Forest Model) for Classification of Threat attack
random_forest_model = RandomForestClassifier(max_depth=3)
random_forest_model.fit(X_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=3)
```

- Deep Neural Network (DNN):

```
#Initialize and train one of the (Deep Neural Network Model) for Classification of Threat attack
dnn_model = Sequential()
dnn_model.add(Dense(128, input_dim=X_train.shape[1], activation='relu'))
dnn_model.add(Dense(64, activation='relu'))
dnn_model.add(Dense(32, activation='relu'))
dnn_model.add(Dense(1, activation='sigmoid'))  # For binary classification; use 'softmax' for multiclass
```

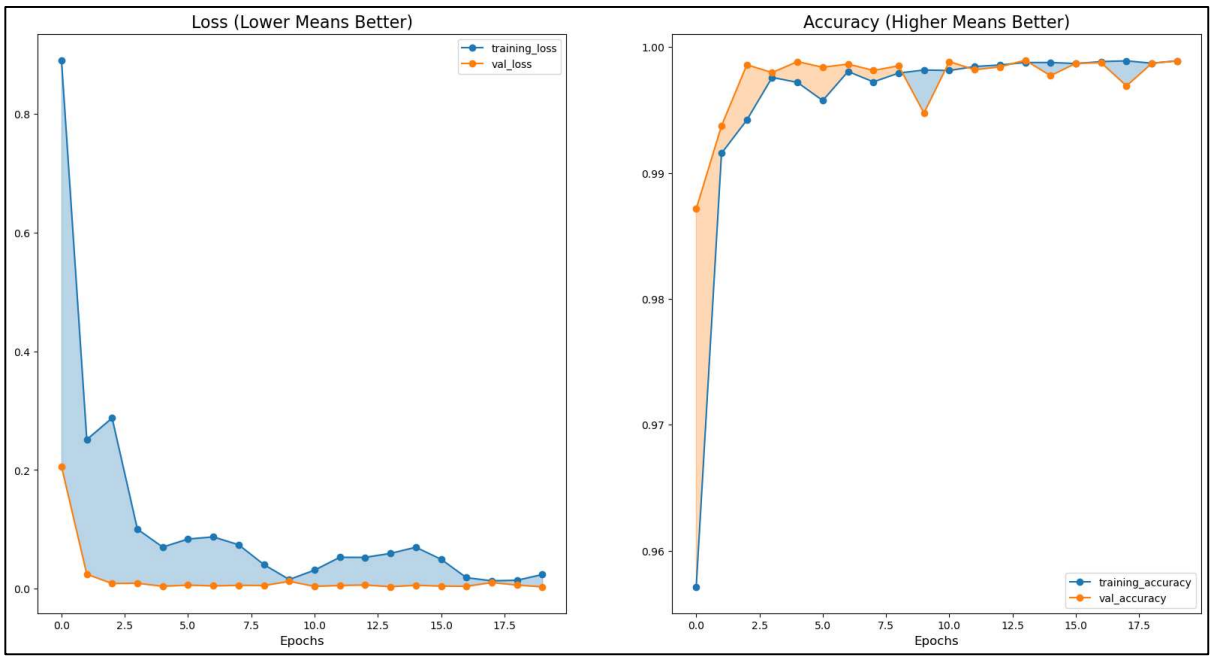
Python

```
# Compile the model
dnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Python

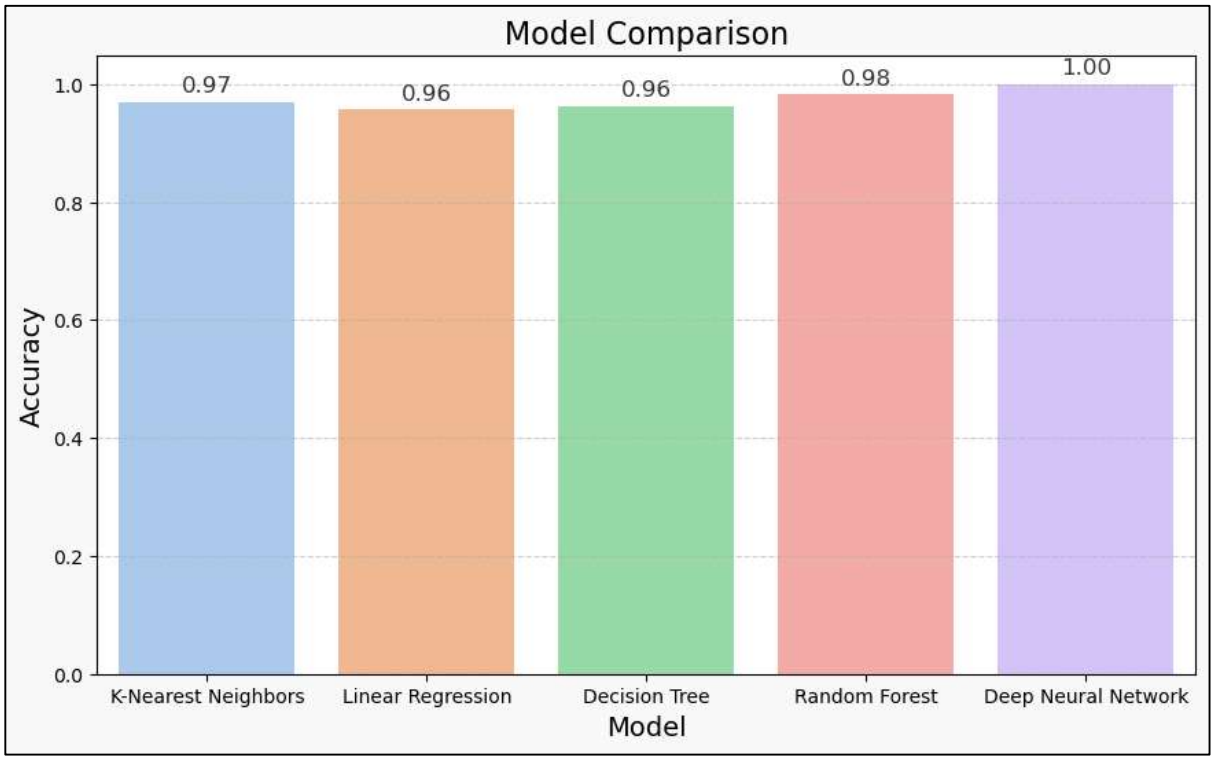
```
# Train the model
history = dnn_model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2, verbose=1)
```

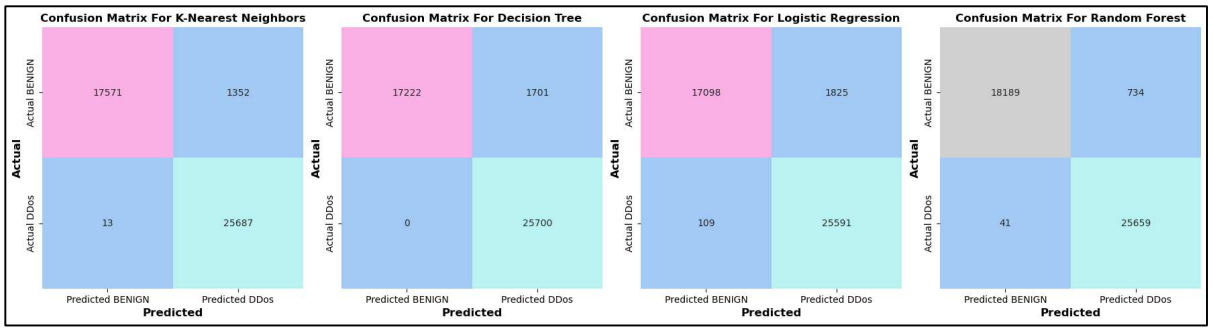
Python



Model Comparison

After training all the models, compare their performance metrics (accuracy, precision, recall, F1 score, etc.) to determine which model performs the best for detecting DDoS attacks.





This manual provides the necessary steps to load, preprocess, analyze, and model the Network Intrusion Dataset using machine learning techniques. By following these steps, you can run the code implementation for a robust threat detection system to classify network traffic as benign or DDos.

References

Python: https://www.python.org
Dataset Source: Network Intrusion Dataset on Kaggle